Collision-Free Kinematics for Redundant Manipulators in Dynamic Scenes using Optimal Reciprocal Velocity Obstacles*

Liangliang Zhao\textsuperscript{1,2}, Jingdong Zhao\textsuperscript{1}, Hong Liu\textsuperscript{1}, and Dinesh Manocha\textsuperscript{2}

\textit{Abstract—} We present a novel algorithm for collision-free manipulation of multiple manipulators in a shared workspace with moving obstacles. Our optimization-based approach simultaneously handles collision-free constraints based on reciprocal velocity obstacles and inverse kinematics constraints. We present an efficient method based on particle swarm optimization that can generate collision-free configurations for each redundant manipulator. Furthermore, our approach can be used to compute safe and oscillation-free trajectories. We highlight the real-time performance of our algorithm on multiple Baxter robots with 14-DOF manipulators operating in a workspace with dynamic obstacles.

I. INTRODUCTION

Redundant manipulators are widely used in complex and cluttered environments for various kinds of tasks. Moreover, many applications use multiple manipulators that work cooperatively in an environment with moving obstacles (e.g. humans). To perform the tasks, the planning algorithms must compute trajectories that simultaneously satisfy two sets of constraints:

1. Inverse Kinematics Constraints: Given the pose of each end-effector, we need to compute joint coordinates that satisfy kinematics constraints \cite{1}.

2. Collision-free Constraints: The manipulations should not collide with each other or with the static or dynamic obstacles in the environment \cite{2}.

There is considerable prior work on achieving efficient inverse kinematics using numerical or optimization methods for redundant manipulators and good software packages are widely known \cite{4-9}. While these methods may account for self-collisions between different links of a manipulator, they do not consider collisions with other manipulators or dynamic obstacles. Similarly, there is a large body of work on collision-avoidance for multi-robot systems or dynamic obstacles. These works are mostly limited to rigid bodies or only take dynamics constraints into account (e.g., maximum acceleration or non-holonomic) \cite{20-30}; they have not been used for high-DOF manipulators.

Main Results: We consider the problem of computing a collision-free inverse kinematics solution for each redundant manipulator that operates in a three-dimensional workspace among other redundant manipulators and dynamic obstacles.

We present an efficient optimization-based algorithm that uses a novel combination of Inverse Jacobian methods, sequential quadratic programming (SQP), and reciprocal velocity obstacles (RVO). We extend the formulation of RVO to redundant manipulators and pose the problem as a high dimensional optimization problem with a large number of constraints. We present an efficient method to solve this problem using particle swarm optimization (PSO) \cite{3}. The basic idea is that we compute inverse kinematics solutions that satisfy the fitness function value of PSO while accounting for collision-avoidance constraints. Our algorithm initially uses a parallel inverse kinematic method (TRCK-IK) for each redundant manipulator in the workspace and builds a swarm of candidate particles for PSO. After PSO initialization, we continue to search for optimized inverse kinematics solutions by updating each particle. The RVO constraints of a redundant manipulator are used to determine whether the collision will occur in a time window. This can be calculated by the inverse kinematics solutions using linear programming. Based on the RVO constraints, the fitness values of all PSO’s particles are evaluated by the fitness function. The movements of each particle are guided by the fitness values to be optimized.

We have evaluated our method in environments with multiple Baxter robots and we compute collision-free inverse
kinematics solutions that satisfy the kinematic and RVO constraints of multiple Baxter arms (each arm has 7-DOF). We also include dynamic obstacles moving in the working environment. In general, the run time to generate a collision-free inverse kinematics solution for each redundant manipulator in a common workspace depends on the IK compute time and the number of iterations of the PSO method. In our benchmarks, it takes a few milliseconds on a single CPU core. Furthermore, the complexity increases almost linearly with the number of manipulators or dynamic obstacles.

The rest of the paper is organized as follows. In Section II, we discuss related work in inverse kinematics and the RVO algorithm. In Section III, we briefly review the inverse kinematics algorithm TRAC-IK, including the inverse Jacobian algorithm and the SQP algorithm. In Section IV, we extend the notion of velocity obstacles redundant manipulator. We combine the inverse kinematics algorithm with the RVO algorithm using PSO in Section V. In Section VI, we present simulation results of the arms of the Baxter robot in a dynamic working environment using our method.

II. PREVIOUS WORK

In this section, we give a brief overview of prior work on collision-free navigation of multiple robots and inverse kinematics of redundant manipulators.

