Efficient representation of ducts and cluttered spaces for realistic motion planning of hyper-redundant robots through confined paths

K. P. Ashwin *, A. N. Chaudhury *, and A. Ghosal †

Abstract

Application of highly articulated hyper-redundant robots to manoeuvre in narrow and confined spaces is gaining popularity due to their obvious advantages. In this paper, we describe an optimization based approach for motion planning of hyper-redundant robots, which results in a natural motion of the links through ducts and confined spaces. It is shown that for a desired motion of the end-effector or the head of the hyper-redundant robot, the motion of the subsequent links attenuate and all the links avoid collision with the walls of the ducts and any other obstacles in the confined spaces. We discuss several ways to represent ducts in 2D and 3D space and also how the proposed algorithm is applied in these representations. It is shown that the complexity of the algorithm, with m constraints is at most $\mathcal{O}(m^{3.5})$ and in case where the ducts can be modeled with polyhedra, the complexity can be as low as $\mathcal{O}(m^{1.5})$. The proposed approach is also used to determine the largest link length in the hyper-redundant robot which can traverse the confined path. The concepts developed in this paper are demonstrated using simulations conducted on three practical scenarios: 1) hyper-redundant manipulators inspecting an industrial pipeline, 2) motion of an endoscopic robot through gastro-intestinal (GI) tract and 3) motion of hyper-redundant manipulators in search and rescue operations. Analysis on the computational complexity and the simulations shows that the method is feasible for practical implementation.

Keywords: Ducts and cluttered spaces, hyper-redundant robots, optimization, motion planning, simulation

*Graduate Student at the Robotics and Design Lab, Department of Mechanical Engineering, Indian Institute of Science, Bangalore 560012, India, email: ashwinkp,arkadeepc@iisc.ac.in

†Corresponding Author, Professor, Department of Mechanical Engineering, Indian Institute of Science, Bangalore, email: asitava@iisc.ac.in.

1 Introduction

A serial robot with more than 6 joints in 3D space is considered to be a redundant system – for a given position and orientation of end-effector, there can be infinite number of joint values with which a robotic end-effector (tool) can achieve its desired position and orientation. A *hyper-redundant* robot, as defined by Chirikjian and Burdick [1], is a robot with large ($n >> 6$) kinematic redundancy. These robots consist of either a large number of small links connected in series or a series of continuous flexible beams joined to each other. In operations, these robots approximate the shape of tentacles, elephant trunks, snakes etc. One of the main advantages of a hyper-redundant robot is that it can be used for avoiding obstacles or for easily maneuvering in cluttered and confined spaces. As a result, hyper-redundant robots have been suggested in applications such as exploring earthquake hit regions for search and rescue operations [2, 3], remotely operated robotic arms for minimally invasive laparoscopic and endoscopic surgeries [4, 5, 6, 7]and inspection of industrial pipelines and nuclear reactors [1, 8, 9, 10, 11]. Identifying a suitable configuration from the many possible configurations, given the position and orientation of the end-effector constitutes the redundancy resolution of the hyper-redundant robots. If the trajectory which the end-effector has to trace is predetermined, then the redundancy resolution problem also constitutes the motion planning of the robot.

There exists extensive literature on motion planning of redundant robots. The most basic approach uses the pseudo-inverse of the manipulator Jacobian matrix (see, for example, [12], [13], [14]). Since the major advantage of using a hyper-redundant robot is to facilitate motion avoiding obstacles and movement in confined spaces, many researchers have also worked on this particular task using the pseudo-inverse [15]. However, the method becomes computationally complex for large number of joints [16]. Many other redundancy resolution methods including variational approach, geometric approach, neural networks and fuzzy logic can also be found in the literature [17, 18, 19, 20]. In [1], obstacle avoidance problem for hyper-redundant manipulator is carried out by fitting a curve through the joints of manipulator and planning the path for this 'backbone curve' which avoids obstacles. Finding the pose of the backbone curve directly gives the co-ordinates of the joints. Hence, in case of obstacle cluttered environments, planning the path of the end-effector which avoids the obstacles will result in redundancy resolution since the trajectory itself forms the backbone curve of the manipulator [21, 22, 23, 24]. Even though the backbone curve approach is very simple, end-effector trajectory with many kinks may sometimes produce high acceleration at the tail of the link which is undesirable and also the motion will look un-natural. In [25], the

authors proposed a tractrix based redundancy resolving algorithm which generates natural motion of a hyper-redundant robot. The realism is imparted from the characteristics of the algorithm that from the end where the displacement is specified, the motion attenuates as it proceeds to the other links as if the robot moves in a damped environment. In [26], the authors have showed that the tractrix based realism can also be achieved by posing the same problem as a minimization problem. Apart from redundancy resolution of robots, the approach has other interesting applications such as animating flexible objects such as string, hair, rope etc. for CAD as well as motion simulators.

In [27], it has been shown that the minimization based approach can be easily implemented for obstacle avoidance of the entire link chain, when the obstacles are represented by simple analytical shapes. However, in the special case of motion through confined spaces within a narrow bounded path (called as ducts in this work), direct implementation of the algorithm is non-trivial, since 1) except for simple shapes, analytical formulation of a duct is not always possible, and 2) directly implementing the algorithm would require modeling the entire half space outside the duct as obstacles, which is impractical. In this paper, we discuss the implementation of the minimization-based scheme for redundancy resolution of a hyper-redundant robot moving in ducts. We present three methods each, to represent ducts in 2D as well as 3D and discuss how the minimization-based approach can be implemented in different representations. All the methods discussed have certain advantages and disadvantages in terms of modelling time, solving time and ease of implementation. The paper is organized as follows: in section 2, for completeness, the tractrix based algorithm is discussed. Representation of ducts in 2D and 3D and motion planning through the ducts is detailed in section 3. In section 4, we illustrate the concepts and algorithms developed in this work by simulating the motion of hyper-redundant robots in three practical applications. The details of optimization based approach, its computational complexity and a few practical limitations are mentioned in section 5. Finally, in section 6, we present the conclusions of the paper and the scope of future work.

2 Overview of tractrix based motion planning

The motion planning problem addressed in this paper corresponds to finding a particular configuration of a hyper-redundant robot, when the trajectory of the leading tip of the robot is known or prescribed. In this section, we present the basic concepts of the tractrix based approach which is utilized to plan the motion (resolve redundancy) of a hyper-redundant robot (see references [25, 26] for more details).

Consider a rigid link of length L_0 positioned in a 2D plane, initially aligned to the $Y-$ axis as shown in figure 1a, with the 'head' being moved along the $X-$ axis. The initial location of the 'tail' is at $\mathbf{X}_{t0} = [X_t, Y_t]_0^T = [0, L_0]^T$ and the initial location of the head is at $\mathbf{X}_{h0} = [X_h, Y_h]_0^T = [0, 0]^T$. Let the coordinates of the 'tail' be denoted by $\mathbf{x}_t = [x_t(p), y_t(p)]^T$ when the head moves to $\mathbf{X}_h = [p, 0]^T$. If the head is given an infinitesimal displacement dp along the $X-$ axis (to $\mathbf{x}_h = [x_h, y_h]$ as shown in figure 1a), the displacement dr of the 'tail' and hence the magnitude of the velocity is least when it is along the link (the figure shows two more exaggerated infinitesimal displacements to illustrate the point).

The curve traced by the 'tail' point, denoted by $\mathbf{x}_t = [x_t, y_t]^T$, when we use the condition that the velocity of the tail (or the tangent to the curve) should always be along the link is called the tractrix. The tractrix equation can be derived from the tangent of the curve given by

$$\frac{dy}{dx} = -\frac{y}{\sqrt{L_0^2 - y^2}} \tag{1}$$

and the above differential equation can be solved in *closed form* in terms of the parameter p as

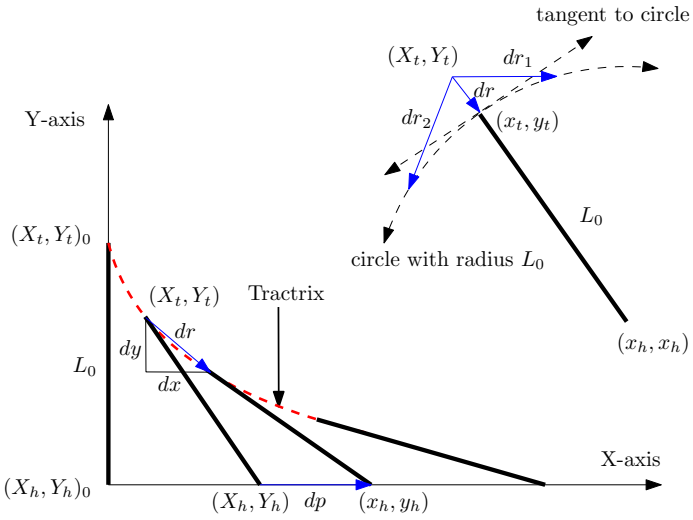
$$[x_t(p), y_t(p)]^T = [p - L_0 \tanh \frac{p}{L_0}, L_0 \operatorname{sech} \frac{p}{L_0}]^T \tag{2}$$

The extension of tractrix equation for a motion of the 'head' in arbitrary direction in 3D as well as an algorithm to compute the location of the 'tail' in 3D can be found in [25]. In case of multiple links connected to each other, as in the case of a hyper-redundant robot or a one-dimensional flexible object approximated as a series of connected links, the algorithm can be applied iteratively from the first link through the last link (see [25]). In this case, only the path traced by the head of the *first* link ($\mathbf{x}_h^1$) is specified. The position of the tail of the *first* link ($\mathbf{x}_t^1$) is calculated using the tractrix approach. Since the tail of the first link is also the head of the second link (as seen in Fig. 1b), the displacement of the head of the second link will be already obtained from solution of the first ($\mathbf{x}_t^1 = \mathbf{x}_h^2$). Using this data, tractrix approach is then used to find the position of tail of the second link ($\mathbf{x}_t^2$) and the procedure is iterated for all the links to obtain the configuration of the entire robot.

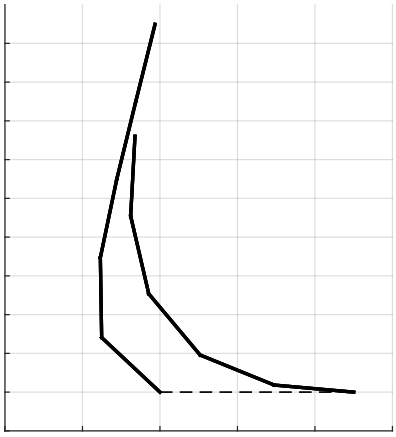
Some of the key properties of the tractrix based approach are as follows:

- It can be shown that $dr \leq dp$. As a consequence when the tractrix based algorithm is applied sequentially from the first link to the last link, the displacement attenuates from the head of the link-chain to end of the chain as illustrated in figure 1b. This imparts a more 'natural looking' and realistic motion to the hyper-redundant manipulator.

- It is shown in reference [25] that the complexity of the tractrix based motion planning algorithm is $\mathcal{O}(n)$, i.e., linear in the number of links n in the hyper-redundant manipulator. Hence the tractrix based approach is computationally efficient.



