Particle Swarm Optimization for High-DOF Inverse Kinematics

Thomas Joseph Collins and Wei-Min Shen
Polymorphic Robotics Laboratory, Department of Computer Science
University of Southern California
Los Angeles, USA
e-mail: collinst@usc.edu, shen@isi.edu

Abstract—The inverse kinematics (IK) problem is a fundamental problem in robotic manipulation. Traditional, Jacobian-based solutions to this problem are known to scale poorly with the number of degrees of freedom (DOF) in the manipulator, necessitating novel IK solutions for high-DOF manipulators. Metaheuristic optimization algorithms such as Particle Swarm Optimization (PSO) are a promising alternative approach to traditional IK techniques due to their strong performance on difficult and high-DOF problems in many diverse domains. Previous applications of PSO to the IK problem have focused on specific classes (e.g., planar) or models of manipulators or specific IK subproblems (e.g., position only IK). Furthermore, the experimental validation of these techniques has considered only manipulators with seven or fewer degrees of freedom and taken place almost exclusively in simulation. In this paper, we (1) generalize previous work to derive a fitness function that can be minimized to solve the full position and orientation IK problem for any serial manipulator while respecting joint limits and avoiding self-collisions, (2) present the first statistical analysis of PSO as a high-DOF IK solver on simulated manipulators with up to 180 DOF using this fitness function, and (3) present an important validation of PSO-based IK using this fitness function on real-world hardware on a difficult precision manipulation task.

Keywords—Inverse Kinematics, Self-reconfigurable Robots, Particle Swarm Optimization

I. INTRODUCTION

The inverse kinematics (IK) problem of a serial manipulator – the problem of finding a set of joint angles that aligns the manipulator's end-effector with a target pose – is a fundamental problem in robotic manipulation. Traditional, Jacobian-based techniques for solving this problem are known to scale poorly with the number of degrees of freedom (DOF) in the manipulator and have well-documented numerical instabilities that are amplified by increases in manipulator DOF [1]. Thus, novel IK solution methodologies are necessary for high-DOF manipulators, such as those encountered in self-reconfigurable robotic systems (e.g., [2]).

Metaheuristic optimization techniques, such as Particle Swarm Optimization [3] are a promising alternative to traditional manipulation techniques for high-DOF IK due to their numerical stability and strong performance on difficult and high-DOF problems in many diverse domains. Previous applications of PSO to the IK problem have presented algorithms applicable only to specific classes (e.g., planar) or models of manipulators or solved only position IK, ignoring the difficulty of minimizing the position and orientation error between end-effector and target poses simultaneously.

Additionally, experimental validation has considered only manipulators with seven or fewer degrees of freedom and has taken place almost exclusively in simulation.

In this paper, we generalize previous work to derive a fitness function that, when minimized by any PSO variant, solves the full position and orientation IK problem of any serial manipulator while respecting joint limits and avoiding self-collisions, a crucial problem for high-DOF manipulators that has not been considered in the general case in previous studies. We present the first statistical analysis of PSO as a high-DOF IK solver using this fitness function. Importantly, we consider how the solution quality and runtime of two popular PSO variants – Bare Bones PSO [4] and Constriction factor PSO [5] – scale against increasingly large simulated manipulators in 3D workspaces with up to 180 DOF. As further validation, we present the results of successfully applying PSO-based IK using this fitness function to real-world hardware on a difficult precision manipulation task – autonomous docking in self-reconfigurable robotics. To the best of our knowledge, this represents the first hardware-based validation of a PSO-based IK solver on a precision manipulation task.

Section II discusses related work and background knowledge on Particle Swarm Optimization and the variants evaluated in this paper. Section III derives our general-purpose PSO IK fitness function. Section IV presents the results of our statistical analysis of PSO as a high-DOF IK solver as well as the results of applying the approach on real-world robotic hardware. Section V concludes with future work.

II. BACKGROUND

A. Related Work

Several studies have evaluated the performance of PSO as an IK solver for specific types or classes of robotic manipulators, including [6], which used PSO to solve the position IK of a particular 6-DOF manipulator in simulation, [7], [8], which used PSO-based IK in both simulation and on real-world hardware to compute 3-DOF leg poses for balancing planar biped walkers during locomotion, and [9], which used a hybrid simulated annealing and PSO approach to solve the IK problem of a particular 6-DOF manipulator in simulation. In [10], the authors present a PSO-based IK solution applicable to 7-DOF manipulators which is validated in simulation.

