Inverse kinematics for cooperative mobile manipulators based on self-adaptive differential evolution

Jesus Hernandez-Barragan, Carlos Lopez-Franco, Nancy Arana-Daniel and Alma Y. Alanis
Department of Computer Science, University of Guadalajara, Guadalajara, Jalisco, Mexico

ABSTRACT

This article presents an approach to solve the inverse kinematics of cooperative mobile manipulators for coordinate manipulation tasks. A self-adaptive differential evolution algorithm is used to solve the inverse kinematics as a global constrained optimization problem. A kinematics model of the cooperative mobile manipulators system is proposed, considering a system with two omnidirectional platform manipulators with n DOF. An objective function is formulated based on the forward kinematics equations. Consequently, the proposed approach does not suffer from singularities because it does not require the inversion of any Jacobian matrix. The design of the objective function also contains penalty functions to handle the joint limits constraints. Simulation experiments are performed to test the proposed approach for solving coordinate path tracking tasks. The solutions of the inverse kinematics show precise and accurate results. The experimental setup considers two mobile manipulators based on the KUKA Youbot system to demonstrate the applicability of the proposed approach.

INTRODUCTION

Dual-arm systems are commonly used to perform complex cooperative tasks such as human-like tasks in domestic and industrial environments (Yang et al., 2019). However, the cooperative tasks for dual-arm systems are limited by the workspace provided by the robot manipulators. Mobile platforms can be used to extend the workspace of the manipulators. For this reason, a cooperative system conformed by two mobile manipulators increases the capabilities with greater accessibility to perform a cooperative task. Dual-arm manipulation can be defined as physical interaction with an object, exerting forces to move or reshape it using two robot manipulators (Smith et al., 2012). There are two main types of robotic manipulation, non-coordinated manipulation, and coordinated manipulation. In non-coordinate manipulation, manipulators perform different tasks independently. In this case, the motion of each manipulator can be analyzed separately. In coordinated manipulation, manipulators are physically interacting with each other. In this case, the relative motion has to be adequately controlled to perform a task. These definitions of cooperative manipulation are not limited to dual-arm systems. Indeed, such concepts have been applied to different cooperative robot systems.
This work introduces a model description of a cooperative mobile manipulators system for coordinate manipulation tasks.

The relative Jacobian matrix is commonly used to solve the coordinate manipulation of dual-arm systems (Lewis, 1996; Jamisola & Roberts, 2015). This method considers the dual-arm system as a single redundant manipulator. Therefore, control algorithms for a single manipulator can be used to control dual-arm systems modeled with the relative Jacobian matrix. The relative Jacobian matrix is extended with the Jacobian null space projection for cooperative tasks of redundant manipulators (Jamisola & Roberts, 2015; Freddi et al., 2016).

Kinematic and dynamic control strategies have been presented to deal with cooperative manipulation. Coordinated cooperative controllers for trajectory tracking of a dual-arm system mounted on an omnidirectional platform is presented in Molina & Ortiz (2018) and Ortiz et al. (2018). The control strategy is based on the kinematic modeling of the entire system, and it allows to transport a common object. In Mora-Aguilar et al. (2018), the authors describe a centralized cooperative control algorithm for two heterogeneous mobile manipulators. The cooperative system is conformed by a mobile differential platform manipulator and an omnidirectional platform manipulator. This control strategy also considers the kinematic model for the two heterogeneous mobile manipulators. A cooperative transportation control strategy for multiple mobile manipulators is proposed in Chen & Kai (2018). This work deals with the problem of distributed control for an object tracking with energy and manipulability optimization. The control strategy is based on the kinematics of redundant robots. In Wu, He & Liu (2016), the authors present a leader-follower structure for cooperative manipulation of two mobile manipulators. The compliance feature is achieved with a classic PID controller, which considers the system dynamic equations. In Pajak (2019), the authors propose a real-time trajectory generation for two cooperating mobile manipulators moving a common rigid object. This proposal introduces a leader-follower control scheme based on the kinematics and dynamics of the system. Moreover, control limitations and collision avoidance are considered. In Korayem & Dehkordi (2019a), the authors present the dynamics equation of closed kinematics cooperative mobile manipulator. The motion control is achieved based on the kinematic chain and applying it in the dynamic model. The dynamic model comprising $M$ manipulators are fixed in a nonholonomic mobile platform. The same authors used these principle concepts to develop a dynamic model for multiple mobile manipulator systems (Korayem & Dehkordi, 2019b). A fully distributed cooperative scheme for a networked mobile manipulator is proposed in Ren, Sosnowski & Hirche (2020). Distributed adaptive controllers are used to achieve motion synchronization, which deals with kinematics and dynamics uncertainties. Manipulation experiments with three mobile manipulators confirm the applicability in the distributed cooperation task.

Motion and path planning for cooperative manipulation have also been presented. In Tallamraju et al. (2019), a kinematic motion planning algorithm for cooperative manipulation is proposed. A convex decentralized model-predictive controller is formulated to plan collision-free trajectories for the robot formation. The problem of path
Planning for spatial payload transportation for mobile manipulators is addressed in Tallamraju, Sripada & Shah (2019). This approach uses non-linear multi-objective optimization to compute optimal paths for the robot formations. Moreover, rapidly-exploring random trees are used to find a kinematically feasible and collision-free trajectory.

The mentioned kinematics and dynamics control strategies use the information of a Jacobian matrix to map the end-effector velocities to the manipulator joint velocities. The joint velocities are used to control the motion of the mobile manipulator, and the end-effector velocities are commonly computed by an error between a desired and a current end-effector positions. The drawback of these methods is the singularities that occur when the Jacobian matrices become rank deficient (Hu, Huang & Yang, 2015). Singularities represent joint configurations where the mobility of the manipulator is reduced, infinite solutions for the inverse kinematics may exist, small velocities of the end-effector may cause large velocities for the joint configuration (Siciliano et al., 2008).

To overcome this inconvenience, this work introduces a kinematic model of cooperative mobile manipulators systems, which is based only on forward kinematics equations. The solutions of the inverse kinematics are obtained based on a global optimization problem, where it is not required the inclusion of any Jacobian matrix.

In a mobile manipulator system, the combination of the degrees of freedom (DOF) of the mobile platform and the manipulator becomes the system into a redundant robot. Consequently, a cooperative mobile manipulator system is considered redundant. Redundancy occurs when the degree of motion is higher than the number of necessary variables to perform a task (Freddi et al., 2016). The inverse kinematics for redundant robots becomes difficult to solve because redundancy admits several joint configurations to reach the same end-effector pose. For this reason, meta-heuristic algorithms are commonly applied to solve the inverse kinematics problem. A trajectory planning of robot manipulators is proposed in Savsani, Jhala & Savsani (2016). The authors included a comparative study among the artificial bee colony (ABC), biogeography-based optimization (BBO), the gravitational search algorithm (GSA), cuckoo search algorithm (CS), firefly algorithm (FA), bat algorithm (BA), and teaching-learning-based optimization (TLBO) algorithms. The TLBO, ABC, and CS algorithms showed a significant improvement over the others.