(a) Tractrix curve in 2D with one link



(b) Tractrix with multiple links

Figure 1: Tractrix in a plane

In reference [26], it is shown that for a single rigid link, the tractrix curve can also be obtained by minimizing an L^2 metric related to the instantaneous velocity subject to a constraint of preserving the length of link[1]. For any rigid link of the hyper-redundant robot, the co-ordinates of the tail of a rigid straight link, due to an infinitesimally small motion of the head, can be obtained from the following minimization problem

$$\arg\min_{\mathbf{x}_t} \|\mathbf{x}_t - \mathbf{X}_t\| \tag{3}$$

$$\text{Subject to: } \|\mathbf{x}_h - \mathbf{x}_t\| - L_0 = 0 \tag{4}$$

$$\text{Given data: } \mathbf{x}_h, \mathbf{X}_t, L_0$$

An advantage of expressing the tractrix solution as a minimization problem is that we can add more constraints in addition to the preservation of the length. For obstacle avoidance,

[1]Using calculus of variation, it is shown for a generic 2D curve the L^2 metric is minimum when the velocity at point is along the tangent at that point – see equation(15) in [26]. The L^2 metric defined in the work is similar to the Euclidean distance traveled by points on the curve. When the 2D curve is assumed to be a straight line (rigid link), the equations of the tractrix curve are derived – see equations (16) through (23).

we can formulate the optimization problem as:

$$\begin{aligned} &\arg\min_{\mathbf{x}_t} \|\mathbf{x}_t - \mathbf{X}_t\| \\ \text{Subject to:} &\|\mathbf{x}_h - \mathbf{x}_t\| - L_0 = 0 \\ &\mathbf{f}(\mathbf{x}_t) \succeq 0 \\ \text{Given data: } &\mathbf{x}_h, \mathbf{X}_t, L_0 \end{aligned} \tag{5}$$

where $\mathbf{f}(x) = 0$ are the analytical equations of the boundaries of the surfaces each of which are to be avoided[2]. For example, if the tail is to avoid a single obstacle represented by a circle with center (x_c, y_c), the expression $\mathbf{f}(\mathbf{x}) = \mathbf{f}(x, y) = (x - x_c)^2 + (y - y_c)^2 - r^2 > 0$ ensures that the point $\mathbf{x}$ always lies outside the circle of radius r. Complex objects can be modeled as a combination of super-ellipses as shown in [27]. In this case, $\mathbf{f}(\mathbf{x})$ will be a vector of all boundary equations $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), ...f_m(\mathbf{x})]^T$. It is also worth noting that the value of constraint function in equation (5) will increase or decrease as the point is farther from the curve $\mathbf{f}(\mathbf{x})$; the value being zero on the curve. It maybe noted that the resulting solution from the optimization with obstacle avoidance may not necessarily be the tractix curve, the motion of the tail will nevertheless appear realistic.

Extending the idea of motion planning as a minimization problem, the motion of the hyper-redundant robot through a duct can be formulated as:

$$\begin{aligned} &\arg\min_{\mathbf{x}_t} \|\mathbf{x}_t - \mathbf{X}_t\| \\ \text{Subject to: } &\|\mathbf{x}_h - \mathbf{x}_t\| - L_0 = 0 \\ C_{\text{ineq}} : \ &f(\mathbf{x}_t) < 0 \\ \text{Given data: } &\mathbf{x}_h, \mathbf{X}_t, L_0 \end{aligned} \tag{6}$$

Here, $\mathbf{x}_h^1$ will trace a series of points inside the duct. While this expression is applicable for a duct represented by a single surface with the boundary $f(x)$, unlike the obstacle avoidance problem, the same will not work in the case of complex surfaces represented by combination of simpler analytical shapes. This is because if a point is classified as inside one of the simpler shapes, then it should be classified as outside the other shapes forming the duct. In other words, if one constraint function $f_k(x) < 0$, then the other constraint functions $f_{i \neq k} \geq 0$. In the next sections, we present different methods to represent the ducts and

[2]The curled inequality symbol $\succeq$ (and its strict form $\succ$) is used to denote generalized inequality: between vectors, it represents component wise inequality; between symmetric matrices, it represents matrix inequality.

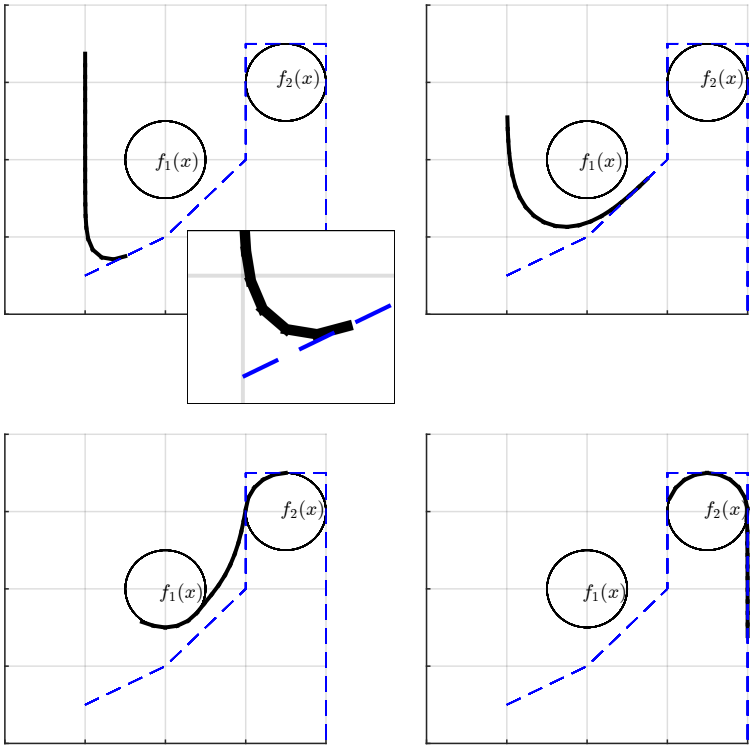


Figure 2: Obstacle avoidance in a plane (Inset shows the individual links of robot)

how confined space motion is achieved in different cases. While it may be observed in the following sections that $\mathbf{x}_h^1$ could be the points on any curve inside the duct, for simplicity, we choose the path of the head as the medial axis of the duct in the further derivations. When the bounding curves are line segments or curves, the medial axis is calculated by taking the mean of points on both boundaries of the duct. For boundaries represented by analytical curves or surfaces (for example ellipses and ellipsoids), the path traced by head is usually generated by combining the axes of these shapes.

2.1 Nature of the optimization problem

In the previous section we discussed how the tractrix based motion planning with obstacles can be formulated as a constrained optimization problem. In this section, we show that the problem in equation (6) and the variations of the same used in the current work can be posed as convex problems and hence it is possible to obtain unique solutions.

A function $f : \mathbb{R}^n \to \mathbb{R}$ is *convex* if and only if the following two conditions hold:

I The domain of f, $\mathbf{dom}(f)$ is a convex set, and,

II For all $x, y \in \mathbf{dom}(f)$ and θ with $0 \leq \theta \leq 1$

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y) \tag{7}$$

In our problem, the function is the L_2 norm of a vector in $\mathbb{R}^2$ or $\mathbb{R}^3$. Though the L_2 norm is defined for all real vectors, the constraints ensuring that the object moves within the confinement of the duct restricts the domain to a feasible closed set $\mathcal{S}$[3]. Therefore, if the conditions I and II associated with the definition of a convex function are satisfied for $\mathcal{S}$ and the L_2 norm, then the solution obtained for equation (3) is unique.
It is a well known result that any p-norm $||x||_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$, $x \in \mathbb{R}$ is a convex function for $p \geq 1$ [28] and all we need to ensure that the feasible set $\mathcal{S}$ is convex. It maybe noted that by themselves none of the problems posed in equations (3), (5) and (6) are convex because of the non-linear equality constraints associated with them. To pose it as a convex problem we approximate the non-linear constraint with linear constraints as described in section 5.2. For the cases of motion planning in 2D and 3D, the paths chosen are either entirely convex or are discretized as such. As discussed in the following sections, we do not represent a duct as a union of closed convex sets as the first condition of convexity cannot be guaranteed for sets formed by union of convex sets. For all our numerical examples, we choose $\mathcal{S}$ to be convex by assigning convex boundaries to it or, by expressing $\mathcal{S}$ as intersections of closed convex sets.

3 Representation of ducts in 2D and 3D

In this section, we present different methods to represent a duct in 2D and 3D and how we can add constraints in different representations of duct so as to ensure that the the tip will always lie inside the duct during motion. Each method is shown to have its own advantages.

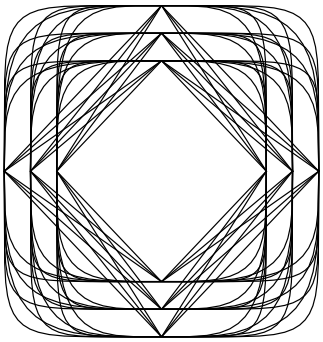
3.1 Duct using super-ellipses

One method to represent a duct in 2D is by overlapping a series of super-ellipses as shown in figure 3b. This is the most straightforward means of representation as shown in [27]. In

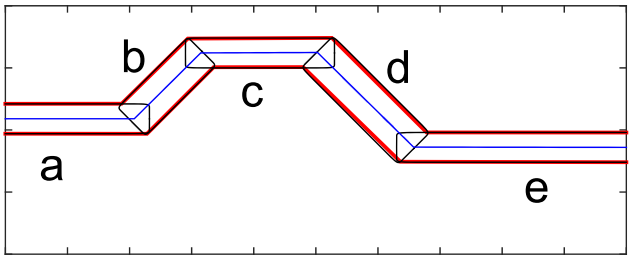
[3]Assuming $\mathcal{S}$ to be a closed set allows the object to physically touch the boundaries of the duct.

Cartesian co-ordinate system $\mathbb{R}^2$, the contour of super-ellipse can be obtained from:

$$f(\mathbf{x}) = f(x, y): \quad \left|\frac{x - x_c}{a}\right|^n + \left|\frac{y - y_c}{b}\right|^n - 1 = 0 \tag{8}$$



(a) Super ellipses



(b) Duct modeled as a combination of super-ellipses – red: outer boundary, blue: desired path of head

Figure 3: Super-ellipses and a duct modeled as combination of 5 super-ellipses

The condition $f(\mathbf{x}_t) < 0$ will ensure that the co-ordinates of the tail of the link (x_t, y_t) always lie inside the bounding curve of a super-ellipse. However, in case of multiple equations $(f_i(x),\ i = 1, 2, ..., m)$, only one of them will be satisfied for the point to be inside the duct. In practical implementation, this translates to saying that the least value amongst all the values of $f_i(x)$ should be less than zero. For the i^{th} super-ellipse which is rotated by an angle ϕ_i about the $Z-$ axis and whose center is translated to the co-ordinates (x_i, y_i) so as to fit a portion of a duct, the co-ordinates of boundary should be multiplied with a transformation matrix

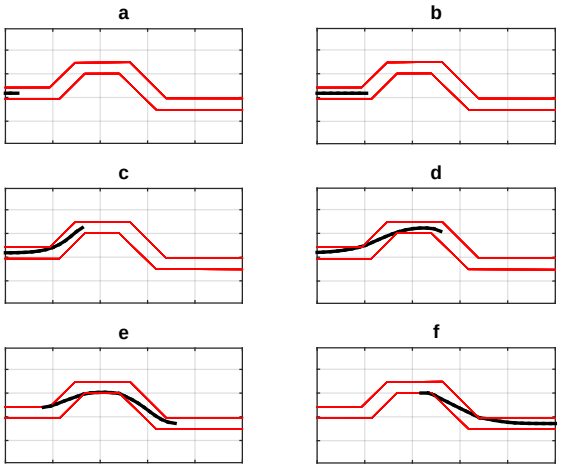
$$T_i = \begin{bmatrix} \cos\phi_i & -\sin\phi_i & 0 & x_i \\ \sin\phi_i & \cos\phi_i & 0 & y_i \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

The constraint equation now becomes $g_i(\mathbf{x}_t) : f_i(T_i^T \mathbf{x}_t) < 0,\ i = 1, 2, ..., m$ and for implementation, we can write the inequality constraint as

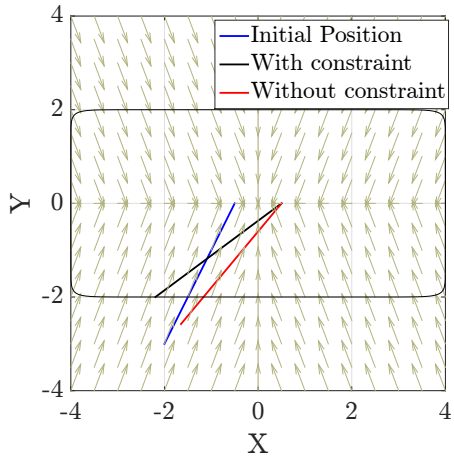
$$C_{\text{ineq}}: \quad \min\left(g_i(\mathbf{x}_t)\right) < 0, \quad i = 1, 2, ..., m \tag{10}$$

An example of single link and multi-segmented chain passing through the duct is shown in figure 4a. The motion of a unit link with and without constraint is shown in figure 4b. The negative gradient of the inequality constraint function is also shown in the figure 4b. The

method shown here is quite fast and scalable as explained in section 5.2 and the majority of time taken is in identifying the super-ellipsoids which fit the duct profile. For the example shown in this section, this identification is done by manually selecting clusters of points in the duct and fitting ellipses which will reduce the fitting error in a least squared sense.



