

Motion planning of hyper-redundant robot through confined paths

K. P. Ashwin*, Arkadeep N. C. *, and A. Ghosal †

Abstract

Keywords:6 keywords

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

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

1 Introduction

In a reachable workspace of a serial robot in 3D, since a given position and orientation of end-effector can be achieved using a maximum of six joints, additional joints would create a redundant system. Such robots are generally termed redundant serial robots. While robots with 6 joints or less can achieve a position and orientation in space with limited configurations, there are theoretically infinite possibilities for a hyper-redundant robot to achieve the same. These additional degrees of freedom can be effectively used in situations where the robot has to selectively choose configurations without altering the end-effector pose, such as robotic manipulation through confined spaces. This renders the use of highly articulated hyper-redundant robots in applications such as exploring earthquake hit regions for search and rescue operations, remotely operated robotic arms for minimally invasive laparoscopic and endoscopic surgeries and inspection of industrial pipelines and ducts[cite refs.]. A large number of redundant joints also make the robots analogous in morphology to snake, elephant trunk, tentacle etc. and hence, a topic of particular interest in bio-inspired robotics. Identifying a suitable configuration from the many possible configurations, given the position and orientation of the end-effector constitutes the redundancy resolution of hyper-redundant robots. If the path through which the end-effector has to follow is predetermined, then the redundancy resolution problem also constitutes the motion planning of the robot.

Redundant robots are extensively used for applications which require additional dexterity.

1.
 - What are redundant robots
 - What is the use of redundancy
 - examples of applications of redundant robots
 - what is the downside?
 - what is called redundancy resolution. Say that the displacement of tip is specified and the co-ordinates(or the joint angles) of the subsequent links should be found out.
2.
 - State that the major advantage of red. robots is obstacle avoidance
 - What is the first redundancy resolution scheme? Did it avoid obstacles?
 - How is obstacle avoidance included in red. resolution
 - Give a review of obstacle avoidance algorithms

- state why tractrix is better, what is tractrix
 - how obstacle avoidance is easily implemented
3. • State that particularly interesting is the case of motion through ducts, endoscopy and inspection
- How the tractrix method as is is difficult to implement
 - Explain the motive of this paper
 - Contents of the paper

2 Overview of tractrix based motion planning

Consider a rigid link of length L_0 positioned in a 2D plane, initially aligned to the Y-axis as shown in the figure[ref fig]. The co-ordinates of the ‘head’ of the link is given as $\mathbf{X}_h = [X_h, Y_h]^T$ and the co-ordinates of the ‘tail’ as $\mathbf{X}_t = [X_t, Y_t]^T$. If the head is displaced to the co-ordinate $\mathbf{x}_h = [x_h, y_h]$ along the positive X – axis by t units, the tail of the link can lie anywhere on the circumference of a circle centered at the co-ordinate $(t, 0)$ with radius L_0 . If we assign a rule that the velocity of the tail of the link is always directed towards the length of the link, we get two diametrically opposite points on the circle. The continuous path traced by the tail point is the well known tractrix curve given by the expressions[ref]:

$$\mathbf{x}_t = [x(t), y(t)] = \left[t - L_0 \tanh \frac{t}{L_0}, L_0 \operatorname{sech} \frac{t}{L_0} \right] \quad (1)$$

The extension of tractrix equation along motion in arbitrary direction as well as an algorithm to calculate the same in 3-D can be found in [ref] and [ref]. In case of multiple links connected to each other as in the case of hyper-redundant robot, or a one dimensional object approximated as a series of connected linkages the algorithm can be applied iteratively from the head to tail as shown in [ref]. By moving along the tractix curve, the tail moves the minimum distance with respect to its initial position. Also, the displacement $\|\mathbf{x}_h - \mathbf{X}_h\| \geq \|\mathbf{x}_t - \mathbf{X}_t\|$ which means that the displacement attenuates from the displaced link to end of the chain in case of serially connected links[fig2]. Due to the minimal displacement of the tail, the motion can be imagined as the one with high lateral resistance on the link and less resistance in the direction of motion of link.

It is shown in [ref] that for a single rigid link, tractrix curve can also be obtained by minimizing an L^2 metric which is essentially the displacement of the tail from its initial

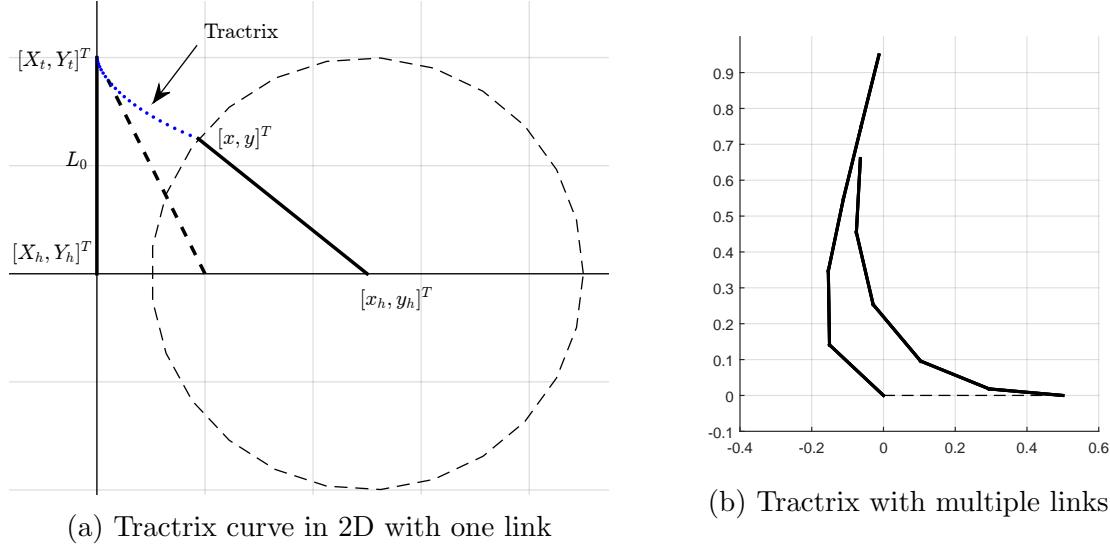


Figure 1: Tractrix in a plane

position subject to the condition that the length of link is always preserved. i.e, the co-ordinates of the tail can be obtained from the following minimization problem:

$$\arg \min_{\mathbf{x}_t} \|\mathbf{x}_t - \mathbf{X}_t\| \quad (2)$$

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

An advantage of expressing tractrix as a minimization problem is that we can add more constraints to the above expression and hence, control the position of the tail point. Though the resulting curve may not necessarily be a tractix curve, the motion of the tail will appear realistic [ref]. For the motion of a link which minimizes its tail velocity (displacement of the tail co-ordinates), obstacle avoidance is achieved by formulating the problem as:

$$\arg \min_{\mathbf{x}_t} \|\mathbf{x}_t - \mathbf{X}_t\| \quad (3)$$

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

$$f(\mathbf{x}) \succeq 0$$

where $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ is the analytical equations of the boundaries of the surfaces which are to be avoided. For example, if the tail is to avoid a single obstacle represented by a circle with center (x_c, y_c) , the expression $f(x) = (x - x_c)^2 + (y - y_c)^2 - r^2 > 0$ ensures that the point x always lies outside the circle of radius r . Complex objects can be modelled as a combination of super-ellipses as shown in [ref]. In this case, $f(x)$ will be a vector of all boundary equations

$\mathbf{f}(x) = [f_1(x), f_2(x), \dots, f_m(x)]^T$ [fig3]. It is also worth noting that the value of function will increase or decrease as the point is farther from the curve $f(x)$; the value being zero on the curve. Hence, this approach can also be imagined as a geometric potential field, with zero potential only at the surface of the obstacle.