Some of the widely used methods to generate valid inverse kinematics solutions are based on Inverse Jacobian methods. The Kinematics and Dynamics Library (KDL), distributed by the Orocos Project, can compute forward position kinematics to inverse kinematics based on the Inverse Jacobian method [4]. Beeson and Ames [5] propose an inverse kinematics algorithm called TRAC-IK that can improve several failure points of KDL. Starke et al. [6] present a biologically-inspired method for solving the inverse kinematics problem of fully-constrained robot geometries. Based on cyclic coordinate descent (CCD) and natural-CCD, Martí et al. [7] present an algorithm to solve the inverse kinematics problem of hyper-redundant and soft manipulators. Stillman et al. [8] introduce a unified representation for task space constraints by global randomized joint space path planning. Marcos et al. [9] introduce a method that combines the closed-loop pseudo-inverse method with a multi-objective genetic algorithm to solve the inverse kinematics of redundant manipulators. None of these methods considers collision-free constraints.

Various shape trajectory control approaches can achieve good performances in terms of obtaining an inverse kinematics solution of the redundant manipulator. These approaches include a spatial curve based on a tractrix curve [10], shape trajectory data from sidewinder rattlesnakes [11-13], mechanics modeling [14], plain spline fitting and extended spline fitting methods [15], the curvature gradient of a constant parameter along the segment arm [16], physical curves [17], a backbone curve [18], and an inchworm step [19].

There is considerable work on collision-free navigation of multiple robots sharing a common workspace. Some of the widely used solutions are based on velocity obstacles [20-22]. Berg et al. [23] propose the Reciprocal Velocity Obstacle algorithm for real time multi-agent navigation. This algorithm can be used to generate smooth paths for agents moving in the same environment and has been extended to handle bounds on acceleration [24] or simple airplanes in 3D [25].

Other velocity-obstacle-based methods account for dynamic constraints. These include differential-drive [26], double integrator [27], arbitrary integrator [28], car-like robots [29], linear quadratic regulator (LQR) controllers [30], non-linear equations of motion [31], etc. Some other algorithms like NH-ORCA [27] transfer non-linear equations of motion into a linear formulation. However, these methods are not designed for high-DOF redundant manipulators.

III. INVERSE KINEMATICS ALGORITHM

In this section, we briefly review inverse kinematics algorithms and highlight some of properties that are used in our PSO algorithm. The functional form of the inverse kinematics problem is given by:

$$\theta_{l-a} = f^{-1}(\xi_e)$$  \hspace{1cm} (1)

where $\xi_e$ is the desired pose of the end effector and $\theta_{l-a}$ are required joint coordinates. Because of redundancy, it is necessary to consider a numerical solution method to compute a feasible solution. The inverse kinematics solver TRAC-IK is a parallel method [6] that combines two inverse kinematics implementations, including a Newton-based convergence algorithm (KDL) and an SQP approach. It performs a parallel search using these methods and terminates when either of these algorithms converges to an inverse kinematics solution.

The most common values of a seed joint $\theta_{\text{seed}}$ for the Inverse Jacobian algorithm and the SQP algorithm are the current joint values. When all elements fall below a stopping criterion, the current joint vector $\theta$ is returned as an inverse kinematics solution.

a) KDL: Using a singular value decomposition, the Moore-Penrose pseudoinverse of the Jacobian $J^{-1}$ is computed to translate the partial derivatives in the joint space to the Cartesian space. Next, an inverse kinematics solution is generated by iterating the function

$$\begin{bmatrix}
\theta_{(k+1)} \\
\vdots \\
\theta_{(n(k+1))}
\end{bmatrix} =
\begin{bmatrix}
\theta_{(k)} \\
\vdots \\
\theta_{(n(k))}
\end{bmatrix} + J^{-1} \begin{bmatrix}
Err_k \\
\vdots \\
Err_n
\end{bmatrix}$$  \hspace{1cm} (2)

where $\theta_{(k+1)}(i = 1, \ldots, n)$ are the current joint values of the manipulator and $Err$ is the Cartesian error of the end effector, which can be computed by the previous joint values $\theta_{(k)}$.

