Collision-free kinematics for hyper-redundant manipulators in dynamic scenes using optimal velocity obstacles

Liangliang Zhao, Zainan Jiang, Yongjun Sun, Jingdong Zhao and Hong Liu

Abstract
Hyper-redundant manipulators have been widely used in the complex and cluttered environment for achieving various kinds of tasks. In this article, we present two contributions. First, we provide a novel algorithm of relating forward and backward reaching inverse kinematic algorithm to velocity obstacles. Our optimization-based algorithm simultaneously handles the task space constraints, the joint limit constraints, and the collision-free constraints for hyper-redundant manipulators based on the generalized framework. Second, we present an extension of our inverse kinematic algorithm to collision avoidance for the hyper-redundant manipulators, where the workspaces may have different types of obstacles. We highlight the performance of our algorithm on hyper-redundant manipulators with various degrees of freedom. The results show that our algorithm has made full use of dexterity of hyper-redundant manipulators in complex environments, enhancing the performance and increasing the flexibility.

Keywords
Inverse kinematics, hyper-redundant manipulators, collision avoidance, dynamic scenario

Introduction
Hyper-redundant manipulators consist of more degrees of freedom (DOF) than those of traditional manipulators. Due to the richness in DOF, hyper-redundant manipulators have been used in many complex and cluttered environments to achieve the tasks that traditional manipulators cannot do, such as medical applications, search, rescue, and reactor repair. Moreover, many applications use hyper-redundant manipulators that work in an environment with static and/or moving obstacles. Therefore, collision-free inverse kinematics is a fundamental problem in the application of hyper-redundant manipulators. The problem can be defined with respect to the sets of constraints, including task space constraints, joint limit constraints, and collision-free constraints. During each control sampling period, the manipulator must compute an inverse kinematic solution based on the desired pose of its end effector and local observations of the workspace, such that it stays free of collisions with the static and/or moving obstacles.
Many kinds of research have been focused on effective methods to resolve the inverse kinematics and collision avoidance problem in robotics. The popular way is to use Jacobian inverse methods, where the collision-free constraints can be regarded as a subtask.\textsuperscript{5–8} Those methods map the collision-free joint space vector into task space velocities based on the projection operator $\textbf{I} - \textbf{J}^\top \textbf{J}$ (i.e., the null space of the transformation). However, the high computational cost of the $\textbf{J}^\top \textbf{J}$ computation makes the Jacobian inverse methods unsuitable for the manipulators with a large number of DOF.\textsuperscript{9,10}

This limit leads to the development of shape trajectory control approaches, which use differential geometry to formulate the closed-form modal inverse kinematic solutions and adjust the manipulator configurations to avoid the obstacles.\textsuperscript{11–15} Those approaches to solve the inverse kinematic problem are only for limited kinematic models, which constrain the performance (dexterity in cluttered environments) of the hyper-redundant manipulators and restrict their potential applicability.

Therefore, some research in this field have been focused on using the data-driven methods, such as learned methods and deep learning methods that use prelearned configurations to match the end-effector’s position to generate a feasible posture from the databases.\textsuperscript{16–23} A major limitation of those methods is that the robots need large datasets to obtain more accurate inverse kinematic solutions. However, the training data are difficult to generate when the robots work in cluttered environments. Furthermore, the discontinuity in configurations is considered as one of the most important disadvantages of data-driven inverse kinematic solvers.

Moreover, there are many studies on collision-avoidance problem for motion planning of robots in dynamic workspaces.\textsuperscript{24–27} The limitations of these studies though are that most of the approaches assume the robots to simple circle models and only take two-dimensional (2D) workspace into account, which limits their applicability to hyper-redundant manipulators.

In this article, we address those shortcomings by introducing a new collision-free inverse kinematic algorithm for a hyper-redundant manipulator, which operates in a dynamic environment with static and/or dynamic obstacles. First, we provide a novel combination of forward and backward reaching inverse kinematics (FABRIK)\textsuperscript{28,29} and velocity obstacles.\textsuperscript{30} The basic idea is that we compute inverse kinematic solutions that satisfy the task space and joint limit constraints while accounting for collision-avoidance constraints. Then, we present an extension of collision-free inverse kinematic algorithm to special cases of workspace for hyper-redundant manipulators\textsuperscript{3} with a different number of joints. Finally, we implement our algorithm in simulation for a variety of hyper-redundant manipulators working with different types of obstacles. The results show that our algorithm is capable of solving the inverse kinematic problem of hyper-redundant manipulators in dynamic working environments.

The main contributions of this article are provided as follows:

i. We provide a novel inverse kinematic algorithm of relating FABRIK to velocity obstacles. We will demonstrate that our algorithm can be applied directly to any general planar and spatial manipulators with high DOF, such as handling the task space constraints, the joint limit constraints, and the collision-free constraints.

ii. We present an extension of our approach to collision avoidance for the hyper-redundant manipulators, where the workspaces may have different types of obstacles. We will show specifically the obstacles such as moving obstacles, pole, and wall.

The rest of the article is organized as follows. In the second section, we briefly discuss related works in inverse kinematics of hyper-redundant manipulators. In the third section, we define the problem of inverse kinematics and extend the notion of velocity obstacles to hyper-redundant manipulators. The fourth section presents our approach for special cases with different types of hyper-redundant manipulators and moving or static obstacles. The fifth section highlights the performance of our method and the sixth section summarizes the conclusion.

**Previous work**

In this section, we give a brief overview of prior work on inverse kinematics of hyper-redundant manipulators.