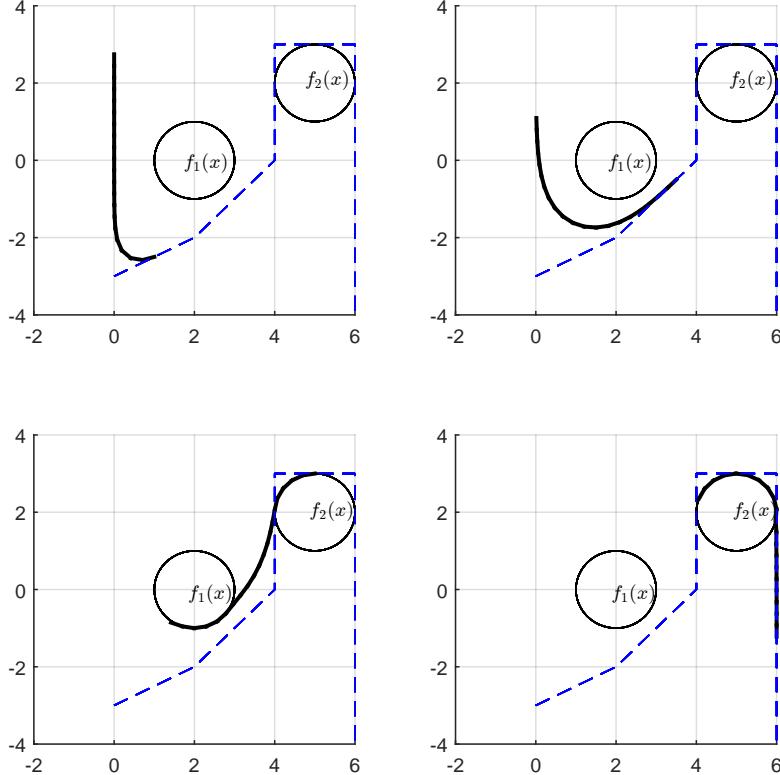


Figure 2: Obstacle avoidance in a plane

In the case of path planning of hyper-redundant manipulator, a majority of problems deal with simulating the motion of the robot while traversing through a confined space within a narrow bounded path, which we term as a “*duct*” for the remainder of the paper. The problem of planning the motion of the robot through a duct may be specified as:

$$\arg \min_{\mathbf{x}_t} \|\mathbf{x}_t - \mathbf{X}_t\| \quad (4)$$

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

$$C_{\text{ineq}} : f(x) < 0$$

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 how confined space motion is achieved in different cases.

2.1 Nature of the optimization problem

In the previous section we have discussed how tractrix based motion planning can be formulated as a constrained optimization problem, which entails a discussion on the suitability of the posed optimization problem and guarantee that the solution is unique and physically tractable. To address this, we will prove that the problem in equation (4) and the variations of the same used in the current work can be posed as convex problems. A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is *convex* if and only if the following two conditions hold:

- a The domain of f , $\text{dom } f$ is a convex set, and,
- b For all $x, y \in \text{dom } f$ and θ with $0 \leq \theta \leq 1$

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y) \quad (5)$$

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} ¹. Therefore, if the conditions a and b associated with the definition of a convex function are satisfied for \mathcal{S} and the L_2 norm, then the solution obtained for (put optimization equation here) is unique.

It is a well known result that any p-norm $\|x\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$, $x \in \mathbb{R}$ is a convex function for $p \geq 1$. The fact that a p-norm is convex can be shown by using the scalability and the triangle inequality associated with a normed vector-space. With θ being a scaling parameter, equation (5) simplifies exactly to the triangle equality for norms. Also, any p-norm is a valid norm on \mathbb{R} for $p \geq 1$ follows from the Minkowski inequality². This leaves us to show that the feasible set \mathcal{S} is convex. For all our modeling examples, we choose \mathcal{S} to be convex by assigning convex boundaries to it or, expressing \mathcal{S} as intersections of closed convex sets. However, as is, none of the problems posed in equations (2) to (4) are convex because of the

¹Assuming \mathcal{S} to be a closed set allows the object to physically touch the boundaries of the duct

²For a measure space \mathbb{S} , $p \geq 1$ and f, g are members of the Lebesgue space $L_P(\mathbb{S})$, the following inequality holds: $\|f + g\|_p \leq \|f\|_p + \|g\|_p$

non-linear equality constraints associated with them, which guarantees that the length of a segment is constant. To pose it as a convex problem we approximate the constraint with linear constraints as described in section(**put complexity section here**). For the cases of path planning in 2-D and 3-D, 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.

3 Motion planning through planar ducts

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

3.1 Representation of duct using super-ellipses

One method to represent a duct is by overlapping a series of super-ellipses as shown in [fig5b]. This is the most simple and straightforward means of representation as shown in [ref Midhun]. In Cartesian co-ordinate system R^2 , the contour of super-ellipse obeys the following equation:

$$f(\mathbf{x}) = f(x, y) : \left| \frac{x - x_c}{a} \right|^n + \left| \frac{y - y_c}{b} \right|^n - 1 = 0 \quad (6)$$

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 ($f_i(x)$, $i = 1, 2, \dots, m$), only one of them will be satisfied for the point to be inside the duct. In practical implementation, we can say that 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

$$T_i = \begin{bmatrix} \cos \phi_i & -\sin \phi_i & 0 & x_i \\ \sin \phi_i & \cos \phi_i & 0 & y_i \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