The inverse kinematics solutions for robot manipulators is presented in Ayyildz & Çetinkaya (2016). This work included a comparative study among the genetic algorithm (GA), particle swarm optimization (PSO), quantum particle swarm optimization (QPSO), and GSA algorithms. The comparative results showed that QPSO is more useful to solve the inverse kinematics than the others. In Ziwu, Chunguang & Lining (2016), a trajectory optimization method for a humanoid manipulator based on differential evolution (DE) is proposed. The DE algorithms minimize the maximum acceleration of the trajectory. PSO algorithm has been used to solve the inverse kinematics for high-DOF inverse kinematics (Collinsm & Shen, 2017). The authors compared the performance of the Constriction factor PSO (CFPSO) and Bare Bones PSO (BB-PSO). The results reported that CFPSO is more convenient to solve the inverse kinematics for high-DOF robots.
In *El-Sherbiny, Elhosseini & Haikal (2018)*, a new variant of ABC is introduced to solve the inverse kinematics of robot manipulators, which is called KABC. The reported results prove that the KABC algorithm performed better than the classical ABC. A PSO algorithm was used to solve the coordinated trajectory planning of dual-arm space robot (*Wang et al., 2018*). In this case, the PSO version uses dynamic weighting factors, and it can handle constraints.

In *Kumar et al. (2019)*, the ABC and Grey Wolf Optimization (GWO) algorithms are considered to solve the inverse kinematics for arm manipulators. The ABC algorithm performed better than GWO with a better minimum error. The path planning optimization of robotic manipulators has been addressed in *Šegota et al. (2020)*. The authors included a comparison among the GA, Simulating Annealing (SA), and DE algorithms, where GA proves to be better than the other algorithms. The Flower Pollination Algorithm (FPA) is another promising meta-heuristic algorithm (*Yang, 2012*). It has been recently applied successfully to solved mobile robots applications (*Low, Ong & Cheah, 2019; Mishra & Deb, 2020*). Based on the studies mentioned above, the DE algorithm has shown to be adequate to solve inverse kinematics problems. However, in *Fan, Xie & Zhou (2019)* a new variant of DE is proposed to improve the local exploitation and global exploration, with convergence accuracy and fast converge rate. This version is called self-adaptive mutation differential evolution constrained optimization (SDE), and it has been used to determine the optimum path generation of a rock-drilling manipulator with nine DOF based on the SDE algorithm. The SDE algorithm solves the inverse kinematics for a single manipulator system. In this work, it is proposed to solve the inverse kinematics of cooperative mobile manipulator system based on the SDE algorithm. This proposed approach introduces the kinematics model for the cooperative system which considers two omnidirectional platform mobile manipulators. Moreover, a comparative study is included to compare the performance of SDE against CFPSO, FPA and KABC.

The contributions of this paper are summarized as follows: The inverse kinematics of cooperative mobile manipulators is solved as a global constrained optimization problem. A kinematic model for the cooperative mobile manipulator system is proposed and an objective function is formulated based on the forward kinematics equations. The proposed approach avoids singularity configurations since it does not require the use of any Jacobian matrix. The formulation of the objective function includes the use of the penalty function to handle constraints. The constraints represent the joint limits of the mobile manipulators, which is crucial for real-world implementations. Finally, the considered cooperative mobile manipulators are conformed by two omnidirectional platform manipulators with \( n \) DOF.

This article is organized as follows: The “Cooperative Mobile Manipulator System” section provides the kinematic model of the considered cooperative system which is conformed by two omnidirectional platform manipulators. In the section “Description of the Proposed Approach”, the objective function is formulated, the description of the SDE algorithm is provided, and the inverse kinematics algorithm for cooperative mobile manipulators is presented. The “Experimental Results” section presents the results of the
proposed approach solving cooperative path tracking tasks. The comparison study is also included in this section. Finally, the conclusion is given in the “Conclusions” section.

Cooperative mobile manipulators system

This work introduces an approach to solve the inverse kinematics for cooperative mobile manipulators system. The proposed method is based on the forward kinematics equations. In this section, kinematics concepts and important details are presented. First, a brief description about the Denavit-Hartenberg model for the kinematics equations of robotic manipulators is presented. Then, a kinematic model for a mobile manipulator is presented, where a manipulator of \( n \) total degrees of freedom (DOF) attached to an omnidirectional mobile platform of three DOF is considered. Finally, the forward kinematics equations for a cooperative mobile manipulators model are provided.

Manipulator kinematics

Robotic manipulators are conformed by a series of joints connected by links that represent a kinematic chain. In a manipulator with an open kinematic chain, the total DOF are defined by the total number of joints (Spong & Vidyasagar, 2008; Siciliano et al., 2008).

A joint variable \( q_m \) is defined as follows

\[
q_m = [q_1 \ q_2 \ q_3 \ \cdots \ q_n]^T
\]

where each joint value \( q_j \) with \( j = 1,2,3,\ldots,n \) represents an articulation, and \( n \) is the total DOF.

Commonly, revolute or prismatic joints are used to represent an articulation. The joint value \( q_j \) is an angle of rotation for revolute joints. Otherwise, the joint value \( q_j \) is a displacement for prismatic joints. In other words

\[
q_j = \begin{cases} 
\theta_j & \text{if joint } j \text{ is revolute} \\
 d_j & \text{if joint } j \text{ is prismatic}
\end{cases}
\]

The manipulator kinematics model can be obtained based on its kinematics chain, see Fig. 1. Each link \( j \) is represented by a homogeneous matrix \( ^jT_j \) that transforms the frame attached to the link \( j − 1 \) into the frame link \( j \). The transformation \( ^0T_n \) represents a homogeneous matrix that contains the position and orientation of the end-effector. The kinematics model can be described using the Denavit–Hartenberg (DH) convention, where the homogeneous matrix \( ^jT_j \) is expressed as

\[
^jT_j = \begin{bmatrix}
c\theta_j & -s\theta_j c\alpha_j & s\theta_j s\alpha_j & a_j c\theta_j \\
s\theta_j & c\theta_j c\alpha_j & -c\theta_j s\alpha_j & a_j s\theta_j \\
0 & s\alpha_j & c\alpha_j & d_j \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

where \( \theta_j \) is a joint angle, \( a_j \) is a link length, \( d_j \) is a link offset and \( \alpha_j \) is a link twist. For brevity, the sin and cos operations are represented with the letters \( s \) and \( c \), respectively. When a manipulator has a revolute joint, then the parameter \( \theta_j \) becomes the joint variable and the other parameters remain constant. When the manipulator has a prismatic joint,
the parameter \( d_j \) becomes now the joint variable and the rest of the parameters remain constant.

The forward kinematics consists in computing the end-effector pose \( ^0T_n \) given the joint variable \( q_m \). The forward kinematics can be computed as

\[
^0T_n(q_m) = ^0T_1(q_1)^1T_2(q_2)^2T_3(q_3) \cdots ^{n-1}T_n(q_n)
\]

(4)

where the matrix \( ^0T_n \) is expressed as

\[
^0T_n(q_m) = \begin{bmatrix}
    r_{11} & r_{12} & r_{13} & t_x \\
    r_{21} & r_{22} & r_{23} & t_y \\
    r_{31} & r_{32} & r_{33} & t_z \\
    0 & 0 & 0 & 1
\end{bmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}
\]

(5)

where the orientation of the end-effector is represented by the matrix \( R \), and its Cartesian position is given by the vector \( t \). More detailed information manipulator kinematics can be found in Craig (2005), Spong & Vidyasagar (2008) and Siciliano et al. (2008).