Some of the popularly used approaches to generate inverse kinematic solutions are Jacobian inverse methods. Maciejewski and Klein\textsuperscript{5} and Nakamura et al.\textsuperscript{6} present an algorithm to handle the end-effector trajectory and obstacle avoidance (OA) constraints by the null space of the transformation. Stilman\textsuperscript{7} presents a global randomized motion planning method for joint space based on tangent-space sampling and first-order retraction. Lee et al.\textsuperscript{8} solve the inverse kinematic problem by the robust task priority (RTP)-closed-loop inverse kinematic algorithm with task-priority strategy.

For shape trajectory control approach, Chirikjian and Burdick\textsuperscript{11,12} introduce a novel method to solve the inverse kinematic problem of hyper-redundant manipulators based on a “backbone curve.” Mochiyama and Kobayashi\textsuperscript{13} adopt Lyapunov design to estimate the desired curve parameter, where a time-varying curve is given to track the manipulator shape. Fahimi et al.\textsuperscript{14} present a mode shape technique-based approach to resolve the OA problem for hyper-redundant manipulators. Mu et al.\textsuperscript{15} propose a new strategy named hybrid OA method, which can be used in OA motion planning for hyper-redundant manipulators.
For the intelligence algorithms, Rose et al.\textsuperscript{16} introduce an inverse kinematic algorithm based on the radial basis function (RBF) interpolation. Rolf et al.\textsuperscript{17} introduce a method to learn inverse kinematics of redundant systems based on a path-based sampling approach and without prior or expert knowledge. Toshani and Farrokhi\textsuperscript{18} apply RBF neural networks in redundant manipulators to obtain the real-time joint values by the Cartesian coordinate of the end effector. Collins and Shen\textsuperscript{19} propose an integrated approach, named path planning with swarm optimization, to solve the inverse kinematics and path planning problems for hyper-redundant manipulators in three-dimensional (3D) workspaces. Peng et al.\textsuperscript{20} introduce a hierarchical learning-based method to control the motion style of a 3D bipedal. Starke et al.\textsuperscript{21} use a hybrid evolutionary approach to solve inverse kinematics for highly articulated and humanoid robots. Huang et al.\textsuperscript{22} propose a parallelizable constrained inverse kinematic technique by modeling and using dynamic joint motion parameters, which can be automatically learned from input mocap data. Xidias\textsuperscript{23} presents a method for time optimal trajectory planning for a hyper-redundant manipulator, which transforms the trajectory planning problem to a global optimization problem and solve it by a genetic algorithm.

There have been some developments for collision avoidance for redundant manipulators in dynamic environments. Duindam et al.\textsuperscript{31} use the inverse kinematic solutions to compute feasible paths for steerable needles to avoid obstacles. Park et al.\textsuperscript{32} present a novel method to compute a collision-free trajectory for the robot by an incremental trajectory optimization framework. Tao and Yang\textsuperscript{33} propose a method for motion planning of a redundant manipulator based on the OA-FABRIK algorithm. Menon et al.\textsuperscript{34} introduce an optimization algorithm that all links of a hyper-redundant manipulator avoid the obstacles using calculus of variation. Other recent optimization methods studied are based on curve-constrained collision-free trajectory control,\textsuperscript{35} vector field inequalities,\textsuperscript{36} and Newton’s method.\textsuperscript{37}

\section*{Materials and methods}
In this section, we present our algorithm for collision avoidance, which extends velocity obstacles to hyper-redundant manipulators based on FABRIK. We derive the formulation for calculating the velocity obstacle constraints while they are induced by other moving obstacles in a time window. We also present an efficient technique to compute the possible safe position for each joint of a hyper-redundant manipulator that simultaneously satisfies collision-free constraints, task space constraints, and joint limit constraints.

\section*{Preliminaries}
Hyper-redundant manipulators have many more DOF than the number of a particular task required. As the number of DOF increases, the widely used methods for solving the inverse kinematic problems, such as analytical methods\textsuperscript{38} and generic kinematics and dynamic library algorithm,\textsuperscript{39} are considered unsuitable for hyper-redundant manipulators. The position and orientation of the end effector cannot be fully defined by link lengths and joint angles because there exists an infinite number of inverse kinematic solutions. In addition, more restrictions will impose on inverse kinematic solutions when the manipulator encounters static and moving obstacles. The challenge of inverse kinematics for hyper-redundant manipulators is that the algorithm must compute changes in joint values to avoid collision.

The functional form of the inverse kinematic problem of the hyper-redundant manipulators is given by

\begin{equation}
\theta_{1,...,n} = f^{-1}(\xi_\varepsilon)
\end{equation}

where $\xi_\varepsilon$ is the desired pose of the end effector and $\theta_{1,...,n}$ are required joint coordinates. Because of redundancy, it is necessary to consider a numerical method that computes a feasible solution. The inverse kinematic solver FABRIK is a heuristic method that combines forward and backward iterative methods, as given by Aristidou and Lasenby.\textsuperscript{28} Instead of using joint vector $\theta$, FABRIK performs a joint position search using these methods and terminates when either of these methods converges to an inverse kinematic solution.

In Figure 1(a), the manipulator is a concatenation of four identical joints, where the first joint is fixed at its base and the last joint is its end effector. Let $p_i$ ($i = 1,\ldots,4$) represent currently the joint position of the manipulator and let $p_1$ be a target position. We assume that the distances between each joint $d_i$ can be derived from the joint position $p_i$ by the function: $d_i = |p_{i+1} - p_i|$.