b) SQP: This algorithm (as introduced in [5]) considers an inverse kinematics problem with the following form:

$$\arg \min_{\theta \in \mathbb{R}^n} (\theta_{\text{seed}} - \theta)^T (\theta_{\text{seed}} - \theta)$$

s.t. $g(\theta) \leq b$,

where $\theta_{\text{seed}}$ is the $n$-dimensional seed value of the joints and $g(\theta)$ is the constraints, including the value limits of each joint, the Euclidean distance, and the angular distance error.
IV. RECIPROCAL VELOCITY OBSTACLES FOR HIGH-DOF MANIPULATORS

In this section, we present our method for collision avoidance, which extends the notion of RVO to redundant manipulators based on RVO constraints. We derive the formulation for calculating the RVO constraints while they are induced by other redundant manipulators and dynamic obstacles in a time window. We also derive the process for computing the velocity of each link based on the inverse kinematics solutions.

To apply RVO during collision avoidance, each movable link of the redundant manipulator is decomposed into a series of spheres, and each movable joint is described by a sphere, as shown in Fig. 2. The number of spheres and their relative positions and radii are dynamically determined by the size of the links and joints. Also, we can choose different modelling purposes based on the working environments and task requirements. This work assumes that we know the number of spheres and their relative positions and radii that used for each manipulator’s decomposition. These spheres move around other manipulators and dynamic obstacles. Each dynamic obstacle is also assumed to be a sphere or is bounded by a sphere.

A. Reciprocal Velocity Obstacles

In the Cartesian space, let \( S_{M_i} \) be one of the spheres on the link of the redundant manipulator \( M_i \) and let \( S_{M_i(1)}, \ldots, S_{M_i(j)} \) be the obstacle or sphere on the other redundant manipulator \( M_j \). As shown in Fig. 2, the sphere \( S_{M_i} \) is centered at \( O_{M_i} = (o_{x_i}, o_{y_i}, o_{z_i}) \) with radius \( r_{M_i} \), and the spheres \( S_{M_i(1)}, \ldots, S_{M_i(j)} \) are centered at \( O_{M_i(1)} = (o_{x_i(1)}, o_{y_i(1)}, o_{z_i(1)}), \ldots, O_{M_i(j)} = (o_{x_i(j)}, o_{y_i(j)}, o_{z_i(j)}) \) with radii \( r_{M_i(1)}, \ldots, r_{M_i(j)} \). The length of the vector between the two centers can be defined as:

\[
\sqrt{(o_{x_i} - o_{x_j})^2 + (o_{y_i} - o_{y_j})^2 + (o_{z_i} - o_{z_j})^2}
\]

If \( r_{M_i} + r_{M_j} \leq D_{M_iM_j} \), we conclude that \( S_{M_i} \) and \( S_{M_j} \) are colliding.

The Minkowski sum of these spheres \( S_{M_i} \) and \( S_{M_j} \) can be described by the equation

\[
S_{M_i} \oplus S_{M_j} = \{ s_{M_i} + s_{M_j} \mid s_{M_i} \in CH(S_{M_i}), s_{M_j} \in CH(S_{M_j}) \}
\]

where \( CH(S_{M_i}) \) and \( CH(S_{M_j}) \) are the convex hulls of two spheres \( S_{M_i} \) and \( S_{M_j} \). The RVO for sphere \( S_{M_i} \) induced by sphere \( S_{M_j} \) for time window \( \tau \) is then given as follows:

\[
RVO_{S_{M_i}|S_{M_j}}^\tau = \{ v \mid \lambda^\tau(O_{M_i}, v - v_{M_j}) \cap S_{M_j} \neq \emptyset, \quad \lambda^\tau(O_{M_i}, v - v_{M_j}) \cap S_{M_j} \}
\]

where \( -S_{M_i} = \{-s_{M_i} \mid s_{M_i} \in CH(S_{M_i}) \} \), \( v_{M_j} \) is the velocity vector of the sphere \( S_{M_j} \), and \( \lambda^\tau(O_{M_i}, v - v_{M_j}) \) is a ray starting at \( (o_{x_i}, o_{y_i}, o_{z_i}) \) with direction \( v \):

\[
\lambda^\tau(O_{M_i}, v - v_{M_j}) = \{ O_{M_i} + t(v - v_{M_j}) \mid t \in [0, \tau] \}
\]

If we let sphere \( S_{M_i} \) have velocity \( v_{M_i} \), we observe that sphere \( S_{M_i} \) may collide with sphere \( S_{M_j} \) in the time interval \([0, \tau]\) if the relative velocity vector of \( v_{M_i} - v_{M_j} \) is inside the
region \( RVO_{S_{m1}}^c \). To avoid a possible collision before time \( \tau \), the relative velocity vector of \( v_{M_k} - v_{(M_2,m_2)} \) must be outside \( RVO_{S_{m1}}^c \).