(a) Motion of 20 link robot through duct modeled as combination of super-ellipses (See Movie 1)



(b) Effect of gradient of inequality constraint in pulling the tail into the duct

Figure 4: Tractrix based algorithm on duct represented by super-ellipses

3.2 Duct as set of connected quadrilaterals

Since the profile of a super-ellipse is always symmetric, for complex and non-symmetric ducts, representation using the previous method might require a large number of shapes. For such cases, a complex duct shape can be represented as a closed shape obtained by stitching convex quadrilaterals as shown in figure 5. The individual quadrilateral patches, denoted as $A_1, A_2, ..., A_n$, are each bounded by the line segments defined by the points $(\mathbf{P_0}, \mathbf{P_1}), (\mathbf{P_1}, \mathbf{P_2}), ..., (\mathbf{P_{n-1}}, \mathbf{P_n})$ for the curve $\zeta_\mathbf{1}$ and $(\mathbf{Q_0}, \mathbf{Q_1}), (\mathbf{Q_1}, \mathbf{Q_2}), ..., (\mathbf{Q_{n-1}}, \mathbf{Q_n})$ for the curve $\zeta_\mathbf{2}$. Classification of a point $\mathbf{x}_t$ as inside or outside a quadrilateral represented by points, say, $\mathbf{P}_1, \mathbf{P}_2, \mathbf{Q}_2$ and $\mathbf{Q}_1$, is essentially checking the placement of the point in the half spaces represented by the four lines spanned by the point set $(\mathbf{P}_1, \mathbf{P}_2)$, $(\mathbf{P}_2, \mathbf{Q}_2)$, $(\mathbf{Q}_2, \mathbf{Q}_1)$, and $(\mathbf{Q}_1, \mathbf{P}_1)$. This can be written as a set of four inequality constraints:

$$A_i^1 x_t + A_i^2 y_t + B_i < 0 \quad i = 1, 2, 3, 4 \tag{11}$$

where $A_i^1 x + A_i^2 y + B_i = 0$ represents a line obtained from one pair of non-diagonal points in the quadrilateral. In matrix form, the inequality will be:

$$C_{\text{ineq}}: \quad [\mathbf{A}]\,\mathbf{x}_t + \mathbf{B} < 0 \tag{12}$$

where $[\mathbf{A}]$ is a 4×4 matrix and $\mathbf{B}$ is a 4×1 vector.

A more convenient method for practical applications is as follows:

The co-ordinates of a point inside the surface patch A_i is given by the parametric expression

$$\mathbf{x}_i(u, v) = [\mathbf{P}_{i-1} + (1-u)\,\mathbf{P}_i]\,(1-v) + [\mathbf{Q}_{i-1} + (1-u)\,\mathbf{Q}_i]\,(v) \tag{13}$$

in parameters u and v. If the vertices of the quadrilateral are given by $\mathbf{P}_i = [{}^xP_i, {}^yP_i]^T$ and $\mathbf{Q}_i = [{}^xQ_i, {}^yQ_i]^T$, then the analytical expressions for the terms u and v, given the value of $\mathbf{x}_i$, can be obtained by solving equation (13) (see appendix A). The values of u, v can be used to classify the point with respect to the surface patch A_i[4].

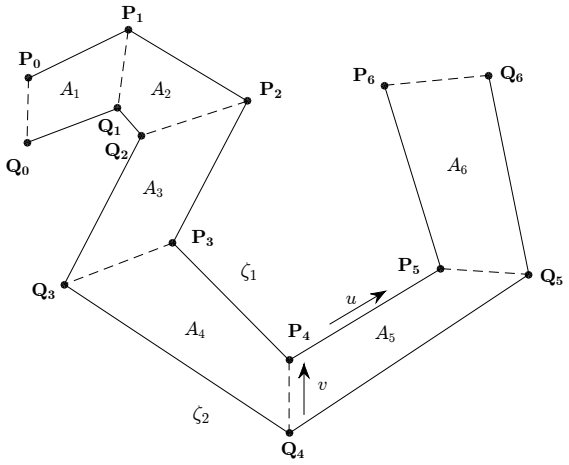


Figure 5: Duct represented by stitched quadrilaterals

In case of a single quadrilateral patch, the inequality constraints will be simply,

$$C_{\text{ineq}}: \; 0 \le u \le 1, \quad 0 < v < 1 \tag{14}$$

[4]It may be noted that there will be two sets of solution and they are not always real and unique. For example, the point $\mathbf{P} = (10, -5)$ when classified with respect to the area A given by the points $\mathbf{P}_1 = (0, 15)\,, \mathbf{P}_2 = (10, 10)\,, \mathbf{Q}_1 = (0, 0)$ and $\mathbf{Q}_2 = (4, 1)\,$, returns the values $u = (1.0 + 0.6\ i,\ 1.0 - 0.6\ i)$ and $v = (2.0 + 1.9\ i,\ 2.0 - 1.9\ i)$. However, it is easy to filter out the imaginary set of solutions, should the algorithm encounter the same.

for real values of u and v. In case of multiple patches, classifying one point with respect to all the patches return the values $(u_1, v_1), (u_2, v_2), ..., (u_m, v_m)$ etc. for the m number of patches $A_1, A_2, ..., A_m$ and consequently, m set of conditions. However, out of the m condition sets, only one set should be satisfied since the point will belong to only one patch at a given instance of motion through the duct.

In order to provide a gradient to the constraint which will direct the joints of the robot into the duct, an inequality constraint is to be included as in the case of super-ellipses described in the previous subsection. If $\hat{u}$ and $\hat{v}$ represent the parameters obtained for a point $\mathbf{x}_t$ classified with respect to the quadrilateral A_i, $x_{\zeta_1}(t) = \mathbf{P}_{i-1} + (1 - \hat{u})\,\mathbf{P}_i$ and $x_{\zeta_2}(t) = \mathbf{Q}_{i-1} + (1 - \hat{u})\,\mathbf{Q}_i$ will give the two points on the duct boundary curves corresponding to the parameter $\hat{u}$. Then we can see that the quantity

$$h = \|\mathbf{x}_{\zeta_1} - \mathbf{x}_t\|^2 + \|\mathbf{x}_{\zeta_2} - \mathbf{x}_t\|^2 - \|\mathbf{x}_{\zeta_1} - \mathbf{x}_{\zeta_2}\|^2 \tag{15}$$

will always assign a negative real value for h when point is inside the duct and a positive real value when the point is outside the duct. The value will be zero only at the boundaries. Hence, for an array of quadrilaterals, it is only necessary that the minimum value of the vector $\mathbf{h} = [h_1, h_2, ..., h_m]$ should be negative for classifying the point with respect to the duct, as in the case of previous section. It may be noted that the inequality only takes into account the parameter variation across the boundaries (along the parameter v) and not in the direction of u. To account for the same, we make use of the function χ which is necessarily a linear combination of two Heaviside step functions $H(0) - H(1)$, defined as:

$$\chi(t) = \begin{cases} 0, & t < 0 \\ 1, & 0 \leq t \leq 1 \\ 0, & t > 1 \end{cases} \tag{16}$$

The function χ applied on the quantity $\hat{u}_i$ (which is the value of parameter u classified with respect to quadrilateral A_i), will return 0 only if the point satisfies the constraint $0 \leq \hat{u}_i \leq 1$. Now, multiplying this quantity $\chi(\hat{u})$ with h_i will return a non-zero negative value only if the point is inside the duct. Then the following inequality constraint:

$$C_{\text{ineq}} : \quad [\chi(\hat{\mathbf{u}})]^T \mathbf{h} < 0 \tag{17}$$

where $\hat{\mathbf{u}} = [\hat{u}_1, \hat{u}_2, ..., \hat{u}_m]^T$ becomes a more practical and convenient way to implement the inequality constraint in the optimization problem.

As an example, motion of a unit link passing through the duct and the effect of the added inequality constraint in equation (17) to pull the tail end of the link which is initially positioned outside the duct, is shown in figure 7. Apart from being more flexible, another advantage of representing a 2D duct as a set of connected quadrilaterals in this parametric form is that by setting the limits of the parameter v to $0 + \delta < v < 1 - \delta, \quad \delta < 0.5$, it is easy to manually add a clearance from the walls of the duct without manipulating the duct itself. Also, it is easy to note that $\delta = 0.5 - \epsilon$ (where ϵ is a small number) would follow the backbone curve motion.

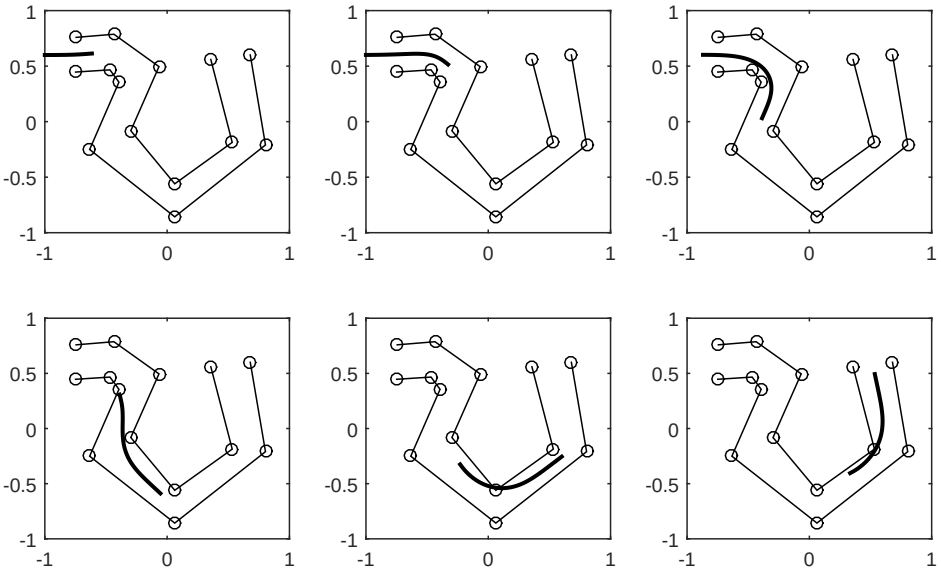


Figure 6: Example of constrained motion of a 40 link robot with stitched quadrilaterals (See Movie 2)

3.3 Duct as two non-intersecting continuous curves

If the non-intersecting border curves of the duct can be analytically expressed, then the equation of the surface patch can be written as

$$\mathbf{x}_i(u, v) = \zeta_1(u)\,(1 - v) + \zeta_2(u)\,(v) \tag{18}$$

For example, figure 8a shows a 2D duct defined by two curves $\zeta_1(u) = [u,\ \sin(u)]^T$ and $\zeta_2(u) = \left[u,\ \sin\left(u + \frac{\pi}{8}\right) + 1\right]^T$ and a path chosen midway between the two curves. The

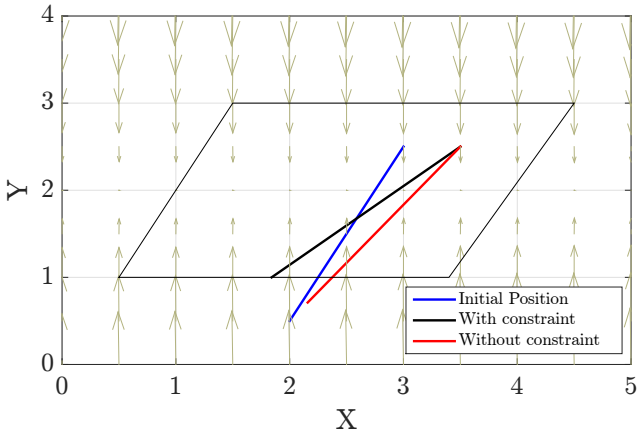


Figure 7: Effect of gradient of inequality constraint in pulling the tail into the quadrilateral duct

equation of the surface generated by this curves will be

$$\begin{bmatrix} x(u,v) \\ y(u,v) \end{bmatrix} = \begin{bmatrix} u \\ \sin(u) + \left[\sin\left(u + \frac{\pi}{8}\right) - \sin(u) + 1\right] v \end{bmatrix} \tag{19}$$

which has the analytical solution for u and v given by

$$\begin{aligned} u &= x \\ v &= \frac{y - \sin(x)}{\sin\left(x + \frac{\pi}{8}\right) - \sin(x) + 1} \end{aligned}$$