**Mobile manipulator kinematics**

Mobile manipulators are conformed by one or more manipulators attached to a mobile platform. Conventional mobile robots such as unicycle, differential drive, and car-like mobile robots are used to increase the manipulator workspace. However, these platforms have limited movement capabilities due to their nonholonomic kinematics constraints (Li et al., 2016). Nonholonomic constraints limit some driving directions of a mobile robot. Conventional mobile robots are usually subjected to nonholonomic...
constraints, which means that not all driving directions are possible. There are also many possible solutions to arrive to the desired pose. The differential, tricycle, and car-like robots can move forward, backward, or rotate to change their orientation, but they cannot move in the lateral direction. In contrast, omnidirectional mobile platforms improve the movement capabilities, which allow them simultaneously to move towards any position and reach any desired orientation (Kundu et al., 2017; Wu et al., 2017).

The omnidirectional mobile platforms have no limitation in their velocity space so all directions of motion in the state space are possible. This section introduces a kinematic model of a mobile manipulator conformed by a robotic manipulator of \( n \) DOF attached to an omnidirectional mobile platform.

The kinematic chain of mobile manipulators is described in Fig. 2. The homogeneous matrix \(^wT_b\) defines the position and orientation of the mobile platform. The transformation \(^bT_m\) is a constant homogeneous matrix between the mobile platform frame and the manipulator base. The matrix \(^mT_e\) can be computed based on the DH model of the manipulator.

Considering an omnidirectional mobile platform, the pose of the robot with respect to the world frame \( w \) is given by three DOF, which are the positions \( x_b \) and \( y_b \), and the orientation \( \theta_b \). Then, the matrix \(^wT_b\) can be defined as

\[
^wT_b = \begin{bmatrix}
\cos(\theta_b) & -\sin(\theta_b) & 0 & x_b \\
\sin(\theta_b) & \cos(\theta_b) & 0 & y_b \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]  

(6)

The matrix \(^bT_m\) is constant and it adjusts the distance from the mobile platform base frame \( b \) to the manipulator base frame \( m \). The values \( d_x \), \( d_y \), and \( d_z \) are used to adjust the
distance in the direction of the x-axis, y-axis and z-axis, respectively. If the frame orientation is not needed to adjust, then the matrix $b^T_m$ can be described by

$$b^T_m = \begin{bmatrix}
1 & 0 & 0 & d_x \\
0 & 1 & 0 & d_y \\
0 & 0 & 1 & d_z \\
0 & 0 & 0 & 1
\end{bmatrix}$$

(7)

Let us consider a joint variable $q$ to represent the platform configuration $q_b = [x_b \; y_b \; \theta_b]^T$ and the manipulator configuration $q_m = [q_1 \; q_2 \; q_3 \; \cdots \; q_n]^T$. The joint variable for the mobile manipulator is given by $q = [q_b^T \; q_m^T]^T$.

Given the joint variable $q$, the computation of $^wT_c(q)$ which is the forward kinematics of the mobile manipulator can be obtained as

$$^wT_c(q) = ^wT_b(q_b) b^T_m m^wT_c(q_m)$$

(8)

where $^wT_c(q)$ represents the end-effector pose with respect to the world frame $w$.

**Cooperative mobile manipulators model**

In a cooperative mobile manipulators system, the mobile manipulators are physically interacting with each other, which means that their relative motions have to be adequately controlled to perform a task (Smith et al., 2012). This section introduces the kinematic model of a cooperative mobile manipulator system with omnidirectional mobile platforms.

The considered cooperative system is illustrated in Fig. 3. The homogeneous matrices $^wT_{c_1}$ and $^wT_{c_2}$ represent the kinematic model of the mobile manipulator 1 and 2,
respectively. Let define two joint variables $q_1$ and $q_2$ to represent the configuration of mobile manipulator 1 and 2, respectively. The forward kinematics for each mobile manipulator can be computed as $^wT_{e_1}(q_1)$ and $^wT_{e_2}(q_2)$ using Eq. (8). Both matrices are defined as

$$^wT_{e_1}(q_1) = \begin{bmatrix} R_{e_1} & t_{e_1} \\ 0 & 1 \end{bmatrix}$$ (9)$$

$$^wT_{e_2}(q_2) = \begin{bmatrix} R_{e_2} & t_{e_2} \\ 0 & 1 \end{bmatrix}$$ (10)

where $R_{e_1}$ and $t_{e_1}$ represent the orientation and position of the end-effector of mobile manipulator 1. Similarly, $R_{e_2}$ and $t_{e_2}$ represent the orientation and position of the end-effector of mobile manipulator 2.

**DESCRIPTION OF THE PROPOSED APPROACH**

This work introduces an approach to solve the inverse kinematics for cooperative mobile manipulators as a global constrained optimization. This section provides a detailed formulation of an objective function to solve the inverse kinematics, which includes penalty functions to handle constraints. The objective function is based on the forward kinematics equation of the cooperative system. It is proposed to use the self-adaptive differential evolution to solve the optimization problem. This section also provides a description of this variant of DE. Finally, a cooperative path tracking algorithm is presented based on the proposed inverse kinematics method.

**Objective function formulation**

Let consider the cooperative mobile manipulator system of Fig. 3. Mobile manipulator 1 is considered to have the role of master. Then, the end-effector position of frame $e_1$ is expressed with respect to the end-effector position of frame $e_2$ using a vector $t_r$. The vector $t_r$ defines the relative position between the end-effector position of mobile manipulators 1 and 2. In order to solve the inverse kinematics of the cooperative system, a desired position of the end-effector for the mobile manipulator with the role master must be provided. Then, the desired position for mobile manipulator 1 is defined as $t_{e_1}^*$. The desired position for mobile manipulator 2 is computed based on the definition of $t_r$ as follows

$$t_{e_2}^* = t_{e_1}^* + t_r$$ (11)

Let start by defining the actual joint configuration $x$ of the whole cooperative system. This vector is represented as $x = \begin{bmatrix} q_1^T \\ q_2^T \end{bmatrix}$ with $x \in \mathbb{R}^{n_1+n_2}$, where $n_1$ and $n_2$ are the total DOF of manipulator 1 and manipulator 2, respectively. Using the actual joint configuration $x$, the end-effector pose for both mobile manipulators can be computed with the forward kinematics of mobile manipulators defined in Eqs. (9) and (10). Then, the positions $t_{e_1}$ and $t_{e_2}$ can be used to define position errors for both mobile manipulators.

The error between a desired position $t_{e_1}^*$ and the actual position $t_{e_1}$ for the end-effector of mobile manipulator 1 can be computed as
The error between a desired position \( t_{e_2}^* \) and the actual position \( t_{e_2} \) for the end-effector of mobile manipulator 1 can be calculated as

\[
\text{error} = \| t_{e_2}^* - t_{e_2} \| \quad (12)
\]

where \( t_{e_2}^* \) is defined in Eq. (11).

To minimize the cooperative manipulation motion, an error between the actual joint configuration \( x \) and an initial joint configuration \( x_0 \) is defined. The vector \( x_0 \) can also represent a previous joint configuration. This error is defined as

\[
\text{error} = \| x - x_0 \| \quad (14)
\]

Then, the formulation of an objective function \( f \) which consider the position errors \( t_{1\text{error}} \) and \( t_{2\text{error}} \), and the error for the joint motion \( x_{\text{error}} \) is defined as

\[
f = \alpha (t_{1\text{error}} + t_{2\text{error}}) + \beta x_{\text{error}} \quad (15)
\]

where \( \alpha \) and \( \beta \) are factors that scale the contribution of each term.