In [11], the authors present a statistical analysis of the performance of several notable PSO variants on the inverse kinematics problem of a simulated planar, two-link, two-joint...
The basic idea of Particle Swarm Optimization (PSO) is that a swarm of \( m \) particles, each \( n \)-dimensional, perform an iterative randomized search in the space of possible \( n \)-dimensional solution vectors while communicating with one another to share information about promising areas of the search space toward which future search will be biased. Each particle \( i \) is a point \( x_i \) in this search space with a certain velocity \( v_i \) and associated fitness given by the objective function value at that point \( f = F(x_i) \), where \( F \) is the function to be minimized. Particles move around in this search space randomly but particle movement is biased toward a direction to the best (lowest \( f \)) position achieved by any particle in the swarm \( g \) (social component) and the direction to the best position achieved by each particle individually, \( p_i \) (history component). This focuses random searches on areas of the search space where an optimum is expected to be. \( g \) and \( p_i \) are updated at each iteration. This randomized searching process continues until a certain fitness threshold \( h \) is reached by some particle in the swarm (i.e., a low enough value of \( F \) is found) or a maximum number of iterations \( N \) is performed. At termination, the global best position found \( (g) \) is returned.

At each iteration \( t + 1 \), every dimension \( j = 1, 2, ..., n \) of particle \( i \) is updated according to the following two equations in traditional PSO:

\[
\begin{align*}
    v_{i,t+1}^j &= v_{i,t}^j + c_1 R_{1,t}^i (p_{i,t}^j - x_{i,t}^j) + c_2 R_{2,t}^i (g_{t}^j - x_{i,t}^j) \quad (1) \\
    x_{i,t+1}^j &= x_{i,t}^j + v_{i,t+1}^j \quad (2)
\end{align*}
\]

In Equation 1, \( c_1 \) and \( c_2 \) are constants (algorithm parameters), \( R_{1,t}^i \sim U(0,1) \) and \( R_{2,t}^i \sim U(0,1) \). Particle positions are often clipped to keep them within certain bounds.

A number of variants of PSO exist, many of which modify the way in which particle velocities are updated in order to ensure convergence of the swarm to an optimum of \( F \). One such popular variant is Constriction factor PSO (PSO-Co) [5], which utilizes the following equation in place of Equation 1:

\[
\begin{align*}
    x_{i,t+1}^j &= (\chi v_{i,t}^j + c_1 R_{1,t}^i (p_{i,t}^j - x_{i,t}^j) + c_2 R_{2,t}^i (g_{t}^j - x_{i,t}^j)) \\
    \chi &= \frac{2}{2 - \phi - \sqrt{\phi^2 - 4\phi}} \quad \text{and} \quad \phi = c_1 + c_2 \quad \text{. Clerc and Kennedy [14] showed that the swarm demonstrates stable convergence when } \phi > 4, \text{ recommending a value of } \phi = 4.1, \quad \text{which leads to } c_1 = c_2 = 2.05 \quad \text{and } \chi = 0.7298. \\
\end{align*}
\]

Probabilistic variants of PSO also exist, many of which forgo the use of a velocity vector, instead sampling the next particle position directly from the current one according to a particular distribution. Bare Bones PSO (BB-PSO) [4] is one important example of this class of probabilistic PSO algorithms. At each iteration, the next particle position is sampled according to the following equation, which takes the place of Equations 1 and 2 in traditional PSO:

\[
\begin{align*}
    x_{i,t+1}^j &\sim N\left(\frac{p_{i,t}^j + g_{t}^j}{2}, \left(\frac{p_{i,t}^j - g_{t}^j}{2}\right)^2\right) \quad (4)
\end{align*}
\]

In words, the \( j \)-th component of particle \( i \) at time \( t + 1 \) is sampled according to a normal distribution whose mean is the
average of the $j$-th component of particle $i$'s personal best position at time $t$ and the $j$-th component of the global best swarm position at time $t$. The variance of this distribution is the square difference between the $j$-th components of these personal and global best positions at time $t$. These two popular variants of PSO represent two important classes of PSO algorithms: velocity-based and probabilistic, which is why they were chosen for comparison.

### III. PSO Inverse Kinematics

Let $M$ be a serial manipulator with $n$ joints. $M$'s configuration is an assignment of valid angles to each of these joints $\mathbf{q} = \{q_0, q_1, \ldots, q_{n-1}\}$, where, without loss of generality, $q_0$ is the joint closest to the base of the manipulator and $q_{n-1}$ is the joint closest to the end-effector. The configuration space of the manipulator is the set of all possible valid assignments of values to these $n$ joint angle variables. Each variable $q_i$ can take only values in a certain range $\text{limits}_i = [q_{i,\text{low}}, q_{i,\text{high}}]$ due to physical limitations of the manipulator. The workspace of $M$ is the set of poses achieved by $M$’s end-effector under all possible manipulator configurations. A pose $\mathbf{E} = (P_E, O_E)$ is the specification of the position and orientation of a body in space – in this case, the end-effector of the manipulator – where $P_E$ is the position coordinates of $E$ and $O_E$ are the orientation coordinates of $E$.