### B. RVO Constraints

As Section IV-A described, the principle of RVO can be used to navigate a sphere in the dynamic workspace without collisions. To select a velocity for sphere \( S_{M_1} \), we introduce a constraint defined with respect to the velocity \( v_{M_k} - v_{(M_2,m_2)} \) and the region \( RVO_{S_{m1}}^c \).

Having identified a velocity \( v_{M_k} \) for sphere \( S_{M_1} \), the vector \( \omega_{m_2} \) from \( v_{M_k} - v_{(M_2,m_2)} \) to the closest point of the boundary of the \( RVO_{S_{m1}}^c \), as shown in Fig. 3, is defined as follows:

\[
\omega_{m_2} = \frac{p}{\| p \|} \cdot \omega_{m_2}
\]

where \( \partial RVO_{S_{m1}}^c \) is the boundary of the velocity obstacle. Let \( p \) represent the vector from point \( (O_{M_k} - O_{(M_2,m_2)}) / \tau \) to \( v_{M_k} - v_{(M_2,m_2)} \). As with previous definitions, the constraint factor \( \psi_{m_2} \) is defined as follows:

\[
\psi_{m_2} = \frac{p}{\| p \|} \cdot \omega_{m_2}
\]

If \( \psi_{m_2} > 0 \), then the sphere \( S_{M_1} \) will collide with the obstacle sphere \( S_{(M_2,m_2)} \). If \( \psi_{m_2} \leq 0 \), there will be no such collision within the time interval \([0, \tau] \).

### V. RELATING INVERSE KINEMATICS ALGORITHM TO RVO

#### A. Particles Initialization

A basic variant of the PSO algorithm is a swarm of candidate particles. These particles are moved around in the search space according to the fitness function. The movements of the particles are guided by the best known positions of individual particles in the search and by the entire swarm's best-known position.

In this work, we assume that the number of DOF of all manipulators in the workspace is \( n \). The PSO consists of particles \( x_{m_n} = (x_{(m_1)}, \ldots, x_{(m_N)}) \) for \( ms = 1, \ldots, N \) representing the number of particles in the swarm. Thus, we can encode the joint variable configuration \( \Theta \) as the particle \( x_{m_n} \) for the individuals, each of whom has a position \( x_{m_n} \in \mathbb{R}^n \) in the search space (caused by the joint limits of each manipulator) and a velocity \( v_{m_n} \in \mathbb{R}^n \). In contrast to the basic PSO algorithm, the position \( x_{(m_1)}, \ldots, x_{(m_N)} \) represents a seed joint \( \Theta_{m_2} \) for the IK algorithm (described in Section III) instead of representing a possible inverse kinematics solution. We can also compute the inverse kinematics solutions through the seed joint and the IK algorithm. During the process of the IK algorithm, the joint limit constraints are used to ensure that
the position remains within the fundamental limits of the manipulator. These limits can be defined as:

$$\theta_{mn,\min} \leq x_{mn} \leq \theta_{mn,\max}, \quad mn = 1, \ldots, n$$  \hspace{1cm} (12)

B. Fitness Function

To avoid multiple moving obstacles in dynamic environments, the constraint factor $\psi$ of RVO is introduced in the fitness function $\Phi(x_m^k)$, which must be minimized to measure the fitness of a particle. In addition to finding a smoother motion, the rank-based selection strategy generates an inverse kinematics solution that bears a relation to the current joint values. The strategy aims to select a solution that minimizes the value of the fitness function $\Phi(x_m^k)$.

$$\Phi(x_m^k) = \alpha \psi(x_m^k) + \beta M(x_m^k)$$  \hspace{1cm} (13)

$$M(x_m^k) = \sum_{mn=1}^{n} \lambda_{mn} \left| \theta_{mn} - \theta_{mn,\min} \right|$$  \hspace{1cm} (14)

where $\psi(x_m^k)$ is the constraint factor computed by summing the RVO constraint of all spheres in the workspace, $M(x_m^k)$ is the angular distance error, $\alpha$ and $\beta$ are the constants used to regulate the variation range of $\psi(x_m^k)$ and $M(x_m^k)$, $\lambda_{mn}$ is the weight of each joint angle, $\theta_{mn}$ is the inverse kinematics solution of all manipulators, and $\theta_{mn,\min}$ is the joint variable of the current configuration.