The definition of the objective function (15) does not include constraints. The constraints are considered as the lower \( q_{l_1} \) and upper \( q_{u_1} \) joint limits for manipulator 1. Similarly, \( q_{l_2} \) and \( q_{u_2} \) define the lower and upper joint limits for manipulator 2, respectively. Let define the lower and upper joint limits for the cooperative system as

\[
x_l = [q_{l_1}^T, q_{l_2}^T]^T \text{ and } x_u = [q_{u_1}^T, q_{u_2}^T]^T
\]

with \( x_l, x_u \in \mathbb{R}^{n_1+n_2} \). A joint value \( x_j \) is considered feasible if \( x_l < x_j < x_u \) for each joint \( j = 1, 2, 3, \ldots, n_1 + n_2 \). To penalize unfeasible joint values, it is defined two penalty functions. The penalty function defined in Eq. (16) handles the constraints when \( x_j < x_l \). On the other hand, the penalty function given in Eq. (17) handles the constraints when \( x_j < x_u \).

\[
g(x_j) = \begin{cases} 0 & \text{if } x_l < x_j \\ (x_j - x_l)^2 & \text{otherwise} \end{cases} \quad (16)
\]

\[
h(x_j) = \begin{cases} 0 & \text{if } x_j < x_u \\ (x_u - x_j)^2 & \text{otherwise} \end{cases} \quad (17)
\]

Finally, the two defined penalty functions are included into Eq. (15) to obtain the proposed objective function \( f' \) that handles joint limits constraints. It is defined as

\[
f' = f + \sum_{j=1}^{n_1+n_2} \delta [g(x_j) + h(x_j)] \quad (18)
\]

where \( \delta \) scales the penalization term that is usually selected as a large constant. We recommend to use \( \delta = 1,000 \). In conclusion, the optimal joint variable \( x \) is computed by solving the constrained optimization problem defined as
\[
\min_{x} f'(x), \quad \text{subject to } x_l < x < x_u
\]

Coordinates manipulators are physically interacting with each other to perform a common task. In this work, the relative motion between two mobile manipulators must be adequately computed to perform the task. The proposed objective function considers the relative position between the end-effector position of the two mobile manipulators. The relative constraints between the two mobile manipulators are not considered in this work.

**Self-adaptive differential evolution**

The classical Differential Evolution (DE) is a population-based optimization algorithm (Storn & Price, 1997). The population is composed of individuals that represent potential solutions for a given global optimization problem. The DE algorithm performs three principal operations: mutation, crossover and selection, as a mechanism to improve the individuals during an iterative process called generations.

To begin, every individual \(x^G_i \in \mathbb{R}^D\) is generated randomly, where \(i = 1, 2, 3, \ldots, N\) with \(D\) as the dimension of the problem, \(N\) as the total numbers of population members, and \(G\) is the current generation. The mutant operation consists in the computation of a mutant vector \(v^G_{i}^{G+1}\) as follows

\[
v^G_{i}^{G+1} = x^G_{r_1} + F \left( x^G_{r_2} - x^G_{r_3} \right)
\]

where \(F \in [0, 2]\) is the amplification factor, and three individuals are chosen randomly \(x^G_{r_1}, x^G_{r_2}, x^G_{r_3}\) with \(r_1, r_2, r_3 \in \{1, N\}\) and \(i \neq r_1 \neq r_2 \neq r_3\).

In the crossover operation, a trial vector \(u^G_{i}^{G+1}\) is generated based on the following crossover scheme

\[
u^G_{j}^{G+1} = \begin{cases} 
  v^G_{j}^{G+1} & \text{if } r_j \leq C_R \text{ or } j = r_l \\
  x^G_{j} & \text{if } r_j > C_R \text{ and } j \neq r_l 
\end{cases}
\]

where \(C_R \in [0, 1]\) is the crossover constant with \(j = 1, 2, 3, \ldots, D\). The values \(r_j\) and \(r_l\) are random numbers computed as \(r_j \in [0, 1]\) and \(r_l \in \{1, D\}\). The value \(r_l\) ensures that the trial vector \(u^G_{i}^{G+1}\) gets at least one element of the mutant vector \(v^G_{i}^{G+1}\).

In the selection operation, the trial vector \(u^G_{i}^{G+1}\) is compared to the actual vector \(x^G_{i}\) with respect to the evaluation of an objective function \(f\). This operation is performed as

\[
x^G_{i}^{G+1} = \begin{cases} 
  u^G_{i}^{G+1} & \text{if } f(u^G_{i}^{G+1}) < f(x^G_{i}) \\
  x^G_{i} & \text{otherwise}
\end{cases}
\]

where, if \(f(u^G_{i}^{G+1})\) yields a better solution than \(f(x^G_{i})\), then \(u^G_{i}^{G+1}\) is considered as \(x^G_{i}^{G+1}\). On the other hand, \(x^G_{i}\) is retained to the next generation.

A new variant of DE is proposed in Fan, Xie & Zhou (2019). This variant is called self-adaptive mutation differential evolution, which includes a modified version to create the mutant vector \(v^G_{i}^{G+1}\). This mutation operation is given by
where \( x^G_b \) represents the population member with the best position, \( x^G_i \) is chosen randomly with \( r_4 \in \{1, N\} \) and \( i \neq r_1 \neq r_2 \neq r_3 \neq r_4 \). Moreover, \( G_{\text{max}} \) indicates the total number of generations and \( \text{rand} \in [0,1] \). The mutation operation based on \( x^G_b \) maintains the diversity of population for global exploration. The mutation based on \( x^G_i \) provides a mutation with local exploitation for a convergence accuracy and fast converge rate.

This new variant of DE proves to be superior than the classical DE algorithm. The self-adaptive mutation differential evolution is called SAMDECO by the authors (Fan, Xie & Zhou, 2019). In this work, SAMDECO is considered to be named as SDE for brevity.

Inverse kinematics for cooperative mobile manipulators

This section introduces an inverse kinematics algorithm for cooperative mobile manipulator based on the SDE algorithm. Moreover, the inverse kinematics algorithm is considered to solve cooperative path tracking tasks.

The aim of the inverse kinematic is to compute the optimal joint configuration \( x \) for the whole cooperative system, given the desired end-effector position \( t_{e_1} \) and the relative position \( t_r \). This relative position is used to compute the desired end-effector position \( t_{e_2} \), Eq. (11). Every individual in SDE represents a set of joint configuration \( x_i \in \mathbb{R}^D \) for the whole cooperative system, where \( D = n_1 + n_2 \) indicates the total DOF of the system. The initialization of each individual can be computed as

\[
x_{ji} = x_{lj} + (x_{uj} - x_{lj}) \, r
\]

where \( r \in [0,1] \) is a random number, \( i = 1,2,\ldots,N \) and \( j = 1,2,\ldots,D \). The joint limits \( x_l \) and \( x_u \) must be provided since the inverse kinematics is considered as a constrained problem. The SDE algorithm was not designed to work with constrained optimization. Initially, the lower and upper joint limits are used to initialize the population as feasible joint solutions. However, the initialization does not guarantee that the SDE algorithm will converge to a feasible solution. Joint limits must be considered during the optimization process to provide a valid joint solution.