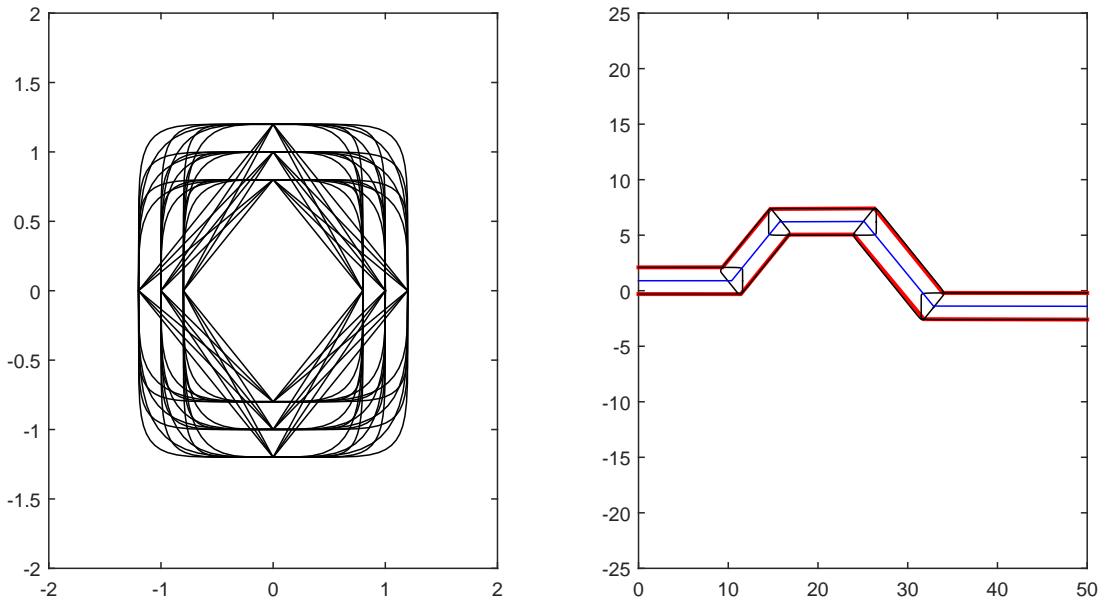


Figure 3: Super-ellipses and a duct modelled as combination of super-ellipses

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

$$C_{\text{ineq}} : \min(g_i(\mathbf{x}_t)) < 0, \quad i = 1, 2, \dots, m \quad (8)$$

An example of single link and multi-segmented chain passing through the duct is shown in [fig5]. Motion of a unit link with and without constraint is shown in [fig6]. The negative gradient (descend direction) of the inequality constraint function is also shown in the figure. The method shown here is quite fast and scalable as explained in section ??, while the majority of time taken for the scheme being 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.

3.2 Representation of duct as a 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 6. The individual quadrilateral patches may

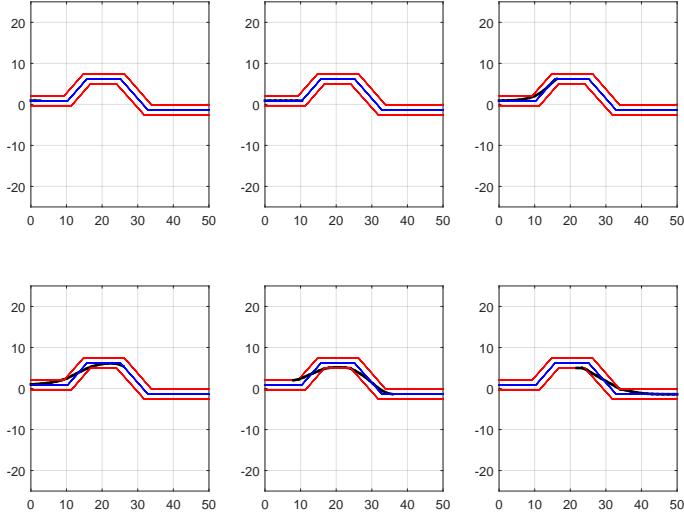


Figure 4: Motion through duct modelled as combination of super-ellipses

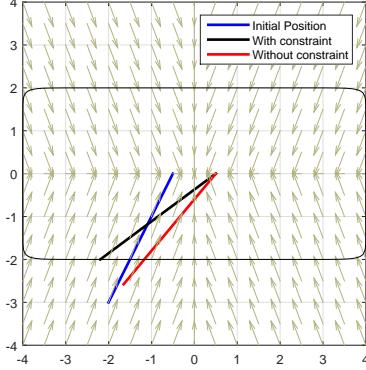


Figure 5: Effect of gradient of inequality constraint in pulling the tail into the duct

be denoted as A_1, A_2, \dots, A_n , each bounded by the line segments defined by the points $(\mathbf{P}_0, \mathbf{P}_1), (\mathbf{P}_1, \mathbf{P}_2), \dots, (\mathbf{P}_{n-1}, \mathbf{P}_n)$ for the curve ζ_1 and $(\mathbf{Q}_0, \mathbf{Q}_1), (\mathbf{Q}_1, \mathbf{Q}_2), \dots, (\mathbf{Q}_{n-1}, \mathbf{Q}_n)$ for the curve ζ_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:

$$A_i^1 x_t + A_i^2 y_t + B_i < 0 \quad i = 1, 2, 3, 4 \quad (9)$$

$$(10)$$

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

$$C_{\text{ineq}} : [\mathbf{A}] \mathbf{x}_t + \mathbf{B} < 0 \quad (11)$$

where $[\mathbf{A}]$ is a 4×4 matrix and \mathbf{B} is a 4×1 vector. But we propose a more convenient method for practical applications which is as follows:

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

$$\mathbf{x}_i(u, v) = [\mathbf{P}_{i-1} + (1-u)\mathbf{P}_i] (1-v) + [\mathbf{Q}_{i-1} + (1-u)\mathbf{Q}_i] (v) \quad (12)$$

in parameters u and v . If the vertices of the quadrilateral are given by $\mathbf{P}_i = [x P_i, y P_i]^T$ and $\mathbf{Q}_i = [x Q_i, y Q_i]^T$, then the analytical expressions for the terms u and v , given the value of \mathbf{x}_i , can be obtained by solving 12 (see Appendix). The values of u, v can be used to classify the point with respect to the surface patch A_i ³.

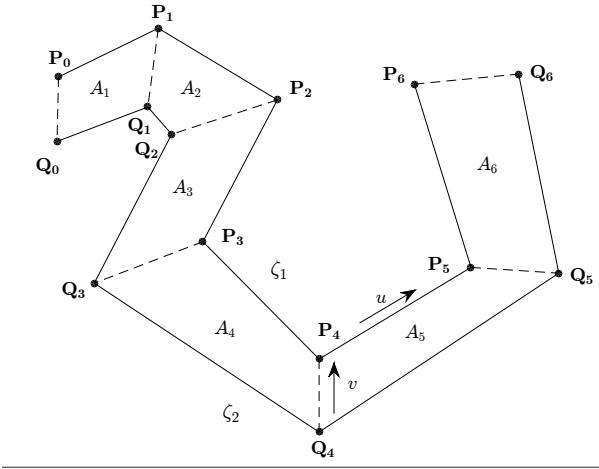


Figure 6: Duct represented by stitched quadrilaterals

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

$$C_{\text{ineq}} : 0 \leq u \leq 1, \quad 0 < v < 1 \quad (13)$$

³It 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 $(u_1, v_1), (u_2, v_2), \dots, (u_m, v_m)$ etc. for the m number of patches A_1, A_2, \dots, A_m and consequently, m set of conditions. But 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. Including a switching statement in the optimization code will be inefficient especially when the point to be classified is close to the common boundary between two patches.