C. Particle Swarm Optimization

After PSO initializes with a group of particles and a fitness function, it continues to search for optimia by updating generations. Based on the fitness function value $\Phi(x_m^k)$, let $p_{mn}$ be the best known position of particle $x_m^k$, and let $g$ be the best known position obtained so far by any particle in the swarm. In every iteration, each particle $x_m^k$ is updated by $p_{mn}$ and $g$. Each particle updates its velocity and positions with Equations (15) and (16).

$$v_{mn}^{k+1} = v_{mn}^k + c_1 r_1^k (p_{mn} - x_{mn}^k) + c_2 r_2^k (g - x_{mn}^k)$$  \hspace{1cm} (15)

$$x_{mn}^{k+1} = x_{mn}^k + v_{mn}^{k+1}$$  \hspace{1cm} (16)

where $c_1$ and $c_2$ are learning factors that can control the behavior and efficacy of the PSO method and $r_1^k$ and $r_2^k$ are the standard. In our formulation, the termination criterion of PSO is the number of iterations performed. The fitness function value $\Phi(x_m^k)$ can be used to measure the success of the inverse kinematics solution where the RVO constraint of all spheres in the workspace is less than zero.

VI. IMPLEMENTATION AND PERFORMANCE

In this section, we highlight the performance of our algorithm in simulated environments that contain multiple redundant manipulators in the 3D space. We exemplify our method in the dynamic working environment with many moving obstacles, and no assumptions are made about their trajectories. It should be noted that our method is applicable for many manipulators and for multiple dynamic obstacles in the working environment. The general method of the combined TRAC-IK and RVO for the right arm of Baxter. The three-dimensional working environment contains a dynamic obstacle (white sphere), which moves at a constant velocity. The red area represents the velocity obstacle RVO $O_{(x_{1}^k, \ldots, x_{n}^k)}$ for the sphere $O_{(M, A)}$ (green sphere) induced by $O_{(M, A)}$ (white sphere) in the time window $\tau$ in the three-dimensional workspace. The yellow sphere represents the neighbor region $R_{(M, A)}$.

These simulations are initialized by decomposing the redundant manipulator into a series of spheres. The fitness values are sufficiently small when the distance between the sphere and the obstacle is sufficiently large. For simplicity, we do not need to take all the spheres into account when computing the RVO constraints. Therefore, a neighbor region $R_{(i, k)}$ ($k = 1, \ldots, m$) around the current position of sphere $O_{(i, k)}$ is defined and we then only consider the spheres of other manipulators and obstacles inside this region (the yellow sphere in Fig. 5). Furthermore, the size of $R_{(i, k)}$ can be determined by the velocity of each sphere and the size of the time-step $\tau$.

We use TRAC-IK to generate enough valid configurations and we use RVO to define the constraint factor $\psi$. We use the PSO algorithm to relate the inverse kinematics algorithm to the RVO and generate an inverse kinematics solution out of the valid configurations. The simulation terminates as soon as all the constraints of the fitness function values $\Phi$ are satisfied by the inverse kinematics solution. Finally, the inverse kinematics solution is assigned as the value of each joint for the manipulator.

Table 1 shows the results of applying our inverse kinematics method to the redundant manipulators in different dynamic environments. The results show that the complexity...
of the PSO increases as a linear function of the number of manipulators and the dynamic obstacles. Specifically, PSO achieves to maintain the previously best known position of a particle in a swarm, while concurrently searching for new inverse kinematics solutions and improve the results. Our benchmarks show that the overall computation time is directly proportional to the number of DOF, the size of PSO’s particles, and the number of iterations of our optimization algorithm.

Our algorithm is implemented in C++ using an ROS Kinetic Kame and Gazebo (Version 7). The code runs on a single notebook CPU i7-7700HQ at 2.81 GHz. The desired end-effector movement directions selected for evaluation are on the right and left arms of the Baxter robot. The end-effector starts from an untucked position. The configurations of two arms (14-DOF) and the position of the obstacle are shown in Fig. 6. The configurations of four arms (28-DOF) are shown in Fig. 7 (a) and (b). The configurations of six arms (42-DOF) are shown in Fig. 7 (c) and (d). The simulation results demonstrate that our method can deal with two redundant manipulators to adjust configurations and avoid collisions without requiring the addition of a collision detection method. Moreover, all joint angles are within their specified limits.