The forward kinematics can be used to compute the matrix transformation \( ^wT_{e_1}(q_1) \) and \( ^wT_{e_2}(q_2) \) with Eqs. (9) and (10), for each individual \( x_i \). These matrices contain the position vectors \( t_{e_1} \) and \( t_{e_2} \) required to compute the position errors Eqs. (12) and (13). The vector \( x_0 \) is chosen as an initial joint configuration. In a path tracking task, this vector becomes the previous joint variable needed to minimize the joint motion, Eq. (14). At this point, the objective function can be evaluated using Eq. (18). The parameter setting for the objective function \( \alpha \) and \( \beta \) needs to be provided by the user. It is recommended to use \( \alpha = 1.5 \) and \( \beta = 0.6 \). This setting was determined by experimentation.

The SDE algorithm performs the mutation, crossover and selection operations to improve the individuals in each generation. The algorithm for the inverse kinematics of cooperative mobile manipulator based on SDE is given in Algorithm 1. The stop criteria is
met when the SDE algorithm reaches the total number of generations or when the evaluation of the fitness function reaches an allowed tolerance $s_{tol}$. The tolerance occurs when $s_{stop} = t_{e_1} + t_{e_2}$.

Respect to the cooperative path tracking tasks, the considered path is divided into $k = 1, 2, 3, \ldots, M$ points, with $M$ as the total number of points in the path. Then, the inverse kinematics for each point $p^*_i$ is solved with Algorithm 1. The description of the cooperative path tracking algorithm is given in Fig. 4.

### EXPERIMENTAL RESULTS

The aim of the experiments is to test the proposed approach for solving cooperative path tracking tasks. A cooperative system with two mobile manipulators was considered for experiments. Moreover, a comparative study is included to compare the performance of CFPSO, FPA, SDE and KABC for the solution of the given path tracking tasks.

The considered mobile manipulator is based on the KUKA Youbot (López-Franco et al., 2018). It is conformed by a manipulator of five DOF, and an omnidirectional mobile platform of three DOF. With respect to the mobile manipulator kinematics, the transformation $^wT_b$ can be computed with the mobile platform pose, which is given by $x_b$, $y_b$ and $\theta_b$. The constant transformation $^bT_m$ is considered to be

$$
^bT_m = \begin{bmatrix}
1 & 0 & 0 & 0.140 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0.151 \\
0 & 0 & 0 & 1
\end{bmatrix}
$$

these values were obtained based on the KUKA Youbot technical specifications.
The DH table and the coordinate frame assignment for the KUKA Youbot are given in Table 1 and Fig. 5, respectively.

The DH table is used to compute the transformation \( ^{w}T_{e_1} \). Moreover, the next homogeneous matrices represent the manipulator kinematics model

\[
0^T_{1}(\theta_1) = \begin{bmatrix}
\cos(\theta_1) & 0 & \sin(\theta_1) & 0.033 \cos(\theta_1) \\
\sin(\theta_1) & 0 & -\cos(\theta_1) & 0.033 \sin(\theta_1) \\
0 & 1 & 0 & 0.147 \\
0 & 0 & 1 & 1
\end{bmatrix}
\]

Figure 4 Cooperative path tracking algorithm. DOI: 10.7717/peerj-cs.419/fig-4
\[ T_2(\theta_2) = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & 0.155\cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2) & 0 & 0.155\sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

\[ T_3(\theta_3) = \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & 0.135\cos(\theta_3) \\ \sin(\theta_3) & \cos(\theta_3) & 0 & 0.135\sin(\theta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

\[ T_4(\theta_4) = \begin{bmatrix} \cos(\theta_4) & 0 & \sin(\theta_4) & 0 \\ \sin(\theta_4) & 0 & -\cos(\theta_4) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]
where the transformation \( m^T \) is computed as
\[
m^T = 0^T (\theta_1) T^1(\theta_2) T^2(\theta_3) T^3(\theta_4) 4^T(\theta_5)
\] (24)

The joint variable \( q \) for the mobile manipulator is
\[
q = [x_b \ y_b \ \theta_b \ \theta_1 \ \theta_2 \ \theta_3 \ \theta_4 \ \theta_5]^T
\]
where the joint values \( \theta_1 - \theta_5 \) represent the joint configuration of the manipulator. Then, it is proposed to use \( q_1 \) and \( q_2 \) to represent the joint variable for mobile manipulator 1 and 2, respectively. The joint configuration for the whole cooperative system is given by
\[
x = [q_1^T \ q_2^T]^T.
\]

The joint limits were obtained based on the KUKA Youbot technical specifications. The lower joint limits for the mobile manipulator 1 (\( q_{1l} \)) and mobile manipulator 2 (\( q_{2l} \)) are set to
\[
q_{1l} = q_{2l} = [-1.5 - 1.5 - 3.1416 - 2.9496 - 1.1345 - 2.6180 - 1.7837 - 2.9234]^T
\]
where the lower joint limits for the cooperative system is \( x_l = [q_{1l}^T \ q_{2l}^T]^T \). Similarly, the upper joint limits for the mobile manipulator 1 (\( q_{1u} \)) and mobile manipulator 2 (\( q_{2u} \)) are set to
\[
q_{1u} = q_{2u} = [1.5 1.5 3.1416 2.9496 1.5708 2.5482 1.7890 2.9234]^T
\]
where the upper joint limits for the cooperative system is \( x_u = [q_{1u}^T \ q_{2u}^T]^T \).

Four trajectories with different degrees of difficulty are considered for testing the proposed approach. Each trajectory is divided into \( k = 1, 2, 3, \ldots, M \) points, where \( M \) is the total number of points in the path. The vector \( t_k^T = [x_k \ y_k \ z_k]^T \) defines the \( k \)th trajectory point, and it is used as a target for the cooperative inverse kinematics. The considered trajectories are provided below

**Trajectory 1 : Sinusoidal**
\[
x_k = 0.5
\]
\[
z_k = 0.4 + 0.05 \sin(30 y_k)
\]
\[
y_k \in [0.5, 1.5]
\]

**Trajectory 2 : Circular**
\[
x_k = 0.5
\]
\[
y_k = 0.05 \cos(\theta_k)
\]
\[
z_k = 0.4 + 0.05 \sin(\theta_k)
\]
\[
\theta_k \in [0, 2\pi]
\]
Trajectory 3 : Trapezoidal
\[ x_k = 0.5 \]
\[ r_k = 0.4 + 0.1 \sin(30 \, y_k) \]
\[ y_k \in [0.5, 1.5] \]
\[ z_k = \begin{cases} 
0.45 & \text{if } r_k > 0.45 \\
0.35 & \text{if } r_k < 0.35 \\
r_k & \text{otherwise}
\end{cases} \]

Trajectory 4 : Rosecurve
\[ x_k = 0.5 \]
\[ y_k = r_k \cos(\theta_k) \]
\[ z_k = 0.4 + r_k \sin(\theta_k) \]
\[ r_k = 0.035 + 0.015 \cos(3 \, \theta_k) \]
\[ \theta_k \in [0, 2\pi] \]

where a total of \( M = 200 \) points were selected. Moreover, the relative vector for the cooperative tasks is set to \( t_r = [0 \quad -1 \quad 0]^T \), and the initial joint configuration is defined as
\[ x_0 = [0 \quad 0.5 \quad \pi/4 \quad 0 \quad \pi/2 \quad -\pi/4 \quad \pi/4 \quad 0 \quad -0.5 \quad \pi/4 \quad 0 \quad \pi/2 \quad -\pi/4 \quad \pi/4 \quad 0]^T \]