In order to provide a gradient to the constraint which will direct the point 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

$$h = \|\mathbf{x}_{\zeta_1} - \mathbf{x}_t\|^2 + \|\mathbf{x}_{\zeta_2} - \mathbf{x}_t\|^2 - \|\mathbf{x}_{\zeta_1} - \mathbf{x}_{\zeta_2}\|^2 \quad (14)$$

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 $\mathbf{h} = [h_1, h_2, \dots, h_m]$ should be negative for classifying the point with respect to the duct, as in the case of previous section. But as we can see, 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:

$$\chi(t) = \begin{cases} 0, & t < 0 \\ 1, & 0 \leq t \leq 1 \\ 0, & t > 1 \end{cases} \quad (15)$$

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:

$$C_{\text{ineq}} : [\chi(\hat{\mathbf{u}})]^T \mathbf{h} < 0 \quad (16)$$

where $\hat{\mathbf{u}} = [\hat{u}_1, \hat{u}_2, \dots, \hat{u}_m]^T$ 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 is shown in [fig11] and the effect of the added inequality constraint 16 to pull the tail end of the link which is initially positioned outside the duct, is shown in [fig12]. Apart from being more flexible, another advantage of representing a 2D duct as a set of connected quadrilaterals in this parameteric form is that by setting the limits of the parameter v to $0 + \delta < v < 1 - \delta$, $\delta < 0.5$, 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.

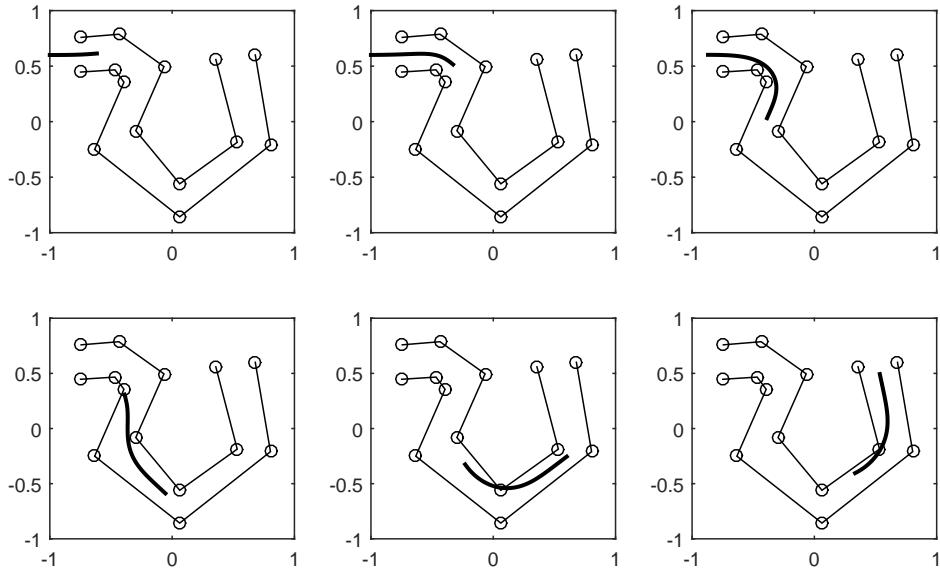


Figure 7: Example of constrained motion with stitched quadrilaterals

3.3 Representation of duct using 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 will simply be,

$$\mathbf{x}_i(u, v) = \zeta_1(u)(1 - v) + \zeta_2(u)(v) \quad (17)$$

For example, [fig11] shows a 2D duct defined by two curves $\zeta_1(u) = [u, \sin(u)]^T$ and $\zeta_2(u) = [u, \sin(u + \frac{\pi}{8}) + 1]^T$ and a path chosen midway between the two curves. The equation of the surface generated by this curves will be

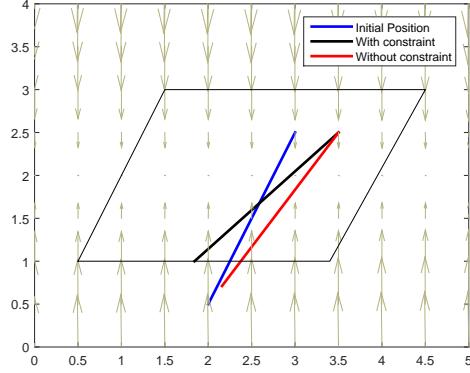


Figure 8: Effect of gradient of inequality constraint in pulling the tail into the quadrilateral duct

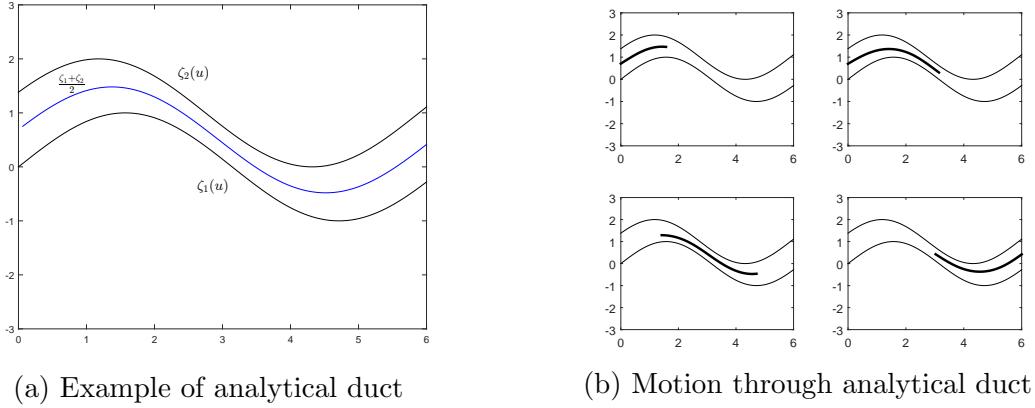


Figure 9: Tractrix based algorithm on analytical duct

$$\begin{bmatrix} x(u, v) \\ y(u, v) \end{bmatrix} = \begin{bmatrix} u \\ \sin(u) + [\sin(u + \frac{\pi}{8}) - \sin(u) + 1]v \end{bmatrix} \quad (18)$$

which has the analytical solution for u and v :

$$u = x$$

$$v = \frac{y - \sin(x)}{\sin(x + \frac{\pi}{8}) - \sin(x) + 1}$$

In this case, we will solve the equations:

$$\min_{\mathbf{x}_t} \|\mathbf{x}_t - \mathbf{X}_t\| \quad (19)$$

$$\text{sub: } \|\mathbf{x}_h - \mathbf{x}_t\| - L_0 = 0$$

$$0 < v|_{\mathbf{x}_t} < 1 \quad (20)$$

An example movement of hyper-redundant manipulator through the duct is shown in [fig12]. 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. Also, since multiple solutions may be possible for such cases, the correctness of the solution would heavily depend on the choice of initial guess provided⁴. The equation to be used is 13.

4 Motion planning through 3D ducts

In this section, we will explain a few methods to represent ducts in 3D and how motion planning is achieved in the same.

4.1 Representation of duct using combination of super-ellipsoids

In Cartesian co-ordinate system in R^3 , the surface of a super-ellipsoid follows the equation:

$$f(x, y, z) : \left[\left\{ \left(\frac{x}{a} \right)^{\frac{2}{e}} + \left(\frac{y}{b} \right)^{\frac{2}{e}} \right\}^{\frac{e}{n}} + \left(\frac{z}{c} \right)^{\frac{2}{n}} \right]^{\frac{n}{2}} - 1 = 0 \quad (21)$$

By changing the parameters a, b, c and n , we get different closed surfaces as shown in [fig13]. By combining different super-ellipse shapes, we can generate a 3D duct as shown in [fig14]

The procedure to calculate the inside-outside condition is same as that of the method described in section [ref sec]. The inequality condition will be [8]. An example problem with duct approximated using super-ellipsoids is shown in [fig15]. As mentioned in the case for super-ellipses, solution to the path planning problem with ducts represented by super-ellipsoids is fast (also explained in section ??). Identifying the shapes which fit the duct, is also same as the method mentioned in section ??.