In the forward search, the end effector position of the manipulator $p_4$ is moved to the target position $p_1$, as shown in Figure 1(b). There exists a line, which passes through the

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure1.png}
\caption{An example of a forward search and a backward search of FABRIK for a manipulator with four joints. (a) The positions of the manipulator and the target at the beginning of the movement. (b) Move the end effector $p_4$ to the target, and then find the joint positions $p'_3$, $p'_2$, and $p'_1$. (c) Move the base joint $p'_1$ to its initial position, and then find the joint positions $p''_3$, $p''_2$, and $p''_4$.}
\end{figure}
Figure 2. In the XYZ-dimensional, an ellipsoid and a sphere move toward one another with velocity \( v_A \) and \( v_B \), respectively.

Relating FABRIK algorithm to velocity obstacles

Given the method above, we would like to extend the notion of velocity obstacles to OA based on FABRIK. As shown in Figure 2, the ellipsoid \( A \) is represented by a center point \( p_1 \), an orthonormal set of axis-direction vectors \( \{U_0, U_1, U_2\} \), and associated extents \( c_i \) with \( e_0 \geq e_1 \geq e_2 \). The sphere \( B \) is centered at \( p_B \) with radius \( r_B \). For ellipsoid \( A \) and sphere \( B \) in the Cartesian space, the Minkowski sums \( A \oplus B \) can be defined as

\[
A \oplus B = \{ a + b | a \in A, b \in B \}
\]

(2)

Let \( \lambda^\tau(p, v) = \{ p + tv | t \in [0, \tau] \} \) denotes a ray for time window \( \tau \), which starts at position \( p \) with direction \( v \). As can be seen in Figure 3, the velocity obstacle \( VO^\tau_{A|B} \) is given by

\[
VO^\tau_{A|B} = \{ v | \lambda^\tau(p_A, v - v_A) \cap -A \oplus B \neq \emptyset \}
\]

(3)

where \( v_A \) is the velocity vector of the ellipsoid \( A \).

As shown in Figure 4, if the sphere \( B \) selects a velocity \( v_B \), we observe that sphere \( B \) may collide with moving ellipsoid \( A \) during the time interval \([0, \tau]\) if the relative velocity vector \( v_A - v_B \) is inside the region \( VO^\tau_{A|B} \). The problem of collision avoidance is now defined as the relative velocity vector that must be outside \( VO^\tau_{A|B} \) before time \( \tau \). Having identified a velocity \( v_B \) for sphere \( B \), the vector \( \omega \) from \( v_A - v_B \) to the closest point of the \( VO^\tau_{A|B} \)'s boundary, as shown in Figure 4, is given as

\[
\omega = \left\{ \begin{array}{l}
\arg\min_{v \in \partial VO^\tau_{A|B}} || v - (v_A - v_B) ||_2 - (v_A - v_B) \\
\end{array} \right.
\]

(4)

Given the defined generalized method for collision avoidance using velocity obstacles, we present the
extension of this method for collision-free inverse kinematics for hyper-redundant manipulators. In FABRIK algorithm, we define a task error

\[
e = ||p_n - p_i||
\]  

(5)

To compute the joint position \( p_t \) for given \( p_i \), without any collisions, we would like to solve the optimization problem with equality constraints as follows

\[
\begin{align*}
\text{argmin} & \quad e \\
\text{s.t.} & \quad v_t - v_O \in \partial VO_{I/O}^t \\
& \quad ||p_{i+1} - p_i|| \in D_i \\
& \quad \theta_i \in JL_i
\end{align*}
\]  

(6)

where \( v_t \) is the velocity of link \( I (I = 1, \ldots, n - 1) \), \( v_O \) is the velocity of obstacle \( O \), \( D_i \) is the distance constraint between adjacent joints \( (i = 1, \ldots, n) \), and \( JL_i \) is the joint limit constraint.

An example for solving equation (6) is shown in Figure 5. We consider a hyper-redundant manipulator and a moving obstacle sharing a common working environment, where the obstacle moves with a known linear trajectory. The start positions of link \( I \) and moving obstacle are shown in Figure 5(a). Let \( p_i \) be the position of joint \( i \). Let us now define \( d_i = ||p_i - p_{i-1}|| \) as the distance between joint \( i \) and joint \( i - 1 \). The moving obstacle \( O \) is centered at \( p_O \) with velocity \( v_O \). To apply velocity obstacles in the process of inverse kinematics, each movable link of the hyper-redundant manipulator is described by an ellipsoid. These ellipsoids move around other dynamic obstacles. Each dynamic obstacle is assumed to be a sphere or is bounded by a sphere. The position \( p_t \) of link (ellipsoid) \( I \) is

\[
p_t = p_{i} + \frac{p_{i+1} - p_i}{2}
\]  

(7)

where \( i = I + 1 \). The extents of ellipsoid \( I \) are defined as \( e^0_i = d_i/2 \), and \( e^1_i \) and \( e^2_i \) are dynamically determined by the size of the links to make sure that there are no collisions.