In this case, we solve the optimization problem:

$$\min_{\mathbf{x}_t} \|\mathbf{x}_t - \mathbf{X}_t\| \tag{20}$$

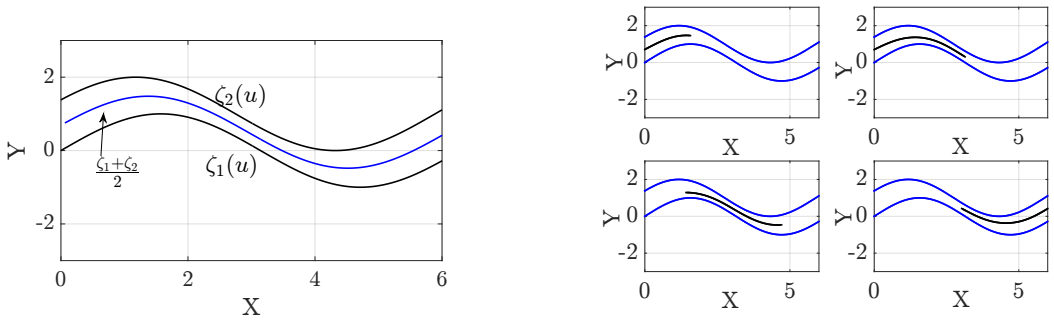
$$\text{subject to:} \quad \|\mathbf{x}_h - \mathbf{x}_t\| - L_0 = 0$$

$$0 < v|_{\mathbf{x}_t} < 1 \tag{21}$$

$$\text{Given data: } \mathbf{x}_h, \mathbf{X}_t, L_0$$

An example movement of a 40 link hyper-redundant manipulator through the duct is shown in figure 8b. However, analytical solution is not always viable for complex equations and numerical procedure must be employed to find the values of u and v corresponding to the given tail point to be classified. Additionally, since multiple solutions may be possible for such cases, the correctness of the solution would heavily depend on the choice of initial guess[5].

[5]The same argument will also hold for analytical surfaces spanned in 3D and hence is not studied further.



(a) Example of analytical duct

(b) Motion through analytical duct (See Movie 3)

Figure 8: Tractrix based algorithm on analytical duct

3.4 Duct in 3D using combination of super-ellipsoids

Representation of ducts in 2D can be extended to 3D by using super-ellipsoids. In a Cartesian co-ordinate system, the surface of a super-ellipsoid follows the equation:

$$f(x,y,z): \left[\left\{\left(\frac{x}{a}\right)^{\frac{2}{e}} + \left(\frac{y}{b}\right)^{\frac{2}{e}}\right\}^{\frac{e}{n}} + \left(\frac{z}{c}\right)^{\frac{2}{n}}\right]^{\frac{n}{2}} - 1 = 0 \tag{22}$$

By changing the parameters a, b, c and n, we get different closed surfaces as shown in figure 9 and by combining different super-ellipsoid shapes, we can generate a 3D duct profile.

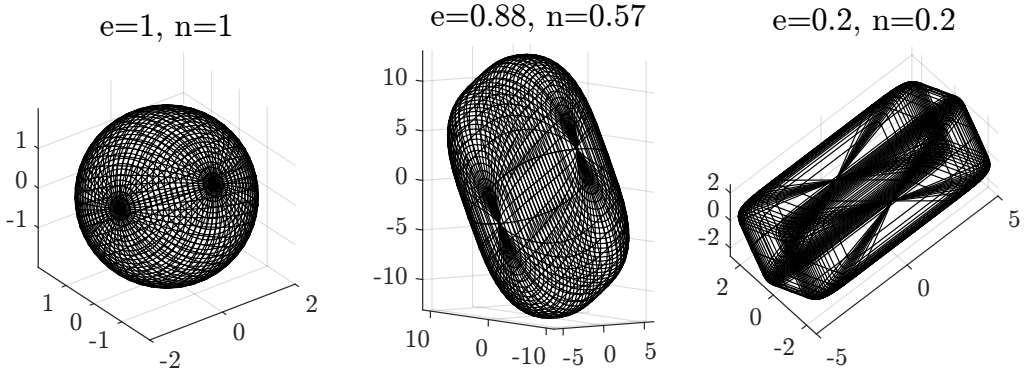


Figure 9: Super-ellipsoids

The procedure to calculate the inside-outside condition is same as that of the method described in section 3.1. The inequality condition will be equation (10). An example problem with duct approximated using super-ellipsoids is shown in figure 3b. As mentioned in the

case for super-ellipses, solution to the motion planning problem with ducts represented by super-ellipsoids is fast (also explained in section 4). Identifying the shapes which fit the duct, is also same as the method mentioned in section 3.1.

3.5 Duct as a set of connected cylinders

Similar to the set of connected quadrilaterals in 2D, a duct in 3D can be represented by series of connected cylinders. By linearly interpolating two circles in space, we get the parametric equation of the cylinder as given in equation (23) below (see figure 18a and appendix B for the expanded form of equation (23))

$$x = C_1(u, v, \theta), \quad y = C_2(u, v, \theta), \quad z = C_3(u, v, \theta) \tag{23}$$

where the parameters u, v and θ varies along the radial, axial and circumferential direction of the cylinder respectively (see figure 18a). Similar to the representation in section 3.2, $0 \leq u < 1$ and $0 < v < 1$ classifies the point as inside the cylinder. The constrained inequality 17 generated will also be valid for cylinders. The quantity h_i is given as $h_i = \hat{u}_i - 1$ which shows the same characteristics as defined by the value of h_i in equation (15). The constraint inequality, hence takes the form:

$$C_{\text{ineq}} : \quad [\chi(\hat{\mathbf{v}})]^T \mathbf{h} < 0 \tag{24}$$

As is the case of quadrilaterals, it is possible to add a clearance from the walls by changing the radius of cylinder from r to $r - \delta$, which is a very desirable characteristic for robots used in medical applications.

3.6 Duct as point clouds

The most direct way of representing the duct would be as a point cloud as obtained from measurements or a depth map. Subsequently, it would be possible to process the raw data (by using alpha shapes [29] and standard Delaunay triangulation algorithms [30]) to obtain the geometric representation of the cloud of points as a convex polyhedron. Stereo-lithographic formatted file (STL) is a standard data structure, which has been used in the current work. Using the current framework, it is possible to pose the motion planning problem in the

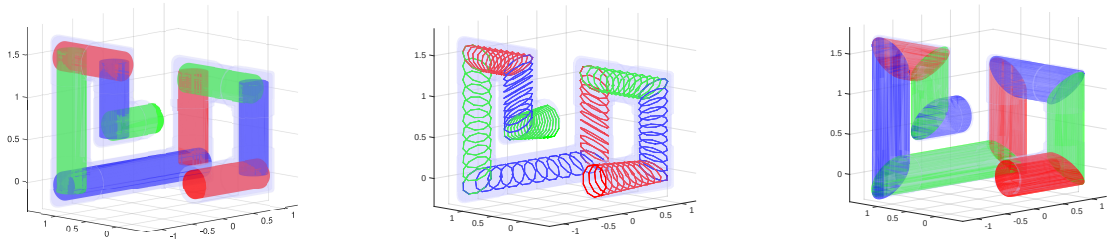
following form:

$$\begin{aligned} \min_{\mathbf{x}_t} & \|\mathbf{x}_t - \mathbf{X}_t\| \\ \text{subject to:} \quad & \|\mathbf{x}_h - \mathbf{x}_t\| - L_0 = 0 \\ & [\mathbf{A}]\mathbf{x}_t + \mathbf{B} \leq 0 \\ \text{Given data:} \quad & \mathbf{x}_h, \mathbf{X}_t, L_0 \end{aligned} \tag{25}$$

where $\mathbf{A}$ is a $m \times 3$ matrix and $\mathbf{B}$ is a $m \times 1$ vector. The left-hand side of m inequalities represent the equations of m number of planes spanned by three points on each triangular facet of convex polyhedron. The i^{th} equation, $A_i^1 x_t + A_i^2 y_t + A_i^3 z_t + B_i$ takes a value less than zero when the point $\mathbf{x}_t$ is in the half space which contains the origin and is greater than zero otherwise. The value also provides the attractive gradient which will ensure that the point stays inside the duct. However, in actual implementation, this procedure will be tedious and for practical convenience, it is possible to classify the point $\mathbf{x}_t$ as inside or outside the hull using the algorithm 2 described in appendix C. The attracting gradient which ensures the point to be inside the duct–as was the case with the previous methods– can be provided using the artificial potential field generated from the centroid of the point cloud in conjunction with the output of the in-out function. The inequality constraint then becomes

$$w(\mathbf{R}) \frac{1}{\|(\overline{\mathbf{R}}) - \mathbf{x}_t\|} \leq 0 \tag{26}$$

where $w(\overline{\mathbf{R}})$ represents the output from in-out function which is either 1 for the point being outside and 0 for the point being inside the cloud or on the bounding surface[6]. Representation of a pipe using ellipsoids, analytical cylinders and as convex point clouds is shown in figure 10.



(a) Fitted ellipsoids (b) Fitted cylinders (c) Convex polyhedra

Figure 10: Representation of duct using ellipsoids, cylinders and point clouds

[6]Unlike the previous classification problems, the bounding surface will also be considered as inside the surface in this case.

In the next section, we present three scenarios where we show the motion planning of a hyper-redundant robot in ducts and in presence of obstacles. The section also includes a discussion of the results in terms of complexity and largest length of a link that can traverse the confined spaces.

4 Examples of motion planning

In this section, we present three practical examples where the above mentioned methods are applied and some discussion on the implementation of the above mentioned algorithms. For all the examples, the algorithms have been implemented in Matlab [31] and the results are rendered using Blender [32].

4.1 Motion planning for inspection robots

One major use of hyper-redundant robots is in the inspection of ducts such as industrial pipelines. By representing the pipe-line using a set of ellipsoids (or cylinders), we use the aforementioned methods to plan the motion of a 20 link hyper-redundant robot through the duct. The path is chosen as the medial axis of the duct, which is also the axis of the cylinders which make up the duct. The configuration of hyper-redundant robot for each path-step is calculated and the simulation of resultant motion is shown in figure 11.