4.2 Representation of 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 (refer Appendix for the detailed expressions):

$$x = C_1(u, t, \theta), \quad y = C_2(u, t, \theta), \quad z = C_3(u, t, \theta) \quad (22)$$

⁴The same argument will also hold for analytical surfaces spanned in 3D and hence is not studied further.

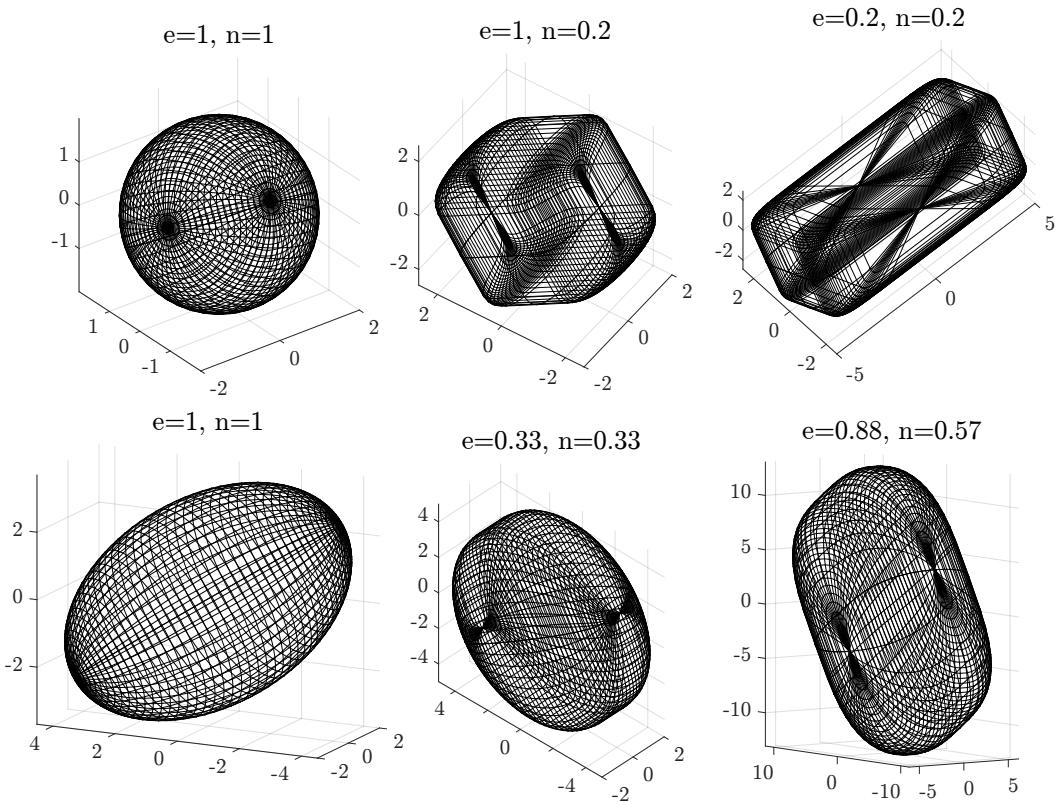


Figure 10: Super-ellipsoids

where the parameters u, t and θ varies along the radial, axial and circumferential direction of the cylinder respectively (refer [fig ??]). Similar to the representation in [sec], $0 \leq u < 1$ and $0 < t < 1$ classifies the point as inside the cylinder. The constrained inequality 16 generated will also be valid for cylinders. The quantity h_i will simply be $h = \hat{u}_i - 1$, which will show the same characteristics as defined by the value of h_i in equation 14. The constraint inequality, hence takes the form:

$$C_{\text{ineq}} : [\chi(\hat{\mathbf{t}})]^T \mathbf{h} < 0 \quad (23)$$

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. Representation of duct using ellipsoids and connected cylinders is shown figure 11.

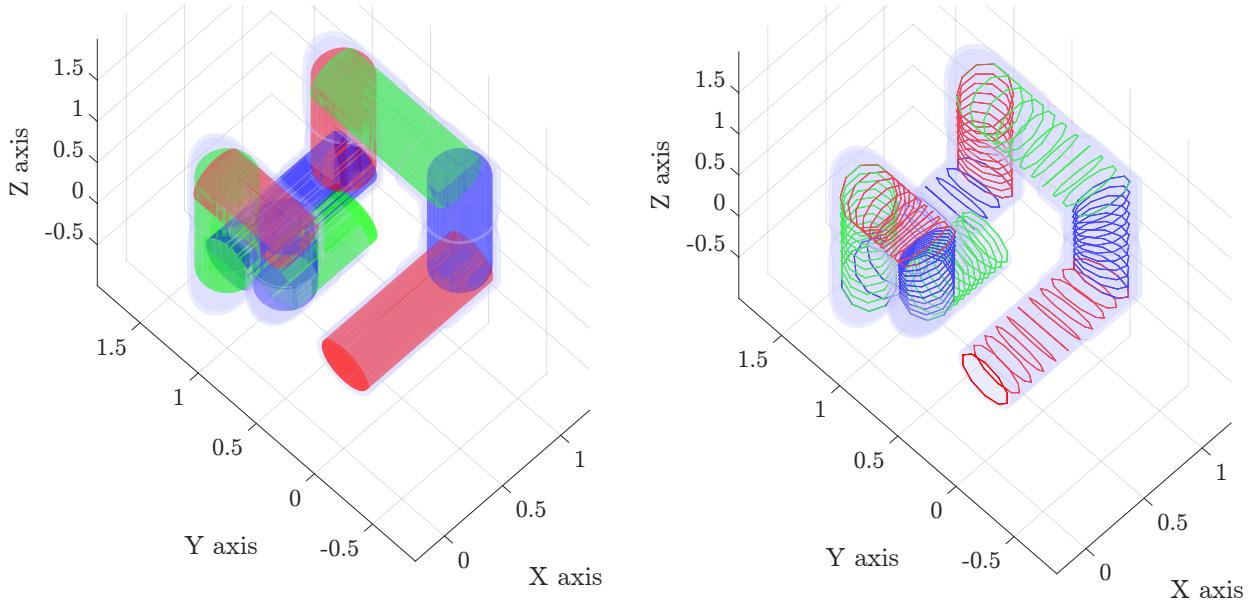


Figure 11: Representation of 3D duct using super-ellipsoids and analytical cylinders

4.3 Representation of duct as point clouds

The most direct way of representing the duct, in a practical sense, would be as a point cloud ($\mathbf{R}_i, i = 1, 2, \dots, N$ where N is the number of points in the cloud) such as obtained from a LiDAR system or a depth map. Subsequently, it would be possible to process the raw data to obtain the geometric representation of the point of cloud as a Stereolithographic formatted file (STL) or the like. Using the current framework, it is possible to pose the motion planning problem in the following form:

$$\min_{\mathbf{x}_t} \|\mathbf{x}_t - \mathbf{X}_t\| \quad (24)$$

$$\text{sub: } \|\mathbf{x}_h - \mathbf{x}_t\| - L_0 = 0 \\ [\mathbf{A}]\mathbf{x}_t + \mathbf{B} \leq 0 \quad (25)$$

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 adjacent points in the cloud. The i^{th} equation, $A_i^1x_t + A_i^2y_t + A_i^3z_t + B_i$ 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 1 described in section 7.3. The attracting gradient which ensures the point to be inside the duct—as 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

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

where $w(\bar{\mathbf{R}})$ 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 surface⁵.