Figure 5(a) also shows the first step of the forward search in the FABRIK algorithm. Joint \( i \) updates its position to \( p'_i \) based on its initial position \( p_i \) and the previous joint position \( p'_{i+1} \). Then, we assume that the velocity \( v_{p_i} \) of joint \( i \) is determined only by its new position \( p'_i \) and the initial position \( p_i \),

\[
v_{p_i} = \frac{p'_i - p_i}{\Delta t}
\]  

(8)

where \( \Delta t \) is the time duration of the end effector moving between the two positions \( p_n \) and \( p_i \) at one movement step. The velocity \( v_t \) can be derived from the velocity of joint \( i \) and joint \( i - 1 \) by

\[
v_t = \frac{v_{p_b} + v_{p_{b-1}}}{2}
\]  

(9)

Then, the velocity obstacles \( VO_{I/O} \) induced by obstacle \( O \) are defined, as shown in Figure 5(b). We can find that the velocities \( v_t \) and \( v_O \) will lead link \( I \) and obstacle \( O \) to a collision course within \( \tau \) time, that is, \( v_t - v_O \in VO_{I/O} \). Let vector \( \omega \) be the closest point to \( v_t - v_O \) on the boundary of \( VO_{I/O} \), and let vector \( u \) be the outward normal of \( VO_{I/O} \) at the point \( v_t - v_O + \omega \). We can then compute set \( CA_{I/O} \) of possible safe velocity for link \( I \) before time \( \tau \)

\[
CA_{I/O} = \{ v | (v - (v_t + \omega)) \cdot u \geq 0 \}
\]  

(10)
Since for link velocity, it holds that \( v_I = (v_p + v_{p-1})/2 \), the set \( CA^I_{j|O} \) can be transformed to determine sets of permitted velocities for both joint \( i \) and joint \( i-1 \) as

\[
CA^I_{j|O} = \{ v | (v - (v_p + \omega)) \cdot u \geq 0 \} \tag{11}
\]

and

\[
CA^I_{i-1|O} = \{ v | (v - (v_{p_i-1} + \omega)) \cdot u \geq 0 \} \tag{12}
\]

where the two joints will share the responsibility for collision avoidance between the link \( I \) and the moving obstacle \( O \) equally. As shown in Figure 5(c), \( CA^I_{j|O} \) is the half-space pointing in the direction of \( u \) containing the point \( v_p + \omega \).

To avoid a collision, joint \( i \) needs to select a new velocity \( v_{p_i} \) that lies inside the region of permitted velocities \( CA^I_{i|O} \) and closes to its velocity \( v_p \). While adhering to a distance constraint and a joint limit constraint

\[
v_{p_i}^{new} = \arg\min_{v \in CA^I_{i|O} \cap D_i \cap JL_i} \| v - v_p \|, \quad v \in CA^I_{i|O} \cap D_i \cap JL_i \tag{13}
\]

Finally, joint \( i \) updates its position

\[
p_I' = v_{p_i}^{new} + \Delta t \tag{14}
\]

and all other joints use the same approach to avoid collisions with the obstacles. For the backward search, apply the same method to the joints, and it would compute the set of feasible changes in velocity and update joint positions to avoid collisions with moving obstacles.

The detailed procedures of our approach for finding the inverse kinematic solutions based on FABRIK and velocity obstacles can be given in Algorithm 1. Moreover, Algorithm 1 terminates after \( \varepsilon \leq \varepsilon \) or a maximum computation time \( t_{\text{max}} \), where \( \varepsilon \) is a stopping criterion. The set \( CA^I_{j|O} \) can be computed using equation (10) within function \textsc{Safe Velocity Set()}, see lines 13 and 25. We can compute a collision-free position of joint \( i \) using equation (14) within function \textsc{Collision Free Set()}, see lines 15 and 27. The joint limits can be applied within function \textsc{Joint Limit Set()}, see lines 10 and 22 (see Aristidou et al.\textsuperscript{28,29} for work specifically including the joint limits in FABRIK algorithm).

If \( CA^I_{j|O} \cap D_i \cap JL_i = \emptyset \) (i.e. the position of joint \( i \) is not able to find when adhering to these constraints), a smaller value of time window \( \tau \) will be selected to increase the region of the set \( CA^I_{j|O} \). This approach will make the link \( I \) move toward the obstacle which is easy to avoid and then adhering to the constraints, which are difficult to handle. However, it does not hold for the situation that the position of joint \( i \) must be changed more to avoid a collision, as shown in Figure 6. In this case, we can rewrite equation (13) as

\[
v_{p_i}^{new} = v_p + \omega \tag{15}
\]

that not only including distance constraint and/or joint limit constraint of joint \( i \) but also of the collision-free constraint in this search (forward or backward). Joint \( i \) updates its position using equation (14). The next link (i.e. link \( I+1 \) for forward search or \( I-1 \) for backward search) can then choose the change in position with respect to this updated position of link \( I \). In other words, link \( I+1 \) or link \( I-1 \) will share the responsibility for avoiding collisions of link \( I \). Note that this approach will imply more iterations to reach the stopping criteria.

The proposed algorithm has the consequence that if the moving obstacle potentially results in a collision with the
hyper-redundant manipulator, then the changes in joint values will be computed to avoid this collision. This algorithm can be applied to general planar and spatial hyper-redundant manipulators with a single end effector for real-time control. It is able to compute collision-free inverse kinematic solutions that satisfy the complex dynamic constraints in the workspace and kinematic constraints (i.e. task space and joint limit constraints).

**Implementation on special cases**

We have introduced the above algorithm for collision avoidance for a hyper-redundant manipulator in an uncertain dynamic environment. We did so through a new algorithm of relating FABRIK algorithm to velocity obstacles. As shown in Algorithm 1, we can find a collision-free inverse kinematic solution for a hyper-redundant manipulator if given the position and velocity of the obstacle. We will now show how the previous algorithm of collision avoidance can be implemented in different working environments.