The parameter setting for the considered meta-heuristics algorithms is conducted as follows: The particular parameter setting of CFPSO are the cognitive factor \( C_1 = 2.05 \), the social factor \( C_2 = 2.05 \), and \( \phi = 4.1 \) for the constriction factor, see Collinsm & Shen (2017). The settings for FPA are the switch probability \( P = 0.8 \) for local and global pollination and a step size \( \lambda = 1.5 \) for strength of the pollination, see recommendation in Yang (2012). In the SDE algorithm, it was considered to use the amplification factor \( F = 0.5 \) and crossover constant \( C_R = 0.8 \) based on Fan, Xie & Zhou (2019). The particular parameter setting for KABC are the forager bees population \( P_f = 20 \), the onlooker bees population \( P_o = N - P_f \), and stagnation limit parameter \( L = (N * D)/2 \), see Karaboga & Basturk (2007) and El-Sherbiny, Elhosseini & Haikal (2018). With respect to the common parameter settings, it was considered to use a population size of \( N = 30 \) individuals, a total of \( G_{\text{max}} = 1,000 \) generations, and a tolerance of \( s_{\text{stop}} = 1 \times 10^{-4} \).

The simulations are presented in two parts. In the first part, the comparative study is presented. To test the performances of the considered meta-heuristics algorithms, the inverse kinematics results for each point \( k \) in the path were used to compare the statistical variations. The position error \( t_{\text{error}} + t_{\text{error}} \) and the motion error \( x_{\text{error}} \) are reported. Additionally, the execution time in seconds is also reported, where the specifications of the test machine are Intel Core i7-4770 CPU 3.4 GHz and 16.0 GB of RAM. To display graphically the statistical variation for the results, box plots were used. The best algorithms will show the smaller data distribution, the lower values results, and the less quantities of outliers. In the second part, it is considered to run multiple experiments on every trajectory to verify the robustness of the SDE algorithm because the mobile manipulator is redundant and there is more than one set of inverse solutions.

The results of the first part of the simulations are shown below. The position error results for the cooperative path tracking are presented in Fig. 6. The CFPSO algorithm
reports the largest data distribution in all trajectories. In contrast, the SDE algorithm shows the smallest data distribution with the lowest value results. The FPA and KABC algorithms have similar data distribution, with better value results than CFPSO. However, KABC performs better than FPA with lower value results. In general, the SDE algorithm presents more consistent results with precision. On the other hand, the CFPSO, FPA and KABC algorithms present outliers results and lower precision.

The motion error results for the cooperative path tracking are illustrated in Fig. 7. It is crucial to mention that the more motion error, the more joint changes are presented in the path following. A big motion error produces abrupt changes in the joint motion. The SDE algorithm reported smaller data distribution with slower motion errors. These results are precise and a mean error value of 0.1033 is obtained, in general. In contrast, the CFPSO algorithm presents the larger data distribution with a mean motion error value of 1.6296. This is not a reliable result due to the presence of abrupt changes in the joint motion. The FPA and KABC performed similarly, with mean error values of 0.0959 and 0.0142, respectively. Finally, the CFPSO and KABC contain more quantity of outliers than the other algorithms.

The execution time results for the cooperative path tracking are reported in Fig. 8. The SDE algorithm outperformed the other with the smaller execution time results with a
mean value of 1.5828 s in general. The CFPSO and FPA algorithms have similar time results, with mean values of 4.9043 s and 4.4768 s. However, the KABC algorithm performed better than CFPSO and FPA, with a mean value of 3.2938 s. The position error results in Fig. 6, suggests that most of the inverse kinematics results did not reach the allowed tolerance, and CFPSO, FPA and KABC reached the total number of iterations. For this reason, the execution time is bigger than SDE. In this case, all compared algorithms have similar data distribution, and the reported data shows outliers, which indicates that execution time is not precise.

Additionally, Table 2 shows the root mean square (RMS) results for the cooperative path tracking tasks. The table contains the position error results for both mobile manipulator position errors $t_1^{\text{error}}$ and $t_2^{\text{error}}$. The SDE algorithm outperformed the others with better accuracy RMS results, with values below to $1 \times 10^{-4}$ which is the tolerance allowed. The RMS values of the other algorithm did not reach the tolerance. The KABC performed better than FPA with lower RMS values, and CFPSO obtained the worst RMS results.

Based on the results given so far, the SDE and KABC performed better than the other algorithms. For this reason, the path tracking and joint motion results are displayed graphically for comparison. First, the path tracking results for the cooperative tasks using the KABC algorithm are presented in Fig. 9. The path tracking results reported are not
consistent, and the position error results have low precision. The statistical variation of position error during the path tracking is indicated with large data distribution, see Fig. 6.

Moreover, the Fig. 10 reports the path tracking results using the SDE algorithm. In this case, the reported results indicate that SDE has precise and accurate path tracking values. The desired end-effector position of manipulator \( i \) and the obtained position are considered identically. The statistical variation of position error during the path tracking is represented by a small data distribution and low value results, see Fig. 6.

**Table 2** RMS values for the position error \( t_{\text{error}} \) results of mobile manipulator \( i \). The best results are highlighted in bold.

| Trajectory | CFPSO   | FPA     | SDE      | KABC     |
|------------|---------|---------|----------|----------|
| 1          | \( t_{1\text{error}} \) (m) | 4.9927 \( \times 10^{-2} \) | 7.6444 \( \times 10^{-3} \) | \( 6.8059 \times 10^{-5} \) | 5.2494 \( \times 10^{-3} \) |
| 2          | \( t_{2\text{error}} \) (m) | 4.8456 \( \times 10^{-2} \) | 7.3751 \( \times 10^{-3} \) | \( 6.8210 \times 10^{-5} \) | 3.5065 \( \times 10^{-3} \) |
| 3          | \( t_{3\text{error}} \) (m) | 5.4808 \( \times 10^{-2} \) | 9.5702 \( \times 10^{-3} \) | \( 6.3632 \times 10^{-5} \) | 2.1096 \( \times 10^{-3} \) |
| 4          | \( t_{4\text{error}} \) (m) | 4.9342 \( \times 10^{-2} \) | 8.2215 \( \times 10^{-3} \) | \( 6.7407 \times 10^{-5} \) | 3.9660 \( \times 10^{-3} \) |
| 5          | \( t_{5\text{error}} \) (m) | 4.3967 \( \times 10^{-2} \) | 8.9793 \( \times 10^{-3} \) | \( 6.4637 \times 10^{-5} \) | 7.6678 \( \times 10^{-3} \) |
| 6          | \( t_{6\text{error}} \) (m) | 4.8983 \( \times 10^{-2} \) | 7.5944 \( \times 10^{-3} \) | \( 6.6192 \times 10^{-5} \) | 2.5669 \( \times 10^{-3} \) |
| 7          | \( t_{7\text{error}} \) (m) | 3.9981 \( \times 10^{-2} \) | 8.0709 \( \times 10^{-3} \) | \( 6.5240 \times 10^{-5} \) | 2.0876 \( \times 10^{-3} \) |
Figure 9  Path tracking results for the cooperative tasks based on the KABC algorithm. The vector $t^*$ represents the desired end-effector position, and vector $t$ is the obtained position. (A–D) The results for mobile manipulator 1. (E–H) The results for mobile manipulator 2.
The legend $t^*$ represents the desired end-effector position, and $t$ is the obtained position. (A–D) The results for mobile manipulator 1. (E–H) The results for mobile manipulator 2.
These results indicate that SDE outperformed the KABC algorithm concerning better path tracking results.