Another interesting prospect of this method is its application in obstacle avoidance where the left hand side of the equation Equation (26) will be greater than zero. The obstacle avoidance methods which follow the equation given by Equation (3) is often limited to shapes which can be modelled analytically. Though the algorithm shown here is limited to convex shapes, they can be applied to more complex and non-symmetric shapes. Example of motion planning through convex, non-analytical point clouds is shown in figure ??.

[fig ??.. Motion of HR- robot through STL duct (to be added)]

5 Examples and discussion

In this section, we present two practical examples where the above mentioned methods are applied and some discussion on the implementation of the above mentioned algorithms.

5.1 Examples of motion planning through ducts

5.1.1 Motion planning for inspection robots

One major use of hyper-redundant robots is in inspection of ducts such as industrial pipelines. By approximating the pipe-line as ellipsoids or cylinders, methods mentioned above are made use of, to plan the motion of a hyper-redundant robot through the same. 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 12.

⁵Unlike the previous classification problems, the bounding surface will also be considered as inside the surface in this case.

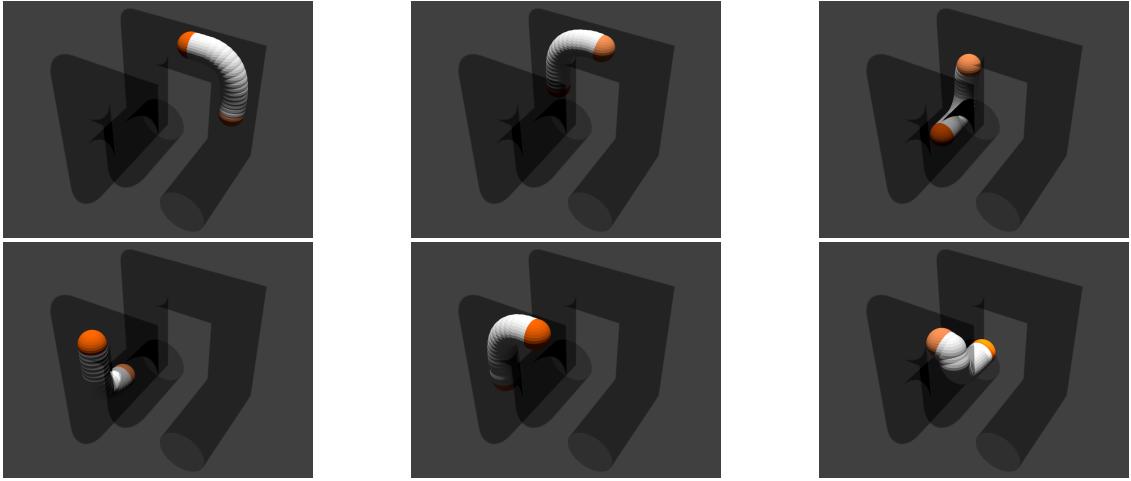


Figure 12: Motion of hyper-redundant manipulator through a duct represented as connected cylinders

5.1.2 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 planning 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 stereolithographic data of GI tract obtained from ???. For demonstration, both the methods presented in [sec] and [sec] 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. Representation of GI tract as series of super-ellipsoids is shown in [fig19]

For representing GI tract as cylinders, we first found out the medial axis of the duct using ???. 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 [22]. Representation of GI tract as a series of connected cylinders is shown in [fig20]

The realistic motion simulation of endoscope through GI tract is shown in [fig21]

5.1.3 Motion planning for search and rescue operations

Exploring obstacle-cluttered environments 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.



(a) GI tract as collection of super-ellipsoids



(b) GI tract with connected cylinder(?? change??)

Figure 13: Representation of GI tract in two methods

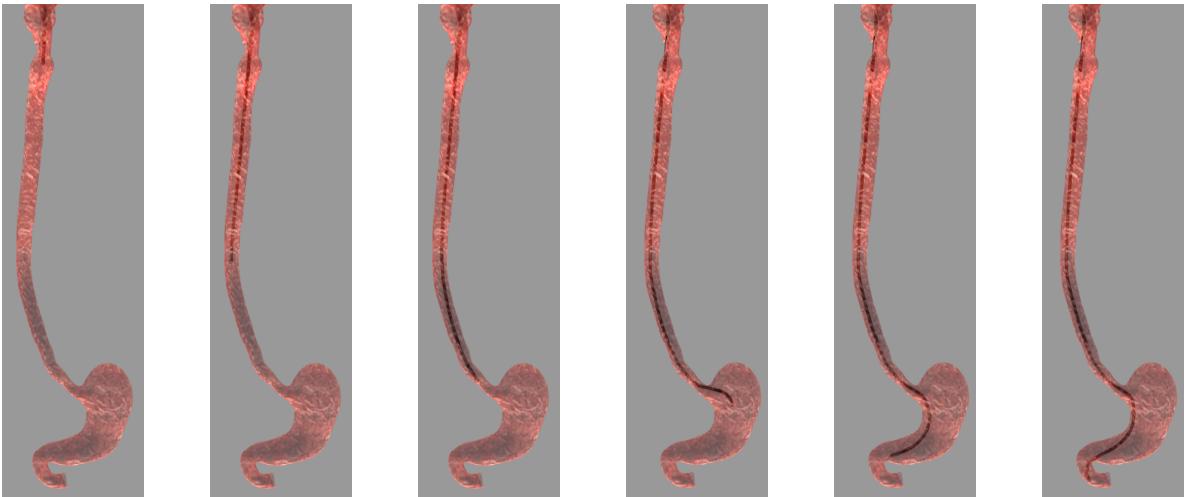


Figure 14: Motion of endoscope through GI tract

With reference to [fig ??], 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 1,2 and 3 located inside the blue region and 4) Exit without colliding with the complex shapes given by objects 4 and 5. Here, the exploration problem is effectively tackled using the following constraints added to the opti-

mization problem (refer Fig ??): 1) An ellipsoid of the size of the hole located in the outer wall 2) collection of cylindrical ducts to represent the vent, 3) STL models for the objects 1,2 and 3 and 4) Ellipsoid at the last bend in the path to avoid collision with the complex shapes of objects 4 and 5 (orange color). It is easy to verify that the geometry based global motion planning techniques for obstacle avoidance existing in literature will entail a very involving problem formulation [ananthanarayanan], while using the methods proposed in this paper, the problem can be addressed with only 6 constraints. The six constraints in this case are actively selected so that they become active only when the position of the head is in the vicinity of the objects/ shapes. Figure [??] shown the simulated results for the motion of manipulator along a chosen path.