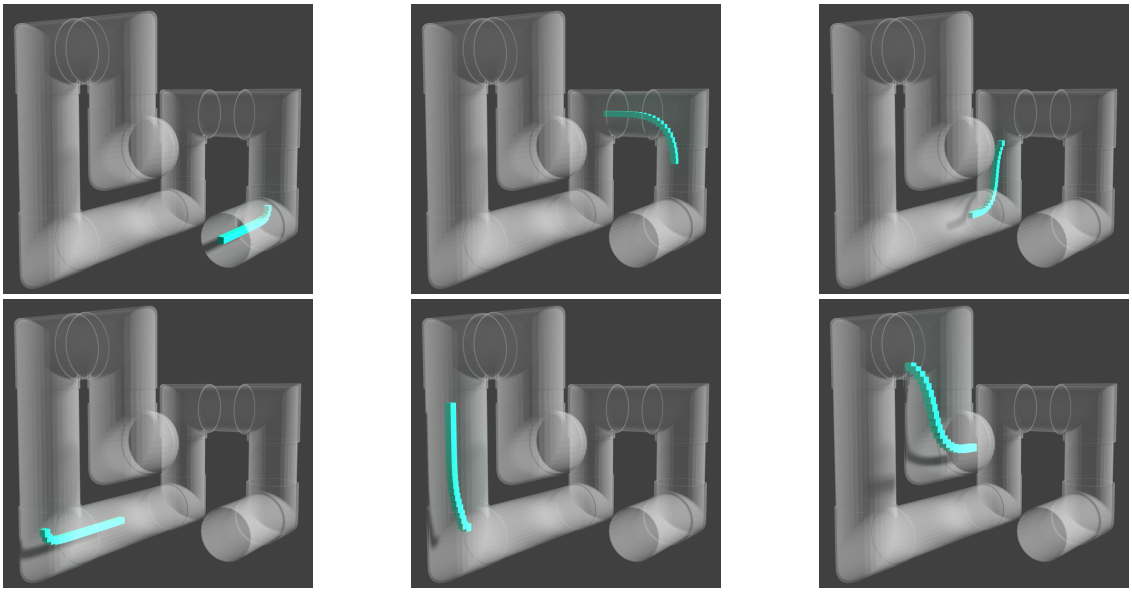
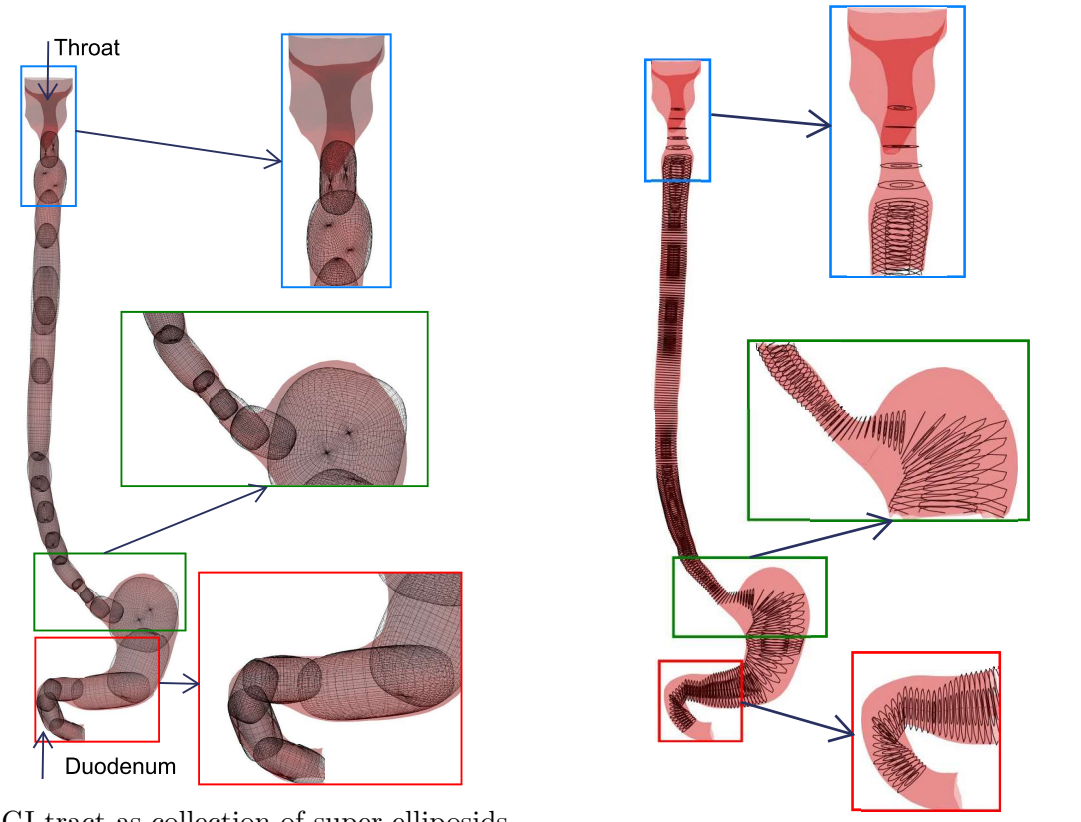


Figure 11: Motion of hyper-redundant inspection robot(See Movie 4)

4.2 Motion planning through a GI tract

There has been a growing interest in simulating motion of endoscope through GI tract for developing simulators for endoscopy and for implementing path and motion plans for endoscopic and laparoscopic surgical robots. In this section, we simulate the natural motion of an endoscope through GI tract. For simulation, we use the stereo-lithographic data of GI tract obtained by processing CT scan data obtained from Visible Human Dataset [33]. For demonstration, both the methods presented in section 3.4 and section 3.5 are used for approximating the GI tract. In the first method, a collection of points are manually selected from the STL file where super-ellipsoids are fit based on least square error minimization techniques. It may be noted that as this domain is inherently non convex, we cannot use the formulation in section 3.6, as approximately classifying a whether a point is inside or outside a domain will incur significant computations. Representation of GI tract as series of super-ellipsoids is shown in figure 12a.



(a) GI tract as collection of super-elliposids

(b) GI tract with connected cylinders

Figure 12: Representation of GI tract in two methods

For representing GI tract as cylinders, we first found out the medial axis of the duct following Cao et al.[34]. Then at equal intervals of distance along the medial axis, planes are drawn normal to the same. The collection of points which are in the close proximity of the plane are selected and a circle is fitted on the points using least square error minimization. The parameters so obtained are used for the cylinder equations in (23). Representation of GI tract as a series of connected cylinders is shown in figure 12b. It maybe noted that the GI tract is a 3D structure with both in-plane and out-of plane curvatures.

The realistic motion simulation of endoscope through GI tract is shown in figure 13. It maybe noted that the insertion of the endoscope is in a roughly horizontal direction and after passing through the throat, the head follows the centerline of the GI tract. In the simulation, the endoscope is discretized with links of dimension 0.5 cm and the motion of each link is computed such that it does not collide with the walls of the GI tract.

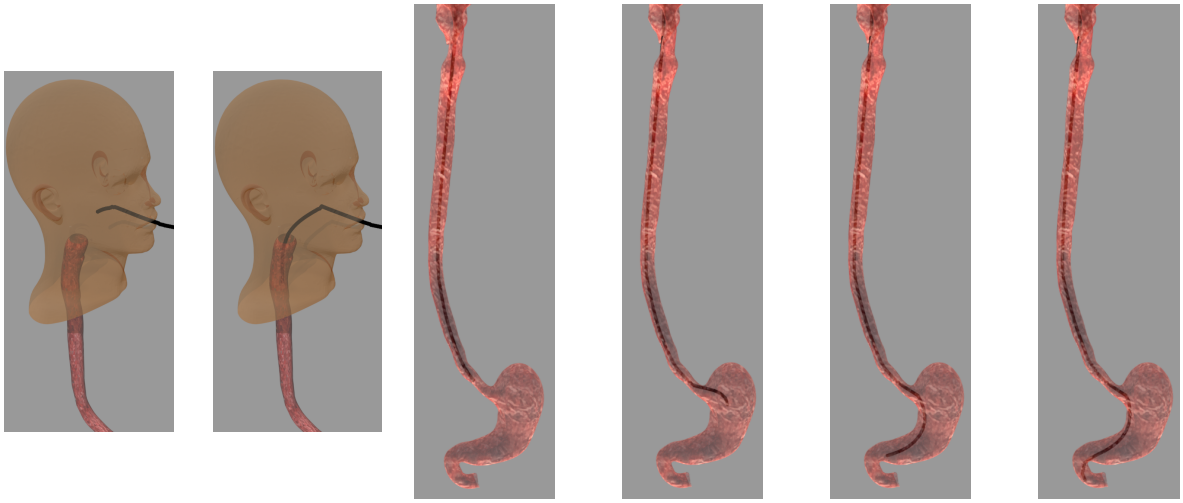


Figure 13: Motion of endoscope (shown in black) through GI tract (See Movie 5)

4.3 Motion planning for search and rescue operations

Exploring a cluttered environment – such as in an earthquake hit zone – is one of the major applications that necessitates the use of hyper-redundant manipulators. In this example, we show how the proposed algorithms can be effectively used in such applications. With reference to figure 14, the objective of the hyper-redundant manipulator, consisting of 20 links is to 1) Enter the scene through the hole in the outer wall, 2) Pass through the vent to get inside the room, 3) Explore the objects 3, 4 and 5 located inside the blue region and 4)

Exit without colliding with the complex shapes given by objects 6 and 7. In this example, motion planning is effectively tackled by combining the various methods discussed so far.

We use the following constraints to define the 6 different problems of the form in equation (6), to generate the entire motion plan. The first problem is to enter through the hole in the wall. For this, an ellipsoid of the size of the hole located in the outer wall, and the feasible set for the first problem is the interior of the ellipsoid, the path being the axis along the direction of the entry. The second problem is a free tractrix motion, where the robot has to reach the opening of the vent. The feasible set in this case is $\mathbb{R}^3$. Next, the robot passes through the vent. This problem is analogous to the one discussed in section 4.1. A collection of cylindrical polyhedra have been used to represent the vent– the feasible set in this problem. This is followed by another free tractrix problem, where the robot exits the vent at the top of the room and reaches to the floor level. Following which, the robot enters the transparent blue search region populated by objects 3, 4, and 5. The feasible set of each of the links are obtained on the fly, as a combination of the faces of the search region (transparent blue polyhedron) and the nearest obstacle. Finally, the robot exits the scene through another 'hole' in the designated search region modeled as a cylinder. This guarantees that the complex objects 6 and 7 have no effect on the robot motion as they have no contribution to the feasible set of the final problem. This would have been particularly difficult in half-space based motion plans [27]. Also, it is easy to see that the geometry based global motion planning techniques for obstacle avoidance existing in literature will entail a very involving problem formulation [35]. Figure 15 shows the simulated results for the motion of manipulator along a chosen path.

5 Optimization method

In this section, we present the details of the optimization method used and its implementation, discuss the computational complexity and some of the limitations of the approach.

5.1 Method

In all the examples provided, we formulated the tractrix problem as inequality-constrained convex optimization problems. The implementation is carried out in `MATLAB R2016b` using the standard `fmincon` function with the interior-point algorithm specified in the algorithm option. A main `MATLAB` script conducts the tractrix iteration which proceeds from the head of the robot to the tail for each step of the robot motion. All necessary pre-processing such

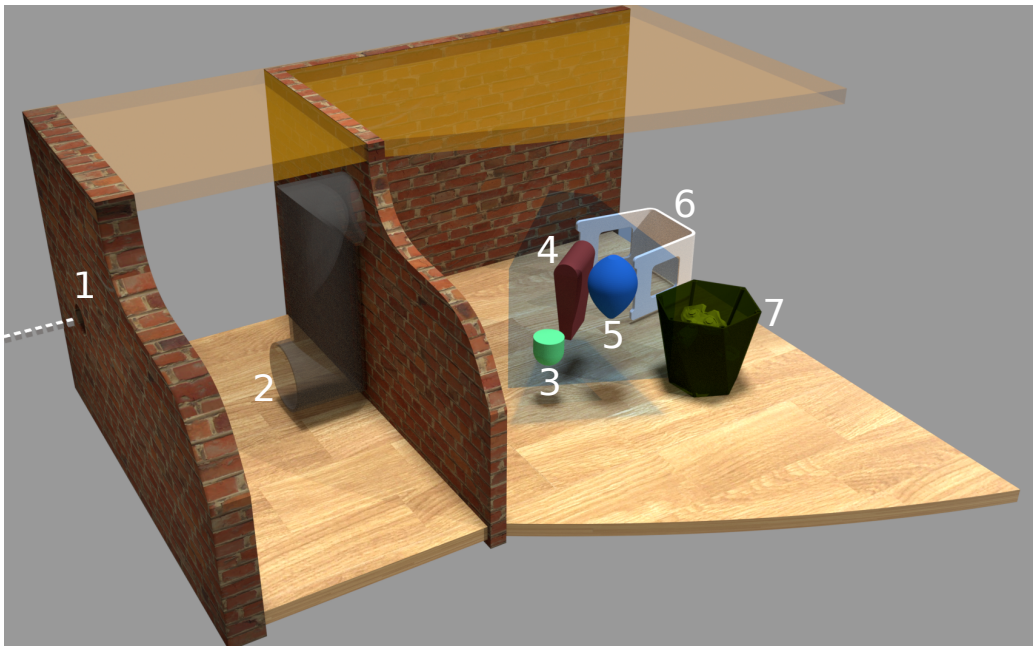


Figure 14: Schematic of search and rescue problem

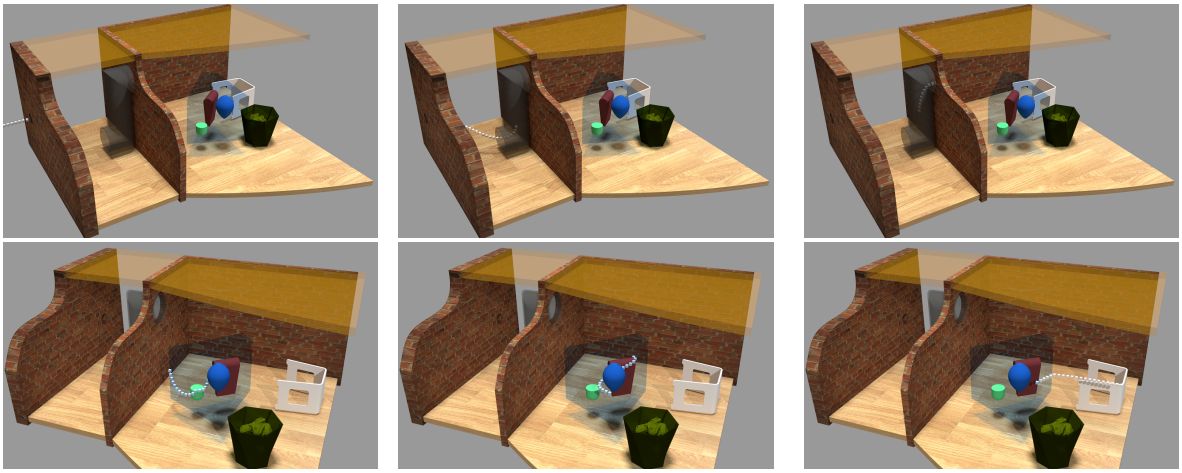


Figure 15: Motion of hyper-redundant robot in confined spaces (See Movie 6)

as the representation of the duct and medial axis determination is performed in this main script. The `fmincon` function has one of the arguments as the function to be minimized which is the function given in equation (3). The user specified data such as the position of the head of the links ($\mathbf{x}_h$), length of each links (L_0) and the position of the tail of the links ($\mathbf{X}_t$) are provided to `fmincon` as parameters. The equality constraint as well as the inequality constraints are given to `fmincon` as arguments. Finally, the information about the duct is also passed on to the `fmincon` function.