VII. CONCLUSION AND FUTURE WORK

We present a novel method for solving the inverse kinematics problem of multiple redundant manipulators by computing collision-free configuration. Our approach combines the inverse kinematics constraints with collision avoidance constraints and uses a novel optimization schedule. To demonstrate the feasibility and efficiency of this method, we have highlighted the performance in our simulator with multiple Baxter robots operating in a shared workspace with moving obstacles. Our algorithm has made full use of the dexterity of redundant manipulators in different dynamic environments, enhancing the performance and increasing the flexibility of the manipulators. Our algorithm is compute a feasible solution in a few milliseconds for multiple robots operating in close proximity.

Our approach has some limitations. The collision avoidance formulation is conservative and it is hard to guarantee that our method can find a feasible solution in all configurations. We would like to evaluate the performance in dynamic scenes with human or other complex obstacles. Furthermore, we can integrate our approach with robots and evaluate their performance in a real-world environment.

| DOF (n) | Environment | Time window $\tau$ (s) | PSO Iterations $T$ | Number of particles $N$ | Processing time for a solution (all arms) (ms) |
|---------|-------------|------------------------|-------------------|------------------------|-----------------------------|
| 14      | One Baxter Robot and one dynamic obstacle | 5 | 2 | 2 | 4.66 |
| 14      | One Baxter Robot and two dynamic obstacle | 5 | 4 | 3 | 12.4 |
| 14      | One Baxter Robot and three dynamic obstacle | 5 | 4 | 4 | 19.2 |
| 28      | Two Baxter Robots | 5 | 3 | 3 | 20.4 |
| 42      | Three Baxter Robots | 5 | 4 | 3 | 45 |

TABLE I. INVERSE KINEMATICS RESULTS OF REDUNDANT MANIPULATORS USING A SINGLE CPU CORE OF 2.81 GHz INTEL I7-7700HQ
REFERENCES

[1] R. J. Webster, and B. A. Jones, “Design and Kinematic Modeling of Constant Curvature Continuum Robot: A Review,” The International Journal of Robotics Research, vol. 29, pp. 1661-1683, 2010.

[2] P. Fiorini, and Z. Shiller, “Motion Planning in Dynamic Environments Using Velocity Obstacles,” The International Journal of Robotics Research, vol. 17, pp. 760-772, 1998.

[3] J. Kennedy, and R. Eberhart, “Particle swarm optimization,” 1995 Int. Con. on Neural Networks, vol. 4, pp. 1942-1948.

[4] Orocos Kinematics and Dynamics, http://www.orocos.org,” online: 01-2016.

[5] N. Sukavanam, and R. Balasubramanian, “An Optimization Approach to Solve the Inverse Kinematics of Redundant Manipulator,” International Journal of Information And Systems Sciences, vol. 6, pp. 414-423, 2011.

[6] P. Beeson, and B. Ames, “TRAC-IK: An open-source library for improved solving of generic inverse kinematics,” 2015 IEEE-RAS 15th In. Con. on Humanoid Robots (Humanoids), pp. 928-935.

[7] M. Andrés, B. Antonio, and d. C. Jaime, “The Natural-CCD Algorithm, a Novel Method to Solve the Inverse Kinematics of Hyper-redundant and Soft Robots,” Soft Robotics, vol. 5, pp. 242-257, 2018.

[8] M. Stilman, “Global Manipulation Planning in Robot Joint Space With Task Constraints,” IEEE Transactions on Robotics, vol. 26, pp. 576-584, 2010.

[9] M. d. G. Marcos, J. A. Tenreiro Machado, and T. P. Azevedo-Perdicóulis, “A multi-objective approach for the motion planning of redundant manipulators,” Applied Soft Computing, vol. 12, pp. 589-599, 2012.

[10] S. Sreenivasan, P. Goel, and A. Ghosal, “A real-time algorithm for simulation of flexible objects and hyper-redundant manipulators,” Mechanism and Machine Theory, vol. 45, pp. 454-466, 2010.

[11] C. Gong et al., “Kinematic gait synthesis for snake robots,” The International Journal of Robotics Research, vol. 35, pp. 100-113, January 1, 2016.