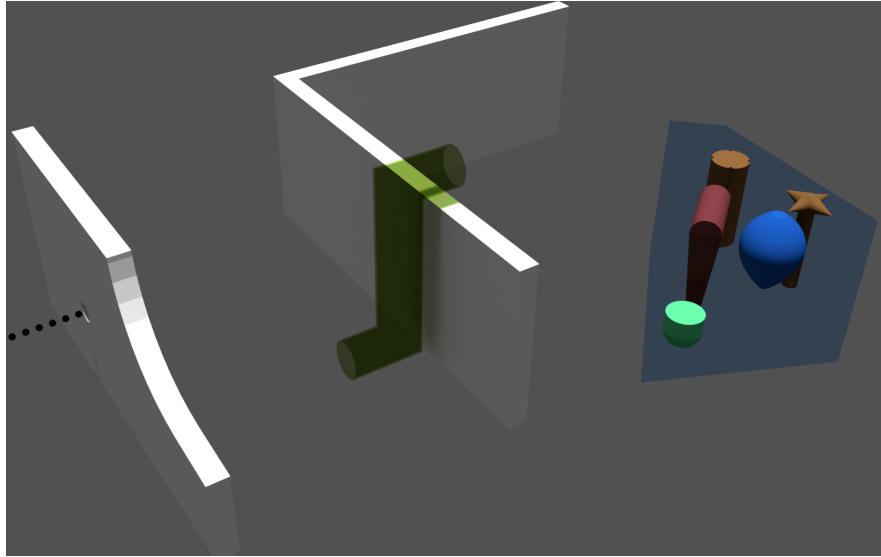


Figure 15: Scene of search and rescue problem

5.2 Computational complexity of the proposed scheme

We want to put the following in this section in the order. Work on this is currently underway.

- Show that the problem in equation (2) can be reformulated as a convex problem with a first order approximation of the equality constraint and a trust region constraint
- Show the Lagrange dual problem and show that the constraints are Slater's conditions compliant and self concordant. This would also justify our use of a primal-dual feasible interior point method

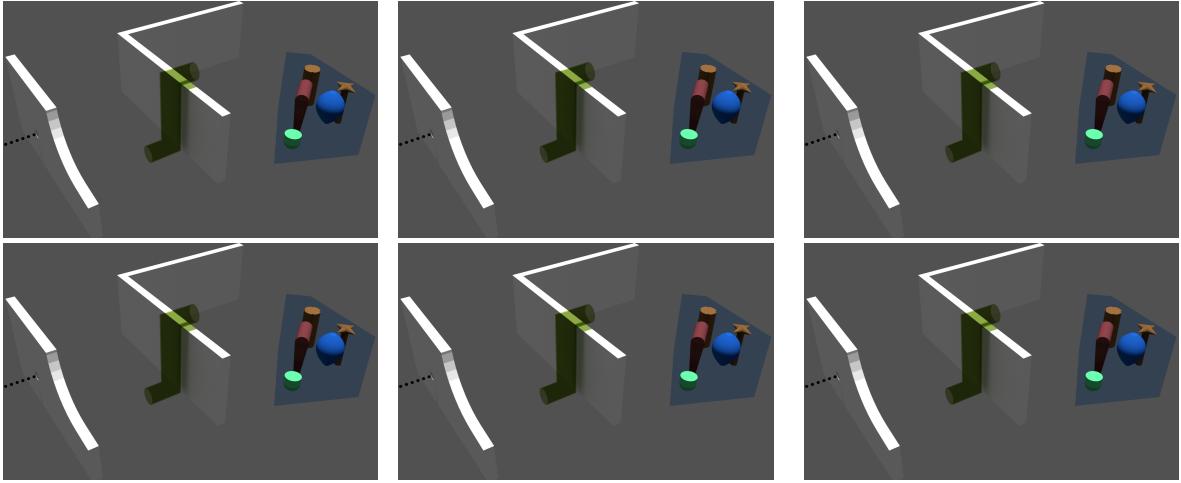


Figure 16: Motion of hyper-redundant robot in confined spaces ?? Fig will be edited??

- Therefore, the interior point algorithm would take certain number of iterations always bounded by a constant like 10-15.
 - Next, we show that at each iteration of the interior point algorithm, the computation time is constant. Therefore, the problem is LINEARLY scalable with the number of links and number of points on the planned path... i.e. for a particular choice of link number and points on the path, it can have a real time implementation on a computer.
 - Finally, we would plot the complexity for a 2D and a 3D case.
-

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[opt book], to be used in the following discussion.

Self Concordance: A convex function $f : \mathbb{R} \rightarrow \mathbb{R}$ is *self-concordant* if $|f'''(x)| \leq 2f''(x)^{3/2}$ for all $x \in \text{dom } f$. It can be shown that the Euclidian 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

$i = 1, 2, \dots, n$ and $j = 1, 2, \dots, m$, the feasible set is $\mathcal{S} = \cap_{i=1}^n \mathbf{dom} f_i \cap \cap_{j=1}^m \mathbf{dom} h_j$. 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 satisfied⁶. 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 these, we move on to our topic of complexity analysis. In section 2.1 we have discussed that the problem as presented in equation (2) is not convex due to the non-linear equality constraint guaranteeing the length of a link is constant at all times. We get around 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 problem 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), $Dg(\mathbf{x}^*)$ is the Jacobian of the objective function.

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

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

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

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-quadratic (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 a interior point method to solve the nonlinear programming problem

⁶We resort to a slight misuse of notation for brevity. For a more complete treatment, one may refer to Chapter 5 of the book by Boyd and Vandenberghe [Boyd book].

at hand and by reformulating equation (28) as a barrier problem with barrier parameter μ . The KKT necessary and sufficient conditions for the given problem is given in equation (29).

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

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 methods, the computation times depend on two principal factors— the number of Newton steps required to converge and the complexity of each Newton step i.e. 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 a integer. In case of our problems, we actually observed this phenomenon where, for a 2D case the total number of iterations were about 10, for $m = 6$ for 4 linear constraints defining \mathcal{S} and 2 trust region constraints. **TELL something about 3d here.**

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

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

where, $\mathbf{A} = \nabla_{\mathbf{x}_t}^2 g(\mathbf{x}_t) + \sum_{i=1}^m \nabla_{\mathbf{x}_t}^2 \lambda_i f_i(\mathbf{x}_t)$ 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 [put nilanjan's ieee tro paper] we can obtain the Newton directions as:

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

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 inversion can be calculated in linear time ($\mathcal{O}(m)$). Overall, the

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.

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 quite computationally involving. We believe, real time implementation of these problems, even with near analytical formulations of the in-out function will be very difficult.