As an example, the algorithm which determine the motion of the robot passing through joined quadrilateral duct is given as follows:

Purpose : To determine the motion of the robot through a duct given by connected quadrilaterals
Input:

1. The set of points which define the duct boundaries $B_{\{1,2\}}^{\{1,2,..,l\}}$ where $\{1,2\}$ represent points on either sides of the duct made of $(l-1)$ quadrilaterals
2. An array containing the initial co-ordinates of the robot $X_0^{(1,2,...m)}$ where $2,3,..,(m-2)$ are the joints of the robot,
3. Link length of the robot L_0

Output: The array $X_{(1,2,...,N)}^{(1,2,...,m)}$ representing the entire motion of the robot

Calculate the mean of the points $B_1^{(\cdot)}$ and $B_2^{(\cdot)}$: $B_{1/2}^{(\cdot)}$ which are the vertices of the line segments forming the medial axis.
Join the points in $B_{1/2}^{(\cdot)}$ and obtain line segments $L_{1/2}^{1,2,...,(m-1)}$
Find points $P_{(1,2,...,N)}$ which are N equidistant points on the connected line segments.
for $i = 1, 2, ..., N$ **do**
    Assign $P^{i+1} \rightarrow$ `next_step`
    $X_i^1 =$ `next_step`
    for $k = 2, ..., m$ **do**
        Solve equation (3) for the tail position: '`tail_pos`' using the constraints (14), equations in Appendix A and also the parameters: `next_step`, X_{i-1}^{k-1}, X_{i-1}^{k}, L_0 and $B_{(1,2)}^{(\cdot)}$
        Assign `tail_pos` $\rightarrow X_i^k$
        Assign `next_step` $\rightarrow$ `tail_pos`
    end for
end for

Algorithm 1: Algorithm for motion planning of a robot through duct with connected quadrilaterals

We use the well-known and extensively used standard interior point algorithm (see for example [28]) to solve the optimization problems. One rationale behind using the interior point algorithm was the intuitive sense of a force field associated with the constraints, the notion of a convex domain– the feasible set where the problem is posed and solved. Additionally, the interior point method solved using the barrier method lends itself for rigorous complexity analysis and this is discussed in the following section.

5.2 Complexity analysis

An important aspect of the formulations described in the current work was to show that the formulation and implementation of motion planning problem using tractrix is intuitive and has a broad scope of application. In concurrence to that theme, in this section, we attempt to analyze the worst case computational complexity of such an implementation to show that the current work has potential for real time application in actual problems. We begin by reviewing some mathematical concepts, after Boyd and Vandenberghe [28], to be used in the following discussion.

Self Concordance: A convex function $f : \mathbb{R} \to \mathbb{R}$ is *self-concordant* if $|f'''(x)| \leq 2f''(x)^{3/2}$ for all $x \in \mathbf{dom}(f)$. It can be shown that the Euclidean norm and linear functions are self concordant.

Feasible set: For an optimization problem with constraints $f_i(\mathbf{x}) \leq 0$ and $h_j(\mathbf{x}) = 0$, for $i = 1, 2, ..., n$ and $j = 1, 2, ..., m$, the feasible set is $\mathcal{S} = \cap_{i=1}^{n} \mathbf{dom}(f_i \leq 0) \cap \cap_{j=1}^{m} \mathbf{dom}(h_j = 0)$. For all our problems, $\mathcal{S}$ is a subset of the real space $\mathbb{R}$ of dimension 2 or 3.

Slater's condition: Slater's conditions of constraint qualification hold when there exists an $\mathbf{x} \in \mathcal{S}$, a strictly feasible point, such that all the inequality conditions and equality conditions are satisfied[7]. This would, in general, mean that strong duality would hold and at the optimum point, the primal and dual problem would have the same solutions.

With the above definitions, we proceed to the complexity analysis of the optimization problems. In section 2.1 we discussed that the problem as presented in equation (3) is not convex due to the non-linear equality constraint guaranteeing the constant link length. We get around this problem by replacing the non-linear equality constraint by it's affine equivalent—a first order approximation of the constraint about a feasible point $\mathbf{x}^* \in \mathcal{S}$, and a trust region constraint as shown in equation (27). It is imperative for the trust region radius to be small— in our implementation, we have chosen the trust region radius ϵ_T to be about 6 orders of magnitude smaller than the smallest dimension of $\mathcal{S}$. In equation (27), $\nabla_{\mathbf{x}_t} g(\mathbf{x})$ is the gradient of the objective function.

$$\begin{aligned} \arg\min_{\mathbf{x}_t} \; & g(\mathbf{x}_t) : ||\mathbf{x}_t - \mathbf{X}_t|| \\ \text{Subject to} \; & : g(\mathbf{x}^*) - \nabla_{\mathbf{x}_t} g(\mathbf{x}^*)(\mathbf{x}_t - \mathbf{x}^*) = 0 \\ & |\mathbf{x}^* - \mathbf{x}_t| \preceq \epsilon_T \\ & C_{ineq} : f(\mathbf{x}) \preceq 0 \end{aligned} \tag{27}$$

[7]For a more complete treatment, refer to Chapter 5 of the book by Boyd and Vandenberghe [28].

We can rewrite equation (27) as a standard nonlinear program with the objective function g, all the m constraints $\mathbf{f}$ and $\mathbf{s}$, a vector of $m \times 1$ slack variables as

$$\begin{aligned} \arg\min_{\mathbf{x}_t} \; & g(\mathbf{x}_t) \\ \text{Subject to:} \; & \mathbf{f}(\mathbf{x}_t) + \mathbf{s} = 0 \\ & \mathbf{s} \succeq 0 \end{aligned} \tag{28}$$

We also observe that the problem in equation (27) corresponding to the implementations studied so far, is actually a minimum norm problem with either polyhedral (or polygonal in 2D) or super-quadric (or super-elliptic) constraints. As all of the constraints qualify Slater's condition, we conclude that strong duality holds for all versions of the problem studied in the current work. We choose an interior point method to solve the nonlinear programming problem at hand and by reformulating equation (28) as a barrier problem with barrier parameter μ. The KKT (Karush Kuhn-Tucker) necessary and sufficient conditions for the given problem is given in equation (29).

$$\begin{aligned} \nabla_{\mathbf{x}_t} g(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \mathbf{f}(\mathbf{x}_t)^T \vec{\lambda} &= 0 \\ \mathbf{f}(\mathbf{x}_t) + \mathbf{s} &= 0 \\ \mathbf{LS} - \mu \vec{e} &= 0 \end{aligned} \tag{29}$$

In equation (29), $\mathbf{L}$ is the $m \times m$ diagonal matrix of the Lagrange multipliers ($\vec{\lambda}$), $\mathbf{S}$ is the $m \times m$ diagonal matrix of the slack variables $\mathbf{s}$ and $\mathbf{e}$ is a $m \times 1$ vector of ones. The above system represents $2m + 1$ equations in $2m + 1$ variables and is known as the KKT system. While using an interior point method, the computation time depends on two principal factors—the number of Newton steps required to converge and the complexity of each Newton step, i.e., the total amount of computation required in solving equation (29) to obtain the Newton direction. For our case with a self-concordant log barrier and polyhedral constraints, which are by default self-concordant, we can conclude that the total number of Newton steps are of the order of $\sqrt{m} + c$ where c is an integer. In the case of our problems, we observed that for a 2D case the total number of iterations were about 10, for the six constraints($m = 6$)—4 linear constraints defining $\mathcal{S}$ and 2 trust region constraints.

Equation (29) can be linearized, and solved (for a 3D case) to obtain the Newton direction as

$$\begin{pmatrix} A_{3\times3} & B^T_{m\times3} & 0_{3\times m} \\ B_{3\times m} & 0_{m\times m} & I_{m\times m} \\ 0_{6\times m} & \mathbf{S} & \mathbf{L} \end{pmatrix} \begin{pmatrix} \triangle x \\ \triangle\lambda \\ \triangle s \end{pmatrix} = \begin{pmatrix} -\mathbf{F}_1 \\ -\mathbf{F}_2 \\ -\mathbf{F}_3 \end{pmatrix} \tag{30}$$

where, $\mathbf{A} = \nabla^2_{\mathbf{x}_t} g(\mathbf{x}_t) + \sum_{i=1}^{m} \nabla^2_{\mathbf{x}_t} \lambda_i f_i(\mathbf{x}_t)$ is a 3×3 symmetric matrix, $B = \nabla_{\mathbf{x}_t} \mathbf{f}(\mathbf{x}_t)$ is a $m \times 3$ matrix of the gradient of the m constraint functions, and $\mathbf{F}_1$, $\mathbf{F}_2$, and $\mathbf{F}_3$ are the left hand sides of equation (29). Following Wright and Nocedal [36] and Chakraborty et al. [37] we can obtain the Newton directions as:

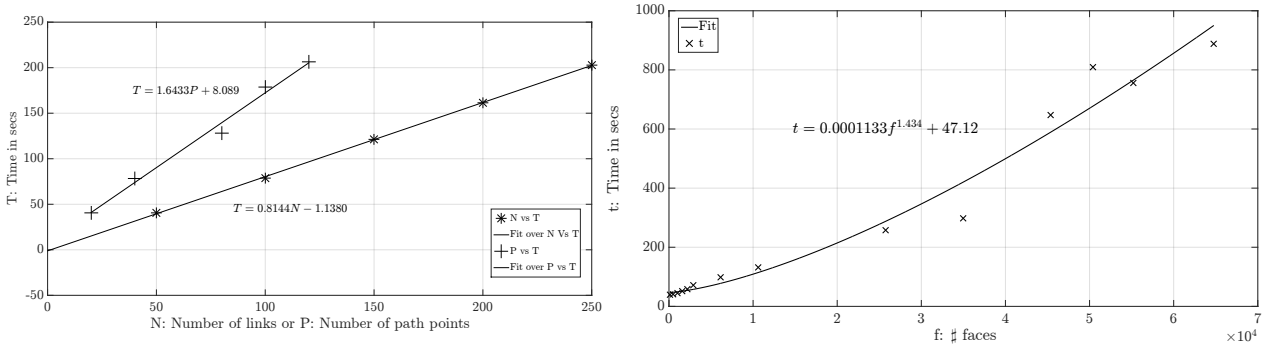
$$\begin{aligned} \triangle \mathbf{x}_t &= \mathbf{G}^{-1}(-\mathbf{F}_1 - \mathbf{B}^T(\mathbf{S}^{-1}\mathbf{L})(\mathbf{F}_2 + \mathbf{L}^{-1}\mathbf{F}_3)) \\ \triangle \lambda &= (\mathbf{S}^{-1}\mathbf{L})(\mathbf{B}\triangle \mathbf{x}_t - \mathbf{F}_2 + \mathbf{L}^{-1}\mathbf{F}_3) \\ \triangle s &= -\mathbf{L}^{-1}(\mathbf{F}_3 + \mathbf{S}\triangle\lambda) \end{aligned} \tag{31}$$

where, $\mathbf{G} = \mathbf{A} + \mathbf{B}^T(\mathbf{S}^{-1}\mathbf{L})\mathbf{B}$, a 3×3 matrix (for a 3D case) which can be inverted symbolically, also, $\mathbf{S}$ and $\mathbf{L}$ are diagonal matrices so their inverse can be calculated in linear time ($\mathcal{O}(m)$). Overall, the asymptotic complexity of solving equation (31) is $\mathcal{O}(m^3)$ as the calculation of G^{-1}, $\triangle x$ and $\triangle \lambda$ are $\mathcal{O}(m^3)$. This would result in a total asymptotic complexity of the implementation to be $\mathcal{O}(m^{3.5})$ – $\mathcal{O}(m^3)$ from Newton iterations and $\mathcal{O}(\sqrt{m})$ due to the self-concordant nature of the optimization problem. However, the computation time of a single iteration does not depend on the number of links of the hyper-redundant robot or the number of path steps, hence the proposed method is linear in these – it maybe recalled that the motion planning algorithm in *absence* of obstacles is also linear in the number of links. The linear nature is observed from figure 16a.