**Avoiding collisions with multiple dynamic obstacles**

Algorithm 1 is presented for collision avoidance with one moving obstacle that can easily be extended to more than two obstacles. Let $O$ be a set of moving obstacles sharing a common workspace with a hyper-redundant manipulator. The position $p_O$, the velocity $v_O$, and the radius $r_O$ for each obstacle $O \in O$ are a priori known, as shown in Figure 7.

Then, the velocity obstacles of the link $I$ with respect to every obstacle can be defined using equation (3), as shown in Figure 8(a). The function `SAFE_VELOCITY_SET()` in Algorithm 1 can now be defined as having the link computes the set of feasible changes in velocity based on the velocity

**Figure 6.** The new velocity $v_{p_i}^{\text{new}}$ as in Figure 5 for different obstacle velocity $v_O$.

**Figure 7.** A case where a hyper-redundant manipulator and two moving obstacles $O_1$ and $O_2$ in the workspace.

**Figure 8.** (a) The velocity obstacles and (b) the set $CA_I$ for two moving obstacles in Figure 7.
obstacles $VO^{T}_{I\mid O_1}$ and $VO^{T}_{I\mid O_2}$, as can be seen in Figure 8(b), which gives

$$CA^T_I = \bigcap_{O \in O} CA^T_{I\mid O} \quad (16)$$

The set of safe changes in velocity for joint $i$ can then be determined by $CA^T_I$. Similar to equation (13), the new joint velocity that is safe from all moving collisions is given as

$$v^{new}_{p_i} = \text{argmin}_{v \in CA^T_{I\mid O}} \|v - v_{p_i}\|, \quad v \in \bigcap_{O \in O} CA^T_{I\mid O} \cap D_i \cap J_L.$$  

(17)

Since the only modification to Algorithm 1 is in the $\text{COLLISION\_FREE\_SET()}$ function, the set $CA^T_{I\mid O}$ is extended to the set $\bigcap_{O \in O} CA^T_{I\mid O}$, which includes other moving obstacles. Note that the problem of collision avoidance between the moving obstacles is not considered in this algorithm.

**Avoiding collisions with static obstacles**

Algorithm 1 gives an inverse kinematic algorithm for the hyper-redundant manipulators and the moving obstacles to avoid collisions. However, the typical robot workspace also contains static obstacles, such as wall and pole. We will apply the same algorithm as above to this situation, it would compute the set $CA^T_I$ for the link $I$ to avoid collisions with static obstacles.

We assume that the geometric shape of the static obstacle is a regularly shaped body. More specifically, we would like to apply the basic space geometric shapes of cuboids and cylinders to model the static obstacles of walls and poles, respectively. This is shown in Figure 9. We can assume that the wall obstacle is modeled as a cuboid, and its geometric shape is determined by the position, orientation, and dimension of the wall obstacle. Let $O$ be the cuboid, and let link $I$ be an ellipsoid centered at $p_I$. Then, the velocity obstacles $VO^T_{I\mid O}$ is defined by equation (3), as

Figure 9. A case where a hyper-redundant manipulator and a static obstacle $O$ in the workspace.

Figure 10. (a) The velocity obstacles $VO^T_{I\mid O}$ and (b) the set $CA^T_I$ for a static obstacle in Figure 9.

Figure 11. The obstacle avoidance inverse kinematic scheme for hyper-redundant manipulators.
shown in Figure 10(a) and (b). If the velocity $v_I$ is inside $VO^t_{O,I}$, link $I$ will collide with the cuboid $O$ within $\tau$ time. To avoid a collision, joint $i$ needs to select a new velocity $v_{new}^i$ that satisfies the conditions of equation (13).

**Implementation and performance**

In this section, we highlight the performance of our algorithm in simulated environments that contain a hyper-redundant manipulator and many moving or static obstacles in the 2D and 3D workspaces. We present the kinematic parameters for hyper-redundant manipulators used in the simulations and the results from those simulations.

**Implementation details**

The simulations are developed in a C++ environment. The code runs on a single notebook running Ubuntu 16.04 64-bit with CPU i7-7700HQ at 2.81 GHz and 8GB RAM. To verify the adaptability of our approach, we run simulations that include the hyper-redundant manipulators with a different number of joints. The main structure of OA inverse kinematic scheme for hyper-redundant manipulators is shown in Figure 11. The simulations are initialized by decomposing the manipulator model into a series of ellipsoids (ellipses for planar manipulators). For simplicity, when the distance between the link and the obstacle is sufficiently large, we do not take this link into account while computing the collision-free constraint. The task space constraints are used to compute positioning error, and the velocity obstacles are used to define the safe velocity sets for each joint of the manipulator. A collision-free joint position $p_{new}^i$ is computed by the iterations in Algorithm 1. The loop terminates as soon as all constraints are satisfied.

Unless otherwise noted, we set a value of $\tau = 2$ s in these simulations. The stopping criteria $\varepsilon$ is $10^{-7}$% of the

![Figure 12. Setup for simulation A.](image-url)

![Figure 13. (a–d) Shape trajectory control approach.](image-url)
Figure 14. (a–d) Natural-CCD algorithm.

Figure 15. (a–d) Our collision-free inverse kinematic algorithm.
manipulator length and the timeout $t_{\text{max}} = 5$ ms. The frequency that is used to control a manipulator is 100 Hz (corresponding to a control sampling period of 10 ms, i.e. $\Delta t = 10$ ms). Moreover, all joints are within their specified limits.