It is easily seen that a given hyper-redundant robot with a particular link length will not be able to negotiate a path with a “very low” curvature. This is shown in figure 17a. In figure 17a, the points 1,2,...,8 denote the coordinates of \mathbf{x}_h across successive iterations. It is seen, that 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 (2)), backtracks its own path. Soon after, the optimization problem stops as the link seems to be “locked” at the trough of $\zeta(u)$. In this section we want to quantify the locking effect. To this end, we borrow the concept of “traversability” from literature on wheeled mobile robots (see e.g. Nilanjan’s JMD paper). 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 matrix⁷ 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

⁷The relative curvature matrix of a parametric surface $S(u, v) : \mathbb{S}^2 \rightarrow \mathbb{R}^3$ and a sphere of radius R at the point of contact $P = S(u^*, v^*)$ and the Gaussian frame attached to the sphere at P is given as:

$$\kappa_R|_P = \begin{bmatrix} S_u \cdot S_u - 1/R & S_u \cdot S_v \\ S_u \cdot S_v & S_v \cdot S_v - 1/R \end{bmatrix}_{u^*, v^*}$$

in figure 17c. This would also give us an idea about the largest link length that can be used in a hyper-redundant robot traversing a given duct. This concept also holds for tessellated surfaces (e.g. see the work by Meyer et al. [??]).

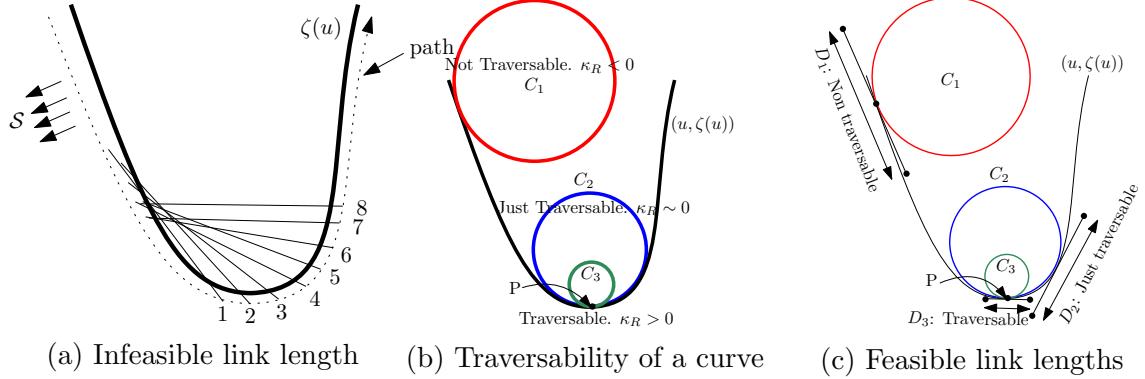


Figure 17: Feasibility of motion planning with a given link length about a given curve.

6 Conclusions

??

6.1 Contributions and scope of future work

uses LAPACK and heavy solvers.. not really required. instead a stripped version of the interior point will do quite well for fast near real-time implementation.

7 Appendix

7.1 Analytical expressions for u, v for quadrilateral patch

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

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

$$a_0 = y - {}^y P_{i-1}, \quad a_1 = {}^y P_{i-1} - {}^y P_i, \quad a_2 = {}^x Q_{i-1} - {}^x P_{i-1}, \quad a_3 = ({}^x Q_i - {}^x P_i) - ({}^x Q_{i-1} - {}^x P_{i-1})$$

$$b_0 = x - {}^x P_{i-1}, \quad b_1 = {}^x P_{i-1} - {}^x P_i, \quad b_2 = {}^y Q_{i-1} - {}^y P_{i-1}, \quad b_3 = ({}^y Q_i - {}^y P_i) - ({}^y Q_{i-1} - {}^y P_{i-1})$$

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

$$u = \frac{b_0}{a_2}, \quad v = \frac{a_0 a_2 - (a_3 + a_2)}{a_1(b_0 - a_2) + b_0(b_3 - b_2)} \quad (33)$$

7.2 Parametric equation of solid cylinder

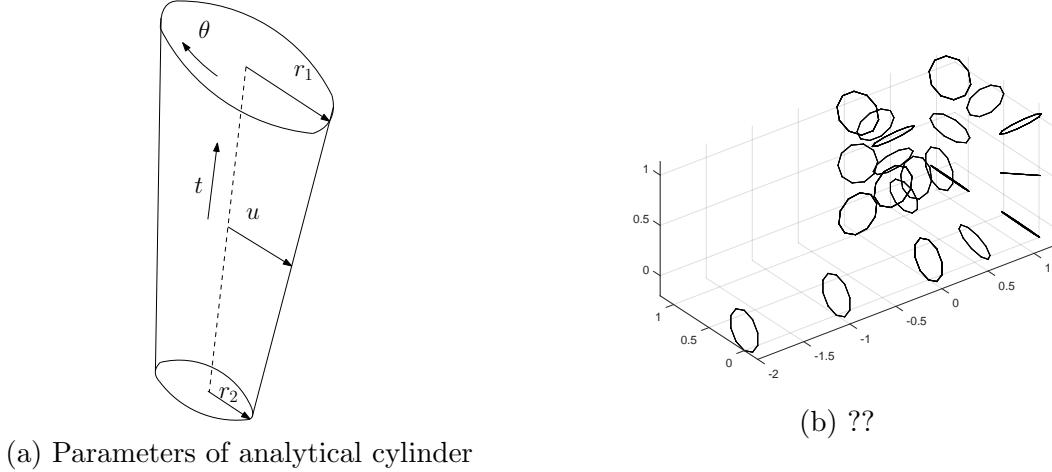


Figure 18: ??(??Figs will be modified??)

$$x = C_1(u, t, \theta) = (r_1 m_1 u \cos \theta + m_2) (1 - t) + (r_2 n_1 u \cos \theta + n_2) t \quad (34)$$

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

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

where

$$\begin{aligned} m_1 &= \cos^1 \phi_2, \quad m_2 = {}^1 x_c, \quad m_3 = \sin^1 \phi_1 \sin^1 \phi_2, \quad m_4 = \cos^1 \phi_1, \\ m_5 &= {}^1 y_c, \quad m_6 = -\cos^1 \phi_1 \sin^1 \phi_2, \quad m_7 = \sin^1 \phi_1, \quad m_8 = {}^1 z_c \end{aligned}$$

and

$$\begin{aligned} n_1 &= \cos^2 \phi_2, \quad n_2 = {}^2 x_c, \quad n_3 = \sin^2 \phi_1 \sin^2 \phi_2, \quad n_4 = \cos^2 \phi_1, \\ n_5 &= {}^2 y_c, \quad n_6 = -\cos^2 \phi_1 \sin^2 \phi_2, \quad n_7 = \sin^2 \phi_1, \quad n_8 = {}^2 z_c \end{aligned}$$

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.

7.3 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 1, we describe a method, to classify a given set of points O with respect to \mathcal{P} as inside (O_{in}) and outside (O_{out}). Our algorithm and it's Matlab implementation⁸ is not very efficient at higher dimensions but seems to perform reasonably well at 3D, by taking less than 8 seconds to classify about 10,000 points with respect to a meshed domain.

Purpose : To classify a set of points O with respect to $\partial\mathcal{P}$

Input: The set O , $O \in \mathbb{R}^3$ 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 the boundary $\partial\mathcal{SP}$
- 2: **for** $i = 1, 2, \dots, N$ **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: $H(i) = \langle O_i - A_i, A_n^i \rangle$
- 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 1: Algorithm for classifying points as inside (O_{in}) or outside (O_{out}) of a triangulated domain

References

We have to see IEEE reference format.

⁸We have used a Matlab function `inhull.m`, a fully vectorized implementation of the algorithm, made available by John D'Errico for free usage.