As discussed in section 3.6, in our implementation of the motion planning problem inside a polyhedral domain, we use a separate algorithm to determine whether to use the constraints of the form $[\mathbf{A}]_{m\times 3}\mathbf{x}_t + \mathbf{B}_{m\times 1} \leq 0$ in equation (25). Algorithm 2 discusses an $\mathcal{O}(m)$ scheme to do the same in case of a polyhedron with m faces. Therefore, the total asymptotic complexity of our implementation should be about $\mathcal{O}(m^{1.5})$. Figure 16b shows that actual complexity of the implementation of our algorithm scales as $(m^{1.43})$, with room for further improvement[8]. Furthermore, the formulations involving parametric shapes (super ellipses and super ellipsoids) are even faster because the classification of a point with respect to the domain is available as a closed form expression. It may also be noted that this analysis is not applicable to cylinders since they require iterative methods to solve the in-out classifier.

To summarize, the choice amongst the three methods to represent 3D ducts can be based on the computational time and required modelling effort. In the first method of representing ducts by a collection of ellipsoids, modelling time is significant as a separate utility is required to identify the points on surface and to fit the ellipsoids on the data points. However, the

[8]Computation times in figure 16 was obtained using the 'tic-toc' function in MatlabR2016b running on a workstation with dual 8 core Intel XEON (2.1GHz), 32 GB RAM and Windows 10.



(a) Our implementation is time-linear in number of path points and link numbers

(b) Our implementation scales as $(m^{1.43})$. Times are for a 20 link robot.

Figure 16: Complexity of our implementations. We plan the motion of the robot through a cylindrical duct of radius 7 and length 80 units and the path was $[3\theta,\ 0,\ 5\sin\theta],\ \theta = [0, 24\pi]$

Representation	Pre-processing	Model flexibility	Computational time	Scope
Ellipses/Ellipsoids	Slow	Low	Very fast	High
Quadrilaterals/Cylinders	Slow	High	Slow	Moderate
2-D analytical curves	Very fast	High	Fast	Very low
Point clouds	Very fast	Low	Fast	Low

Table 1: Comparison between different representations in 2D and 3D

solution method is quite fast due to the availability of a closed form function for the in-out classification. The second method of fitting cylinders also involve significant amount of pre-processing to identify cylinders that fit the duct. The implementation time is also high due to the non-analytical nature of the in-out classifier. However, the provision of having a constant clearance from the wall makes it particularly attractive for formulating motion plans for which the thickness of the robot has to be taken into account. Finally, the method of using convex polyhedra to represent ducts requires the least pre-processing time, and is easier to implement with the worst case complexity of $\mathcal{O}(m^{1.5})$.

Table 1 shows a comparison between different representations discussed in this paper and how they differ from each other in terms of 1) Pre-processing time, 2) Flexibility of model (such as updating and manipulation), 3) Computational time and 4) Scope of implementation

5.3 Limitations of the tractrix based scheme

In spite of the certain advantages regarding formulation and computational aspects of the tractrix based motion planning approach, there are a few limitations which are of importance. A main limitation is that the ducts represented by analytical curves and cylinders would require an in-out classifier to demarcate the feasible space of the posed optimization problem. Often, this would involve numerical formulations which are computationally expensive. Secondly, it can be seen that a given hyper-redundant robot with a particular link length will not be able to negotiate a path with a "very high" curvature. This is shown schematically in figure 17a. In figure 17a, the points 1,2,...,8 denote the coordinates of $\mathbf{x}_h$ across successive iterations. From iteration 6 onwards, the constraints demarcating the feasible space $\mathcal{S}$ and the one guaranteeing a constant link length cannot be simultaneously satisfied unless the tail $\mathbf{x}_t$ (the result of equation (3)), backtracks its own path. Soon after, the optimization problem stops as the link seems to be "locked" at the trough of $\zeta(u)$[9].

For an initial attempt at quantifying the locking effect, we borrow the concept of "traversability" from literature on wheeled mobile robots (see e.g. [38]). A curve $\zeta(u)$ is traversable by a circle C_i, $C_i : \begin{pmatrix} R_i\cos(v) \\ R_i\sin(v) \end{pmatrix}$, $0 \leq v \leq 2\pi$ if C_i can roll over the curve $\zeta(u)$ while maintaining "only one" point of contact at all times. Traversability for planar curves can be described by the relative curvature κ_R of the circle and the curve at the point of contact. For a planar curve and a circle of radius R_i, the relative curvature can be given as

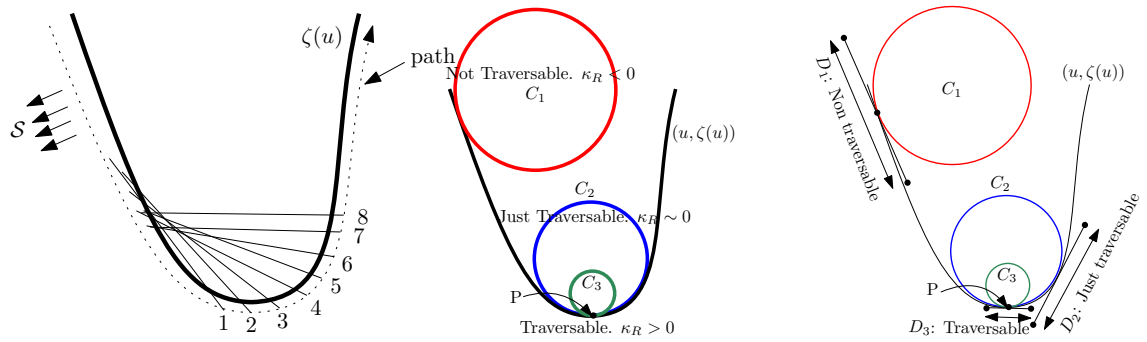
$$\kappa_R = \frac{\zeta_{uu}}{(1+\zeta_u^2)^{3/2}} - \frac{1}{R_i}$$

A curve is traversable if the relative curvature is greater than 0, just traversable if it is equal to zero and not traversable if it is negative. This is explained in figure 17b. This concept would generalize to traversable surfaces when the minimum of the two eigenvalues of the relative curvature matrix[10] is positive for traversability or vice versa. Based on this result, we hypothesize that if curve is traversable by a circle of diameter D_i, then a hyper-redundant robot of link length D_i can move below the curve without intersecting the curve as shown in figure 17c. This would also give us an idea about the largest link length that can be used

[9] In backbone curve approach, this problem is addressed, but at the cost of a very large displacement of the tail point.

[10] The relative curvature matrix of a parametric surface $S(u,v) : \mathbb{S}^2 \to \mathbb{R}^3$ and a sphere of radius R at the point of contact $P = S(u^*, v^*)$ and on the Gaussian frame attached to the sphere(or S) at P is given as: $\kappa_R|_P = \begin{bmatrix} S_{uu}.\mathbf{N} - 1/R & S_{uv}.\mathbf{N} \\ S_{uv}.\mathbf{N} & S_{vv}.\mathbf{N} - 1/R \end{bmatrix}_{(u=u^*, v=v^*)}$ where $\mathbf{N}$ is the normal vector to S at P

in a hyper-redundant robot traversing a given duct. This concept can also be extended to tessellated surfaces with information about the principal curvatures of the surface at a point (e.g. see Meyer et al. [39]).



(a) Infeasible link length (b) Traversability of a curve (c) Feasible link lengths

Figure 17: Feasibility of motion planning with a given link length for a given curve.

6 Conclusions

This paper deals with motion planning for hyper-redundant robots in narrow ducts and confined spaces using a tractrix based redundancy resolution scheme. Motion generated using the methods discussed in this paper emulates realism as opposed to the backbone-based redundancy resolution schemes due to the attenuation of displacement from head to tail of the robot. As opposed to obstacle avoidance problems, the constraint equations used in the minimization scheme for redundancy resolution is not straightforward and they depend on the representation of the ducts. To this end, we explored three different methods to represent ducts in 2D plane–as a combination of ellipses, as a series of connected quadrilaterals and as a bounded planar surface formed from two non-intersecting analytical curves. Methods to formulate inequality constraints which will impose all the joints in the robot to always lie inside the duct are investigated for all the cases. In 3D, the representations of ducts as series of ellipsoids, series of connected cylinders and using point clouds are discussed. The basic formulation as well as formulation for efficient practical implementation are also explained. The methods discussed are demonstrated using three examples of motion planning of 1) a hyper-redundant inspection robot through an industrial pipeline, 2) a highly articulated endoscopic robot through a GI tract and 3) a hyper-redundant robot in search and rescue operation. From the examples, it is shown that the methods developed in the paper can

be effectively used in motion planning of a hyper-redundant robot maneuvering in confined bounded spaces while emulating realism.

In this paper, we have also addressed the computational complexity for the proposed methods. The complexity of the methods scale linearly with the number of links as well as the number of path points for a given problem for all our implementations. For ducts modeled as polyhedra, the worst case complexity has been shown to be $\mathcal{O}(m^{1.5})$, in contrast to $\mathcal{O}(m^{3.5})$ while naively using interior point solvers with m planar constraints. A qualitative comparison has been carried out between the different representations based on pre-processing time, flexibility, computational time and scope of implementation. The locking condition for links in negotiating ducts of very high radius of curvature is discussed. A maximum link length which could overcome the issue, based on the curvature of the duct walls is suggested. In spite of the efficiency of the formulations, in all our implementations we could not reach real time or near real time performances. We believe that this is attributed to our implementation in Matlab which has in-built overheads. A stand alone and a leaner version of an interior point solver tailor made for constrained least norm problems is planned as future work. Finally, research on more efficient algorithms to overcome the locking problem while maintaining motion realism is also currently underway.