**Simulation results**

*Simulation A: Planar manipulator with a static obstacle.* To illustrate our algorithm, we have provided a variety of simulations. In simulation A, the algorithm applied to a 7-DOF planar manipulator is compared with the shape trajectory control approach based on backbone curve\(^\text{12}\) and the natural-cyclic coordinate descent (CCD) algorithm.\(^\text{40}\) The simulation setup is shown in Figure 12. A 7-DOF planar manipulator (each joint has one DOF) and a circle-shaped obstacle are involved in this simulation. The black lines are the desired path of the end effector (i.e. joint position $p_e$), which moves at a constant velocity norm. The

![Figure 16. Angle variation for each joint for simulation A.](image)

![Figure 17. Setup for simulation B.](image)

![Figure 18. (a–d) A simulation that contains a dynamic obstacle, which is a red sphere with velocity $v_O = [0, -0.08, 0]$ (m/s).](image)
(a–d) A simulation that contains four dynamic obstacles. In the first screenshot, the velocities of two red spheres on the left are \( v_{O1} = \frac{1}{2} C0_{0.065} \), and the velocities of two red spheres on the right are \( v_{O3} = \frac{1}{2} C0_{0.065} \).

Figure 19. (a–d) A simulation that contains two dynamic obstacles with varying velocities. In the first screenshot, the velocity of left red sphere is \( v_{O1} = [0.06, 0, 0] \) (m/s), and the velocity of right red sphere is \( v_{O2} = [0, -0.08, 0] \) (m/s).

Figure 20. (a–d) A simulation that contains four dynamic obstacles. In the first screenshot, the velocities of two red spheres on the left are \( v_{O1} = v_{O2} = [0.065, 0, 0] \) (m/s), and the velocities of two red spheres on the right are \( v_{O3} = v_{O4} = [-0.065, 0, 0] \) (m/s).
The motion command for the manipulator is generated based on the shape trajectory control approach, natural-CCD algorithm, and our OA-FABRIK algorithm, respectively. In addition, the end effector must maintain the task constraint during the execution.

The link length of the manipulator is 
\[ d = [d_1, \ldots, d_7] = [10, 9, 8, 7, 6, 5, 4] \text{ (mm)} \]

The associated extents of ellipses are 
\[ e_{i,0} = d_i/2 \quad \text{and} \quad e_{i,1} = d_i/6 \]

The joint limit 
\[ J_{Li} = ([0, 3], [0, 3]) \text{ (rad)} \]

The desired velocity norm of the end effector during the simulations is 
\[ ||v_E|| = 5.7 \]

The circle-shaped obstacle with \( r_O = 2 \text{ mm} \) and \( p_O = [15, 15] \text{ (mm)} \) in the workspace.

The selected screenshots of simulation A are shown in Figures 13 to 15. As can be seen, when the manipulators use the shape trajectory control approach (i.e. Figure 13) and the natural-CCD algorithm (i.e. Figure 14), it would lead to a collision with the obstacle during the operation. The configurations in Figure 15, with manipulator solving the inverse kinematics using our algorithm (i.e. OA-FABRIK), show no collisions over its entire operation. The joint trajectory computed by our algorithm in the manipulation phase in Figure 16 shows no sudden activation and no oscillation over its entire length.

**Simulation B: Spatial manipulator with dynamics obstacles.** One set of simulations consists of a spatial hyper-redundant manipulator and groups of moving obstacles, where the
manipulator and the obstacles were simulated separately. The setup of simulation B is shown in Figure 17. The workspace includes a 30-DOF manipulator with 10 joints (each joint has three DOF) and two moving obstacles, in which the velocity and position can be obtained. In this simulation, the manipulator tries to keep its end effector \( \mathbf{p}_{11} = [0, 0, 1.3] \text{ m} \) in a fixed position and the moving obstacle may collide with the manipulator if the joints remain in the initial positions, as shown in Figure 17.

The parameters for the manipulator and the obstacles in this simulation are chosen as follows: the link length of the manipulator is \( d_i = 0.2 \text{ m} \). The associated extents of ellipsoid are \( e_{0,i} = 0.1 \text{ m} \), \( e_{1,i} = 0.075 \text{ m} \), and \( e_{2,i} = 0.075 \text{ m} \). The joint limit \( J_{L_i} = (-3.14, +3.14) \text{ (rad)} \). The sphereshaped obstacle with \( r_{O_j} = 0.1 \text{ mm} \) in the workspace.

Based on the simulation setup, we performed three simulations, where one, two, and four moving obstacles cross the workspace with different velocity and direction, respectively. The manipulator can update its control input by the positions and velocities of those moving obstacles to avoid collisions. Selected screenshots of those simulations are shown in Figures 18 to 20.

**Simulation C: Spatial manipulator with static obstacles.** In simulation C, we test the performance of our algorithm on spatial hyper-redundant manipulator with the static obstacles. In this simulation setup, the typical static obstacles are used

---

**Figure 23.** (a–d) A simulation in the scenario with a cylinder obstacle.

**Figure 24.** (a, b) The computational time of the inverse kinematic problem resolution is made for several number of simulations shown above.
in the workspace. As shown in Figure 21(a) and (b), two challenging scenarios are presented, respectively, that is, scenario 1: a manipulator with 10 joints (each joint has three DOF) and a wall and a hole; and scenario 2: a manipulator with 19 joints (each joint has three DOF) and a cylindrical obstacle. In the simulations, a path planner for the end effector is needed, which has been given by the user. The joint motion command for the manipulator is generated based on our collision-free inverse kinematic algorithm.