[12] R. Ariizumi, M. Tanaka, and F. Matsuno, “Analysis and heading control of continuum planar snake robot based on kinematics and a general solution thereof,” Advanced Robotics, vol. 36, pp. 301-314, 2016.

[13] Z. Weikun, G. Chaohui, and H. Choset, “Modeling rolling gaits of a snake robot,” in (ICRA), 2015 IEEE Int. Con. Robotics and Automation, pp. 3741-3746.

[14] D. B. Camarillo, C. F. Milne, C. R. Carlson, M. R. Zinn, and J. K. Salisbury, “Mechanics Modeling of Tendon-Driven Continuum Manipulators,” IEEE Transactions on Robotics, vol. 24, pp. 1262-1273, 2008.

[15] A. Bayram, and M. K. Ögören, “The position control of a spatial binary hyper redundant manipulator through its inverse kinematics,” Part C: Journal of Mechanical Engineering Science, vol. 227, pp.359-372, 2012.

[16] M. Ivanescu, N. Popescu, and D. Popescu, “The shape control of tentacle arms,” Robotica, vol. 33, pp.684–703, 2014.

[17] K. Nakagaki, S. Follmer, and H. Ishii, “LineFORM: Actuated Curve Interfaces for Display, Interaction, and Constraint,” 28th Annual ACM Symposium on User Interface Software & Technology, pp. 333-339, 2015.

[18] G. S. Chirikjian, and J. W. Burdick, “A modal approach to hyper-redundant manipulator kinematics,” IEEE Transactions on Robotics and Automation, vol. 10, pp. 343-354, 1994.

[19] N. Shvalb, B. B. Moshe, and O. Medina, “A real-time motion planning algorithm for a hyper-redundant set of mechanisms,” Robotica, vol. 31, pp. 1327-1335, 2013.

[20] Z. Shiller, F. Large, and S. Sekhavat, “Motion planning in dynamic environments: obstacles moving along arbitrary trajectories,” IEEE Int. Con. on Robotics and Automation, vol. 4, pp. 3716-3721, 2001.

[21] B. Kluge, and E. Prassler, “Reflective navigation: individual behaviors and group behaviors,” IEEE Int. Con. on Robotics and Automation, vol. 4, pp. 4172-4177, 2004.

[22] Y. Abe, and M. Yoshiki, “Collision avoidance method for multiple autonomous mobile agents by implicit cooperation,” IEEE/RSJ Int. Con. on Intelligent Robots and Systems, vol. 3, pp. 1207-1212, 2001.

[23] J. v. d. Berg, L. Ming, and D. Manocha, “Reciprocal Velocity Obstacles for real-time multi-agent navigation,” IEEE Int. Con. on Robotics and Automation, pp. 1928-1935, 2008.

[24] J. v. d. Berg, J. Snape, S. J. Gay, and D. Manocha, “Reciprocal collision avoidance with acceleration-velocity obstacles,” IEEE Int. Con. on Robotics and Automation, pp. 3475-3482, 2011.

[25] J. Snape, and D. Manocha, “Navigating multiple simple-airplanes in 3D workspace,” IEEE Int. Con. on Robotics and Automation, pp. 3974-3980, 2010.

[26] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, “Optimal Reciprocal Collision Avoidance for Multiple Non-Holonomic Robots,” Distributed Autonomous Robotic Systems: The 10th International Symposium, pp. 203-216, 2013.

[27] E. Lalish, and K. A. Morgansen, “Distributed reactive collision avoidance,” Autonomous Robots, vol. 32, pp. 207-226, 2012.

[28] M. Rufli, J. Alonso-Mora, and R. Siegwart, “Reciprocal Collision Avoidance With Motion Continuity Constraints,” IEEE Transactions on Robotics, vol. 29, pp. 899-912, 2013.

[29] J. Alonso-Mora, A. Breitenmoser, P. Beardsley, and R. Siegwart, “Reciprocal collision avoidance for multiple car-like robots,” IEEE Int. Con. on Robotics and Automation, pp. 360-366, 2012.

[30] D. Bareiss, and J. v. d. Berg, “Reciprocal collision avoidance for robots with linear dynamics using LQR-Obstacles,” IEEE Int. Con. on Robotics and Automation, pp. 3847-3853, 2013.

[31] D. Bareiss, and J. van den Berg, “Generalized reciprocal collision avoidance,” The International Journal of Robotics Research, vol. 34, pp. 1501-1514, 2015.