Appendix

A Analytical expressions for u, v for quadrilateral patch

$$v = \frac{k_1 - k_2 \pm \sqrt{(k_2 - k_1)^2 - 4k_3k_4}}{2k_3}, \quad u = \frac{(x - {}^xP_{i-1} + v)}{(a_2 + va_3)} \tag{32}$$

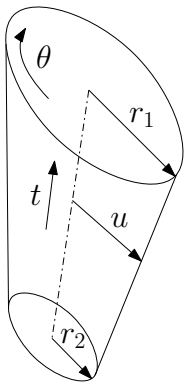
where, $k_1 = (b_1b_2 + b_0b_3)\,, \quad k_2 = (a_1a_2 + a_0a_3)\,, \quad k_3 = (a_1a_3 - b_1b_3)\,, \quad k_4 = (a_0a_2 - b_0b_2)$

$$a_0 = y - {}^yP_{i-1},\ a_1 = {}^yP_{i-1} - {}^yP_i,\ a_2 = {}^xQ_{i-1} - {}^xP_{i-1},\ a_3 = ({}^xQ_i - {}^xP_i) - ({}^xQ_{i-1} - {}^xP_{i-1})$$

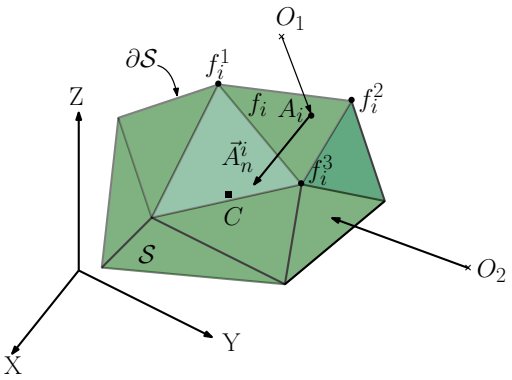
$$b_0 = x - {}^xP_{i-1},\ b_1 = {}^xP_{i-1} - {}^xP_i,\ b_2 = {}^yQ_{i-1} - {}^yP_{i-1},\ b_3 = ({}^yQ_i - {}^yP_i) - ({}^yQ_{i-1} - {}^yP_{i-1})$$

If ${}^xP_{i-1} = {}^xP_i$ and ${}^xQ_{i-1} = {}^xQ_i$,

$$u = \frac{b_0}{a_2}, \quad v = \frac{a_0a_2 - (a_3 + a_2)}{a_1\,(b_0 - a_2) + b_0\,(b_3 - b_2)} \tag{33}$$



(a) Parameters of a general cylinder



(b) Schematic of algorithm 2

Figure 18

B Parametric equation of solid cylinder

The parametric cylinder, as shown in figure 18a, with the parameters $u,\ \theta$ and v is given as:

$$x = C_1(u,v,\theta) = (r_1 m_1 u\cos\theta + m_2)(1-v) + (r_2 n_1 u\cos\theta + n_2)\,v \tag{34}$$

$$y = C_2(u,v,\theta) = (r_1 m_3 u\cos\theta + r_1 m_4 u\sin\theta + m_5)(1-v) + (r_2 n_3 u\cos\theta + r_2 n_4 u\sin\theta + n_5)\,v \tag{35}$$

$$z = C_3(u,v,\theta) = (r_1 m_6 u\cos\theta + r_1 m_7 u\sin\theta + m_8)(1-v) + (r_2 n_6 u\cos\theta + r_2 n_7 u\sin\theta + n_8)\,v \tag{36}$$

where

$$m_1 = \cos{}^1\phi_2,\ m_2 = {}^1x_c, \qquad m_3 = \sin{}^1\phi_1 \sin{}^1\phi_2,\ m_4 = \cos{}^1\phi_1,$$
$$m_5 = {}^1y_c,\ m_6 = -\cos{}^1\phi_1 \sin{}^1\phi_2, \qquad m_7 = \sin{}^1\phi_1,\ m_8 = {}^1z_c$$
$$n_1 = \cos{}^2\phi_2,\ n_2 = {}^2x_c, \qquad n_3 = \sin{}^2\phi_1 \sin{}^2\phi_2,\ n_4 = \cos{}^2\phi_1,$$
$$n_5 = {}^2y_c,\ n_6 = -\cos{}^2\phi_1 \sin{}^2\phi_2, \qquad n_7 = \sin{}^2\phi_1,\ n_8 = {}^2z_c$$

The quantity ${}^1(\cdot)$ and ${}^2(\cdot)$ represent the corresponding parameters of the circles at the ends of the cylinder. ϕ_1 and ϕ_2 are the angles about the Y and X-axes which the plane of the circle is rotated, (x_c, y_c, z_c) is the co-ordinate of the center of the circle.

C An algorithm for classifying a point with respect to a polyhedron

Classification a point with respect to a triangulated domain is a very well known problem and many methods exist, which offer various advantages in terms of computational complexities.

In algorithm 2, we describe a method, to classify a given set of points O with respect to a polyhedron $\mathcal{P}$ as inside (O_{in}) and outside (O_{out}). The implementation of our algorithm[11] is $\mathcal{O}(m)$. The algorithm can be visualized from figure 18b.

Purpose : To classify a set of points O with respect to boundary $\partial\mathcal{P}$
Input: The set O, $O \in \Re^3$ and $O \equiv O_{in} \cup O_{out}$
Output: O_{in}, O_{out}

1: Obtain C, the center of $\mathcal{P}$ by taking the mean of the vertices of the N facets constituting $\partial\mathcal{P}$
2: **for** $i = 1, 2, ..., N$ **do**
3: Calculate the normal to the i^{th} facet, $A_n^i = (f_i^1 - f_i^2) \times (f_i^2 - f_i^3)$
4: Choose a point A^i on the i^{th} facet and move A_n^i to A^i
5: Ensure that A_n^i is directed inside, towards C
6: $H(i) = \langle O_i \vec{-} A_i, A_n^i \rangle$
7: **end for**
8: **if** $H(i) \geq 0 \ \forall i$ **then**
9: Classify $O_1 \in O_{in}$ as inside, otherwise classify $O_1 \in O_{out}$
10: **end if**

Algorithm 2: Algorithm for classifying points as inside (O_{in}) or outside (O_{out}) of a triangulated domain.

7 Acknowledgements

The second author would like to acknowledge ISRO-IISc Space Technology Cell for partially funding the research through the grant ISTC/MME/AG/0394.

References

[1] G. S. Chirikjian, J. W. Burdick, Hyper-redundant robot mechanisms and their applications, in: Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on, IEEE, 1991, pp. 185–190.

[2] A. Wolf, H. B. Brown, R. Casciola, A. Costa, M. Schwerin, E. Shamas, H. Choset, A mobile hyper redundant mechanism for search and rescue tasks, in: Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, Vol. 3, IEEE, 2003, pp. 2889–2895.

[11] We have used a modified version of the Matlab function inhull.m, made available by John D'Errico for free usage.

[3] R. R. Murphy, S. Tadokoro, D. Nardi, A. Jacoff, P. Fiorini, H. Choset, A. M. Erkmen, Search and rescue robotics, in: Springer Handbook of Robotics, Springer, 2008, pp. 1151–1173.

[4] A. B. Slatkin, J. Burdick, W. Grundfest, The development of a robotic endoscope, in: Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on, Vol. 2, IEEE, 1995, pp. 162–171.

[5] R. D. Howe, Y. Matsuoka, Robotics for surgery, Annual Review of Biomedical Engineering 1 (1) (1999) 211–240.

[6] G.-B. Cadiere, J. Himpens, O. Germay, R. Izizaw, M. Degueldre, J. Vandromme, E. Capelluto, J. Bruyns, Feasibility of robotic laparoscopic surgery: 146 cases, World Journal of Surgery 25 (11) (2001) 1467–1477.

[7] J. Burgner-Kahrs, D. C. Rucker, H. Choset, Continuum robots for medical applications: A survey, IEEE Transactions on Robotics 31 (6) (2015) 1261–1280.

[8] E. Paljug, T. Ohm, S. Hayati, The JPL serpentine robot: a 12-DoF system for inspection, in: Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on, Vol. 3, IEEE, 1995, pp. 3143–3148.

[9] R. L. William II, J. B. Mayhew IV, Obstacle-free control of the hyper-redundant NASA inspection manipulator, in: Proc. of the Fifth National Conf. on Applied Mechanics and Robotics, 1997, pp. 12–15.

[10] L. Gargiulo, P. Bayetti, V. Bruno, J.-C. Hatchressian, C. Hernandez, M. Houry, D. Keller, J.-P. Martins, Y. Measson, Y. Perrot, et al., Operation of an ITER relevant inspection robot on Tore Supra tokamak, Fusion Engineering and Design 84 (2-6) (2009) 220–223.

[11] S. Ma, S. Hirose, H. Yoshinada, Development of a hyper-redundant multijoint manipulator for maintenance of nuclear reactors, Advanced Robotics 9 (3) (1994) 281–300.

[12] A. Liegeois, Automatic supervisory control of the configuration and behaviour of multibody mechanisms, IEEE Transactions on Systems, Man and Cybernetics 7 (12) (1977) 868–871.

[13] J. Baillieul, J. Hollerbach, R. Brockett, Programming and control of kinematically redundant manipulators, in: Decision and Control, 1984. The 23rd IEEE Conference on, Vol. 23, IEEE, 1984, pp. 768–774.

[14] A. A. Maciejewski, C. A. Klein, Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments, The International Journal of Robotics Research 4 (3) (1985) 109–117.

[15] Y. K. Hwang, N. Ahuja, Gross motion planning—a survey, ACM Computing Surveys (CSUR) 24 (3) (1992) 219–291.

[16] S. Chiaverini, Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators, IEEE Transactions on Robotics and Automation 13 (3) (1997) 398–410.

[17] L. Sciavicco, B. Siciliano, A solution algorithm to the inverse kinematic problem for redundant manipulators, IEEE Journal on Robotics and Automation 4 (4) (1988) 403–410.

[18] Z. Mao, T. C. Hsia, Obstacle avoidance inverse kinematics solution of redundant robots by neural networks, Robotica 15 (1) (1997) 3–10.

[19] G. Antonelli, S. Chiaverini, Fuzzy redundancy resolution and motion coordination for underwater vehicle-manipulator systems, IEEE Transactions on Fuzzy Systems 11 (1) (2003) 109–120.

[20] B. Dasgupta, A. Gupta, E. Singla, A variational approach to path planning for hyper-redundant manipulators, Robotics and Autonomous Systems 57 (2) (2009) 194–201.

[21] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, in: Autonomous Robot Vehicles, Springer, 1986, pp. 396–404.

[22] V. J. Lumelsky, A. A. Stepanov, Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape, Algorithmica 2 (1-4) (1987) 403–430.

[23] H. Choset, Coverage for robotics–a survey of recent results, Annals of Mathematics and Artificial Intelligence 31 (1-4) (2001) 113–126.

[24] J.-C. Latombe, Robot motion planning, Vol. 124, Springer Science & Business Media, 2012.

[25] S. Sreenivasan, P. Goel, A. Ghosal, A real-time algorithm for simulation of flexible objects and hyper-redundant manipulators, Mechanism and Machine Theory 45 (3) (2010) 454–466.

[26] M. S. Menon, G. Ananthasuresh, A. Ghosal, Natural motion of one-dimensional flexible objects using minimization approaches, Mechanism and Machine Theory 67 (2013) 64–76.

[27] M. S. Menon, V. Ravi, A. Ghosal, Trajectory planning and obstacle avoidance for hyper-redundant serial robots, Journal of Mechanisms and Robotics 9 (4) (2017) 041010.

[28] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.

[29] H. Edelsbrunner, E. P. Mücke, Three-dimensional alpha shapes, ACM Transactions on Graphics (TOG) 13 (1) (1994) 43–72.

[30] D.-T. Lee, B. J. Schachter, Two algorithms for constructing a Delaunay triangulation, International Journal of Computer & Information Sciences 9 (3) (1980) 219–242.

[31] MATLAB, version 9.1 (R2016b), The MathWorks Inc., Natick, Massachusetts, 2016.

[32] Blender Online Community, Blender - a 3D modelling and rendering package, Blender Foundation, Blender Institute, Amsterdam (2018).
URL http://www.blender.org

[33] V. M. Spitzer, D. G. Whitlock, The visible human dataset: the anatomical platform for human simulation., The Anatomical Record: An Official Publication of the American Association of Anatomists 253 (2) (1998) 49–57.

[34] J. Cao, A. Tagliasacchi, M. Olson, H. Zhang, Z. Su, Point cloud skeletons via Laplacian based contraction, in: Shape Modeling International (SMI 2010), IEEE, 2010, pp. 187–197.

[35] H. Ananthanarayanan, R. Ordóñez, Real-time inverse kinematics of (2n+ 1) DoF hyper-redundant manipulator arm via a combined numerical and analytical approach, Mechanism and Machine Theory 91 (2015) 209–226.

[36] S. Wright, J. Nocedal, Numerical optimization, Springer Science 35 (67-68) (1999) 7.

[37] N. Chakraborty, J. Peng, S. Akella, J. E. Mitchell, Proximity queries between convex objects: An interior point approach for implicit surfaces, IEEE Transactions on Robotics 24 (1) (2008) 211–220.

[38] N. Chakraborty, Modeling of wheeled mobile robots on uneven terrain, MSc (Engg.) Thesis, Dept. of Mechanical Engg., IISc Bangalore, 2003.

[39] M. Meyer, M. Desbrun, P. Schröder, A. H. Barr, Discrete differential-geometry operators for triangulated 2-manifolds, in: Visualization and Mathematics III, Springer, 2003, pp. 35–57.