These two manipulators have the same kinematic parameters, which have been presented in simulation B. In addition, the desired velocity norm of the end effector during the simulations is $\|r_E\| = 0.25$. Figures 22 and 23 show simulation results of these two scenarios, respectively. As can be seen, the end effector can maintain the task constraint, and there are no collisions during task execution.

**Performance analysis**

**Computational time.** To quantify the computational time of our algorithm, we calculate the computation time of the inverse kinematic problem resolution, that is, the computation time it takes for one hyper-redundant manipulator to determine its joint positions of safe changes with respect to the moving or static obstacles in a single control cycle. As shown in Figure 24, the computation time grows with an increase in the number of moving obstacles and joints, as expected. In the simulation, we found that the safe velocity sets are computed without significantly increasing the computational time when the manipulator needs to avoid obstacles. As can be seen in Figure 24(b), the computational time also shows that for a single obstacle, the static obstacle could require a further increase in the computational time (at least in the workspace we implemented).

**Comparison.** To further highlight the performance and efficiency of the proposed algorithm, the comparisons with some existing inverse kinematic approaches are listed in Table 1. It can be observed that the algorithms in the literature do not consider the obstacles in the workspaces. In the literature, although the proposed approaches can avoid collisions, they cannot be applied to the dynamic working environments. Moreover, the algorithms in Rolf et al. and Safeea et al. only apply to 2D workspaces. Note that the proposed algorithm can compute the collision-free inverse kinematic solutions in dynamic workspaces and can handle hyper-redundant manipulators. Furthermore, this algorithm is faster than other inverse kinematic algorithms, does not need any training, and can be applied in real-time control.

**Conclusions**

In this article, we present a novel method toward solving the inverse kinematic problem of hyper-redundant manipulators in complex working environments. To do so, we extend the notion of velocity obstacles to OA based on FABRIK. We then show how to implement the OA inverse kinematic algorithm in special cases. In our simulations, we see that the hyper-redundant manipulators can perform better OA using our algorithm.

The proposed approach has shown the expected performance of the collision-free kinematics for hyper-redundant manipulators in dynamic scenes. Due to the structural characteristic of soft manipulators, a future line of research will be focused on the integration of our algorithm with the soft robotic arm by converting the discrete inverse kinematic solutions to continuous curves. Moreover, we would like to evaluate the performance in dynamic scenes with human or other complex obstacles.

**Declaration of conflicting interests**

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.
1. Chirikjian GS and Burdick JW. Hyper-redundant robot mechanisms and their applications. In: 1991 IEEE international workshop on intelligent robots and systems (ed F Harashima), Osaka, Japan, 3–5 November 1991, pp. 185–190. IEEE.

2. Chirikjian GS and Burdick JW. A hyper-redundant manipulator. IEEE Robot Autom Mag 1994; 1(4): 22–29.

3. Fahimi F. Hyper-redundant manipulators. In: 2009 Autonomous robots: modeling, path planning, and control (ed DS Loyanachan), Boston, MA, USA, 2009, pp. 1–30.

4. Ivanescu M and Cojocaru D. Hyper redundant manipulators. In: Advanced strategies for robot manipulators (ed SE Shafiei), Rijeka: IntechOpen, 2010, pp. 27–60.

5. Maciejewski AA and KleinObstacle CA. Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. Int J Robot Res 1985; 4(3): 109–116.

6. Nakamura Y, Hanafusa H, and Yoshikawa T. Task-priority based redundancy control of robot manipulators. Int J Robot Res 1987; 6(2): 3–15.

7. Stilman M. Global manipulation planning in robot joint space with task constraints. IEEE Trans Robot 2010; 26(3): 576–584.

8. Lee W, Lee Y, Park G, et al. A whole-body rescue motion control with task-priority strategy for a rescue robot. Autonom Robot 2017; 41: 243–258.

9. Pasquale C, Stefano C, Lorenzo S, et al. Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy. Int J Robot Res 1991; 10(4): 410–425.

10. Colombé A and Torras C. Closed-loop inverse kinematics for redundant robots: comparative assessment and two enhancements. IEEE/ASME Trans Mechatron 2015; 20(2): 944–955.

11. Chirikjian GS and Burdick JW. An obstacle avoidance algorithm for hyper-redundant manipulators. In: 1990 IEEE international conference on robotics and automation (ed RA Volz), Cincinnati, OH, USA, 13–18 May 1990, pp. 625–631. IEEE.

12. Chirikjian GS and Burdick JW. A modal approach to hyper-redundant manipulator kinematics. IEEE Trans Robot Autom 1994; 10(3): 343–354.

13. Mochiyama H and Kobayashi H. The shape Jacobian of a manipulator with hyper degrees of freedom. In: 1999 IEEE international conference on robotics and automation, Detroit, Michigan, USA, 10–15 May 1999 (ed H Akeel), pp. 2837–2842. IEEE.

14. Fahimi F, Ashrafiuon H, and Nataraj C. Obstacle avoidance for spatial hyper-redundant manipulators using harmonic potential functions and the mode shape technique. J Robot Syst 2003; 20(1): 23–33.

15. Mu Z, Liu T, Xu W, et al. A hybrid obstacle-avoidance method of spatial hyper-redundant manipulators for servicing in confined space. Robotica 2019; 37(6): 998–1019.

16. Rose III CF, Sloan P-PJ, and Cohen MF. Artist-directed inverse-kinematics using radial basis function interpolation. Comput Graph Forum 2001; 20(3): 239–250.