The joint motion results obtained by the KABC algorithm are illustrated in Fig. 11. In this case, the joint values during the cooperative tasks present a rough motion. The statistical variation of the motion error during the path tracking indicates large data distribution with big value results see Fig. 7. It is important to notice that rough motion may cause hardware failure.

Figure 12 shows the joint motion results obtained by the SDE algorithm. In this case, the joint values obtained during the cooperative task reports smooth motion results. The statistical variation of the motion error during the path tracking is represented by a small data distribution with low value results see Fig. 7. The SDE algorithm obtains reliable motion results compared to the KABC algorithm.

Considering the results reported in this experimental section, the SDE algorithm outperformed the CFPSO, FPA and KABC algorithms, in general. The SDE algorithms reported better position and motion error results with a lower execution time. Additionally, path tracking and motion results validate the precision and accuracy values provided by SDE. The CFPSO algorithms performed poorly in all tests, and the FPA and KABC algorithms performed similarly in general. The SDE algorithm proves to be more adequate to solve the inverse kinematics for cooperative mobile manipulators than the compared algorithms.

The results of the second part of the simulations are shown below. In this case, the initial joint is defined as

\[
x_0 = [0.3 \ 0.5 \ \pi/4 \ 0 \ \pi/2 \ 0 \ \pi/2 \ \pi/4 \ 0.3 \ -0.5 \ -\pi/4 \ 0 \ \pi/2 \ 0 \ \pi/2 \ -\pi/4]^T
\]

Moreover, the SDE algorithm runs 25 times, and the set of joint solutions are shown graphically. The rest of the experimental setup is the same as the previous experiments.

Figure 13 shows the results of the multiple joint motion results for every considered trajectory. As can be seen, multiples joint motion results are achieved. The joint motion results are very similar because the proposed objective function considers the minimum error between the current and the previous joint configuration. Moreover, the joint solutions for the first point on the trajectory will depend on the initial joint values \(x_0\). If a different set of joint solutions is required, then the initial joint values must be different.

One of the advantages of the SDE algorithm is its mutation mechanism that provides a balance between the exploration and exploitation characteristics. This improves the performance of SDE with respect to convergence accuracy and fast converge rate. The experimental results indicate that thanks to these characteristics, the SDE algorithm has the lowest execution time, and the lowest position and displacement errors. In addition, the statistical analysis demonstrates the consistency of the results obtained with the smallest data distribution. The SDE algorithm also shows that it can reach the allowed tolerance while the other compared algorithms are not. All these results guarantee the convergence to the global optimum value and the computation of smooth trajectories.
Figure 11 Joint motion results based on the KABC algorithm. (A–D) The results for mobile manipulator 1. (E–H) The results for mobile manipulator 2.

DOI: 10.7717/peerj-cs.419/fig-11
Figure 12 Joint motion results based on the SDE algorithm. (A–D) The results for mobile manipulator 1. (E–H) The results for mobile manipulator 2.
Figure 13  Multiples joint motion results based on the SDE algorithm. (A–D) The results for mobile manipulator 1. (E–H) The results for mobile manipulator 2.

Hernandez-Barragan et al. (2021), PeerJ Comput. Sci., DOI 10.7717/peerj-cs.419/fig-13
CONCLUSIONS
This work introduced an approach to solving the inverse kinematics for cooperative mobile manipulators based on self-adaptive differential evolution (SDE). Simulation experiments were performed to prove the effectiveness of the proposed approach for solving cooperative path tracking tasks. It was considered to include four trajectories with different degrees of difficulty. The simulations also included a comparison among CFPSO, FPA, SDE, and KABC algorithms. The inverse kinematics results were used to compare the algorithms statistically. The SDE algorithm outperformed the other with consistent statistical results, such as the smallest data distribution and the lowest value results. The execution time of SDE also is the best, with a mean value of 1.5828 seconds. Moreover, the path tracking errors of SDE reported precise and accurate results, with RMS position error results below 0.1 mm. Additionally, the motion error results of SDE reported smooth joint displacement during path tracking. The FPA and KABC algorithms performed similarly. They reported position error results near 1 mm. However, their statistical results with respect to the motion error results present large data distribution. This inconvenience is observed with rough joint displacements. The CFPSO performed poorly in general, its position error result is greater than 10 mm, and motion error results indicate abrupt changes in the joint displacement. The CFPSO algorithm is not recommended to solve the inverse kinematics for cooperative mobile manipulators. The experimental setup considered two mobile manipulators based on the KUKA Youbot system, which is conformed by an omnidirectional mobile platform with three DOF, and a manipulator with five DOF. However, the proposed approach can be used to solve the inverse kinematics for cooperative omnidirectional mobile manipulators with \( n \) DOF. As future research, the proposed approach can be extended to solve cooperative dual-arm systems attached to the same mobile platform. It is also appealing to include cooperative mobile manipulators with nonholonomic mobile platforms.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding
This work was supported by Council of Sciences and Technology (CONACYT), Mexico, through the following projects: CB-256769, CB-258068 and PN-2016-4107. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Grant Disclosures
The following grant information was disclosed by the authors:
Council of Sciences and Technology (CONACYT), Mexico: CB-256769, CB-258068 and PN-2016-4107.

Competing Interests
Alma Yolanda Alanis Garcia is an Academic Editor for PeerJ.
Author Contributions

- Jesus Hernandez-Barragan performed the experiments, performed the computation work, prepared figures and/or tables, and approved the final draft.
- Carlos Lopez-Franco conceived and designed the experiments, performed the computation work, prepared figures and/or tables, supervision and project administration, and approved the final draft.
- Nancy Arana-Daniel conceived and designed the experiments, analyzed the data, authored or reviewed drafts of the paper, supervision and project administration, and approved the final draft.
- Alma Y. Alanis conceived and designed the experiments, analyzed the data, authored or reviewed drafts of the paper, supervision and project administration, and approved the final draft.

Data Availability

The following information was supplied regarding data availability:

Data and code are available at GitHub: https://github.com/JesusHernandezBarragan/PeerJ—Inverse-kinematics-for-cooperative-mobile-manipulators-based-on-self-adaptive-differential-e.git.

REFERENCES

Ayyldz M, Çetinkaya K. 2016. Comparison of four different heuristic optimization algorithms for the inverse kinematics solution of a real 4-dof serial robot manipulator. Neural Computing and Applications 27(4):825–836 DOI 10.1007/s00521-015-1898-8.

Chen J, Kai S. 2018. Cooperative transportation control of multiple mobile manipulators through distributed optimization. Science China Information Sciences 61(12):79 DOI 10.1007/s11432-018-9588-0.

Collinsm TJ, Shen W. 2017. Particle swarm optimization for high-DoF inverse kinematics. In: 2017 3rd International Conference on Control, Automation and Robotics (ICCAR), 1–6.

Craig J. 2005. Introduction to robotics mechanics and control. Third Edition. London: Pearson Education.

El-Sherbiny A, Elhosseini MA, Haikal AY. 2018. A new ABC variant for solving inverse kinematics problem in 5 DoF robot arm. Applied Soft Computing 73(4):24–38 DOI 10.1016/j.asoc.2018.08.028.

Fan S, Xie X, Zhou X. 2019. Optimum manipulator path generation based on improved differential evolution constrained optimization algorithm. International Journal of Advanced Robotic Systems 16(5):1729881419872060.

Freddi A, Longhi S, Monteriù A, Ortenzi D. 2016. Redundancy analysis of cooperative dual-arm manipulators. International Journal of Advanced Robotic Systems 13(5):1729881416657754 DOI 10.1177/1729881416657754.

Hu Y, Huang B, Yang G-Z. 2015. Task-priority redundancy resolution for co-operative control under task conflicts and joint constraints. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Piscataway: IEEE, 2398–2405.

Jamisola RS, Roberts RG. 2015. A more compact expression of relative Jacobian based on individual manipulator Jacobians. Robotics and Autonomous Systems 63(7):158–164 DOI 10.1016/j.robot.2014.08.011.
Karaboga D, Basturk B. 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization* 39(3):459–471 DOI 10.1007/s10898-007-9149-x.

Korayem M, Dehkordi S. 2019a. Dynamic modeling of flexible cooperative mobile manipulator with revolute-prismatic joints for the purpose of moving common object with closed kinematic chain using the recursive Gibbs–Appell formulation. *Mechanism and Machine Theory* 137:254–279 DOI 10.1016/j.mechmachtheory.2019.03.026.

Korayem MH, Dehkordi SF. 2019b. Motion equations of cooperative multi flexible mobile manipulator via recursive Gibbs–Appell formulation. *Applied Mathematical Modelling* 65(2):443–463 DOI 10.1016/j.apm.2018.08.035.

Kumar A, Banga VK, Kumar D, Yingthawornsuk T. 2019. Kinematics solution using metaheuristic algorithms. In: 2019 15th International Conference on Signal-Image Technology Internet-Based Systems (SITIS). 505–510.

Kundu AS, Mazumder O, Dhar A, Lenka PK, Bhaumik S. 2017. Scanning camera and augmented reality based localization of omnidirectional robot for indoor application. *Procedia Computer Science* 105:27–33.

Lewis CL. 1996. Trajectory generation for two robots cooperating to perform a task. *Proceedings of IEEE International Conference on Robotics and Automation* 2:1626–1631.

Li Z, Yang C, Su C, Deng J, Zhang W. 2016. Vision-based model predictive control for steering of a nonholonomic mobile robot. *IEEE Transactions on Control Systems Technology* 24(2):553–564.

Low ES, Ong P, Cheah KC. 2019. Solving the optimal path planning of a mobile robot using improved q-learning. *Robotics and Autonomous Systems* 115(3):143–161 DOI 10.1016/j.robot.2019.02.013.

López-Franco C, Hernández-Barragán J, Alanis AY, Arana-Daniel N, López-Franco M. 2018. Inverse kinematics of mobile manipulators based on differential evolution. *International Journal of Advanced Robotic Systems* 15(1):1729881417752738.

Mishra A, Deb S. 2020. Mobile robot path planning using a flower pollination algorithm-based approach. Singapore: Springer Singapore, 127–147.

Molina MF, Ortiz JS. 2018. Coordinated and cooperative control of heterogeneous mobile manipulators. In: Ge SS, Cabibihan J-J, Salichs MA, Broadbent E, He H, Wagner AR, Castro-González Á, eds. *Social Robotics*. Cham: Springer International Publishing, 483–492.

Mora-Aguilar J, Carvajal CP, Sánchez JS, Andaluz VH. 2018. Cooperative control of sliding mode for mobile manipulators. In: Ge SS, Cabibihan J-J, Salichs MA, Broadbent E, He H, Wagner AR, Castro-González Á, eds. *Social Robotics*. Cham: Springer International Publishing, 253–264.

Ortiz JS, Molina MF, Andaluz VH, Varela J, Morales V. 2018. Coordinated control of an omnidirectional double mobile manipulator. In: Kim KJ, Kim H, Baek N, eds. *IT Convergence and Security 2017*. Singapore: Springer Singapore, 278–286.

Pajak I. 2019. Real-time trajectory generation methods for cooperating mobile manipulators subject to state and control constraints. *Journal of Intelligent and Robotic Systems* 93(3–4):649–668 DOI 10.1007/s10846-018-0878-5.

Ren Y, Sosnowski S, Hirche S. 2020. Fully distributed cooperation for networked uncertain mobile manipulators. *IEEE Transactions on Robotics* 36(4):984–1003 DOI 10.1109/TRO.2020.2971416.

Savsani P, Jhala RL, Savsani VJ. 2016. Comparative study of different metaheuristics for the trajectory planning of a robotic arm. *IEEE Systems Journal* 10(2):697–708 DOI 10.1109/JSYST.2014.2342292.
Šegota SB, Andelić N, Lorencin I, Saga M, Car Z. 2020. Path planning optimization of six-degree-of-freedom robotic manipulators using evolutionary algorithms. *International Journal of Advanced Robotic Systems* 17(2):1729881420908076.

Siciliano B, Sciavicco L, Villani L, Oriolo G. 2008. *Robotics—modelling, planning and control: advanced textbooks in control and signal processing*. Second Edition. New York: Springer.

Smith C, Karayiannidis Y, Nalpantidis L, Gratal X, Qi P, Dimarogonas DV, Kragic D. 2012. Dual arm manipulation—a survey. *Robotics and Autonomous Systems* 60(10):1340–1353 DOI 10.1016/j.robot.2012.07.005.

Spong MW, Vidyasagar M. 2008. *Robot dynamics and control*. Hoboken: John Wiley & Sons.

Storn R, Price K. 1997. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* 11(4):341–359 DOI 10.1023/A:1008202821328.

Tallamraju R, Salunkhe DH, Rajappa S, Ahmad A, Karlapalem K, Shah SV. 2019. Motion planning for multi-mobile-manipulator payload transport systems. In: 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE). 1469–1474.

Tallamraju R, Sripada V, Shah SV. 2019. Path planning through tight spaces for payload transportation using multiple mobile manipulators. In: 2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN). Piscataway: IEEE, 1–6.

Wang M, Luo J, Yuan J, Walter U. 2018. Coordinated trajectory planning of dual-arm space robot using constrained particle swarm optimization. *Acta Astronautica* 146(4):259–272 DOI 10.1016/j.actaastro.2018.03.012.

Wu J, Lv C, Zhao L, Li R, Wang G. 2017. Design and implementation of an omnidirectional mobile robot platform with unified i/o interfaces. In: 2017 IEEE International Conference on Mechatronics and Automation (ICMA). Piscataway: IEEE, 410–415.

Wu M, He Y, Liu S. 2016. Collaboration of multiple mobile manipulators with compliance based leader/follower approach. In: 2016 IEEE International Conference on Industrial Technology (ICIT). Piscataway: IEEE, 48–53.

Yang M, Yang E, Zante RC, Post M, Liu X. 2019. Collaborative mobile industrial manipulator: a review of system architecture and applications. In: 2019 25th International Conference on Automation and Computing (ICAC). 1–6.

Yang X-S. 2012. Flower pollination algorithm for global optimization. *Lecture Notes in Computer Science* 7445:240–249.

Ziwu R, Chunguang L, Lining S. 2016. Minimum-acceleration trajectory optimization for humanoid manipulator based on differential evolution. *International Journal of Advanced Robotic Systems* 13(2):73 DOI 10.5772/63070.