17. Rolf M, Steil JJ, and Gienger M. Goal babbling permits direct learning of inverse kinematics. IEEE Trans Automon Ment Dev 2010; 2(3): 216–229.

18. Toshani H and Farrokhi M. Real-time inverse kinematics of redundant manipulators using neural networks and quadratic programming: a Lyapunov-based approach. Robot Autonom Syst 2014; 62(6): 766–781.

19. Collins T and Shen WM. PASO: an integrated, scalable PSO-based optimization framework for hyper-redundant manipulator path planning and inverse kinematics. 2016 ISI Tech Report, University of Southern California, Los Angeles, USA, 2016, pp. 1–7.

20. Peng XB, Berseth G, Yin K, et al. DeepLoco: dynamic locomotion skills using hierarchical deep reinforcement learning. ACM Trans Graph 2017; 36(4): 4101–4113.

21. Starke S, Hendrich N, Krupke D, et al. Evolutionary multi-objective inverse kinematics on highly articulated humanoid robots. In: 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS) (ed A Bicchi), Vancouver, BC, Canada, 24–28 September 2017, pp. 6959–6966. IEEE.

22. Huang J, Fratarcangeli M, Ding Y, et al. Inverse kinematics using dynamic joint parameters: inverse kinematics animation synthesis learnt from sub-divided motion micro-segments. Visual Comput 2017; 33(12): 1541–1553.

23. Xidias EK. Time-optimal trajectory planning for hyper-redundant manipulators in 3D workspaces. Robot Comput Integ Manuf 2018; 50: 286–298.

24. Shiller Z, Large F, and Sekhavat S. Motion planning in dynamic environments: obstacles moving along arbitrary trajectories. In: 2001 IEEE international conference on robotics and automation (ed MJ Chung), Seoul, South Korea, 21–26 May 2001, pp. 3716–3721. IEEE.

25. Van Den Berg J, Snape J, Gey SJ, et al. Reciprocal collision avoidance with acceleration velocity obstacles. In: 2011 IEEE international conference on robotics and automation (ed A Ming), Shanghai, China, 9–13 May 2011, pp. 3475–3482. IEEE.

26. Rufli M, Alonso-Mora J, and Siegwart R. Reciprocal collision avoidance with motion continuity constraints. IEEE Trans Robot 2013; 29(4): 899–912.
27. Bareiss D and van den Berg J. Generalized reciprocal collision avoidance. *Int J Robot Res* 2015; 34(12): 1501–1514.
28. Aristidou A and Lasenby J. Fabrik: a fast, iterative solver for the inverse kinematics problem. *Graph Models* 2011; 73(5): 243–260.
29. Aristidou A, Chrysanthou Y, and Lasenby J. Extending FABRIK with model constraints. *Comput Animat Virt World* 2016; 27(1): 35–57.
30. Fiorini P and Shiller Z. Motion planning in dynamic environments using velocity obstacles. *Int J Robot Res* 1998; 17(7): 760–772.
31. Duindam V, Xu JJ, Alterovitz R, et al. Three-dimensional motion planning algorithms for steerable needles using inverse kinematics. *Int J Robot Res* 2010; 29(7): 789–800.
32. Park C, Pan J, and Manocha D. ITOMP: incremental trajectory optimization for real-time replanning in dynamic environments. In: *22nd international conference on automated planning and scheduling (ICAPS)* (ed L McCluskey), Atibaia, Sao Paulo, Brazil, June 2012, pp. 207–215. AAAI Press.
33. Tao SQ and Yang YM. Collision-free motion planning of a virtual arm based on the FABRIK algorithm. *Robotica* 2017; 35(6): 1431–1450.
34. Menon MS, Ravi VC, and Ghosal A. Trajectory planning and obstacle avoidance for hyper-redundant serial robots. *J Mech Robot* 2017; 9(4): 041010 (9 pages).
35. Dalla VK and Pathak PM. Curve-constrained collision-free trajectory control of hyper-redundant planar space robot. *Proc Instit Mech Eng Part I* 2017; 231(4): 282–298.
36. Marinho MM, Adorno BV, Harada K, et al. Dynamic active constraints for surgical robots using vector-field inequalities. *IEEE Trans Robot* 2019; 35(5): 1166–1185.
37. Safee M, Bearee R, and Neto P. Collision avoidance of redundant robotic manipulators using Newton’s method. *J Intell Robot Syst* 2020; 99(3): 673–681.
38. Shimizu M, Kakuya H, Yoon W, et al. Analytical inverse kinematic computation for 7-dof redundant manipulators with joint limits and its application to redundancy resolution. *IEEE Trans Robot* 2008; 24(5): 1131–1142.
39. Ruben S and Erwin A. Inverse kinematics with the kinematics and dynamics library (KDL). Orocos Developers. http://www.orocos.org/forum/orocos/orocos-users/inverse-kinematics-kdl (2010, accessed July 2020).
40. Andres M, Antonio B, and Jaime DC. The natural-CCD algorithm, a novel method to solve the inverse kinematics of hyper-redundant and soft robots. *Soft Robot* 2018; 5(3): 242–257.
41. Trivedi D, Rahn CD, Kier WM, et al. Soft robotics: biological inspiration, state of the art, and future research. *Appl Bionics Biomech* 2008; 5(3): 99–117.
42. Kim S, Laschi C, and Trimmer B. Soft robotics: a bioinspired evolution in robotics. *Trends Biotechnol* 2013; 31(5): 287–294.