The forward kinematics problem is that of finding the end-effector pose $\mathbf{E} = (P_E, O_E) = \mathbf{K}(\mathbf{q}_E)$ corresponding to some manipulator configuration $\mathbf{q}_E$. The function $\mathbf{K}$ has a closed-form representation in terms of homogeneous transformation matrices, and the solution to this problem is unique. The inverse kinematics problem is that of finding one of the possibly infinite number of manipulator configurations $\mathbf{q}_T = \mathbf{K}^{-1}(T)$ that causes the end-effector to achieve pose $\mathbf{T}$. In the general case of redundant manipulators, $\mathbf{K}$ cannot be inverted analytically. We now develop an optimization objective function to be minimized by PSO that requires only the forward kinematics model of the manipulator.

Let $T$ be a given target end-effector pose and let $\mathbf{E}(\mathbf{q})$ be the end-effector pose corresponding to configuration $\mathbf{q}$. Let $P_T$ and $P_E(\mathbf{q})$ be the 3D position vectors of poses $T$ and $\mathbf{E}(\mathbf{q})$, respectively for general manipulators. For planar manipulators, $P_T$ and $P_E(\mathbf{q})$ will be 2D position vectors. Define:

$$P_{\text{error}}(\mathbf{q}) = \|P_T - P_E(\mathbf{q})\|$$

Let $O_T$ and $O_E(\mathbf{q})$ be the vectors of Roll-Pitch-Yaw Euler Angles of poses $T$ and $\mathbf{E}(\mathbf{q})$, respectively (in radians) for general manipulators. For planar manipulators, $O_T$ and $O_E(\mathbf{q})$ will each be scalar values (in radians). Define:

$$O_{\text{error}}(\mathbf{q}) = ||O_T - O_E(\mathbf{q})||$$

Note that $P_E(\mathbf{q})$ and $O_E(\mathbf{q})$ can be calculated as $(P_E(\mathbf{q}), O_E(\mathbf{q})) = \mathbf{K}(\mathbf{q})$ using forward kinematics model $\mathbf{K}$ for any $\mathbf{q}$, while $P_T$ and $O_T$ are given. Let $\mathcal{C}(\mathbf{q})$ be a collision function which returns 0 if that configuration is self-collision-free – meaning that, at configuration $\mathbf{q}$, the manipulator does not collide with itself – and 1 otherwise. Then, the general position and orientation IK problem for given target pose $T$ for any manipulator can be solved by using any PSO variant to find a vector of joint angles $\mathbf{q}^*$ minimizing Equation 7:

$$F(\mathbf{q}) = a_T P_{\text{error}}(\mathbf{q}) + a_O O_{\text{error}}(\mathbf{q}) + a_\mathcal{C} \mathcal{C}(\mathbf{q}) \tag{7}$$

Where $a_T \geq 0$ and $a_O \geq 0$ are constants weighing the differing importance of $P_{\text{error}}$ and $O_{\text{error}}$, which are measured on different scales. $a_\mathcal{C} \gg 1$ is a large positive constant penalizing self-collisions. $a_\mathcal{C}$ should be set large enough that any solution of value greater than or equal to $a_\mathcal{C}$ cannot be valid. One easy way to set this value is to make it much larger than the largest possible $P_{\text{error}}$ plus the largest possible $O_{\text{error}}$. Note that $F(\mathbf{q}) \geq 0$ for any configuration $\mathbf{q}$ and any configuration $\mathbf{q}^*$ achieving $F(\mathbf{q}^*) = 0$ is an optimal solution to this minimization problem as it has zero position and orientation error and is self-collision-free. Conveniently, by setting $a_O = 0$ in Equation 7, PSO can easily be made to solve position only inverse kinematics (a common IK subproblem).

A general method for implementing $\mathcal{C}(\mathbf{q})$ for both planar and general manipulators is to use geometric overlap queries. We can associate objects of known geometry at a number of different points on the manipulator and test each pair of them for overlaps at any configuration $\mathbf{q}$. If any pair of objects is overlapping, there is a self-collision and we return 1. Otherwise, we return 0, indicating no self-collision. The poses of these objects can be computed efficiently for any $\mathbf{q}$ using the forward kinematics model $\mathbf{K}$ analogously to the way in which the end-effector pose is computed.

Note that PSO is searching directly in the configuration space of the manipulator. Each particle $x_i$ represents a candidate set of joint angles, where each dimension $j$ of particle $x_i$ is a candidate angle for joint $j$. Intuitively, at each PSO iteration, we evaluate each candidate configuration $x_i$ by passing it through the forward kinematics model and measuring the position and orientation error between where the end-effector would be at configuration $x_i$ and the target end-effector pose, adding on a large penalty value if configuration $x_i$ is in self-collision.

In order to enforce joint limits, each dimension $j$ of particle $x_i$ should be limited to searching in the range of valid joint angles $\text{limits}_j = [q_{j,\text{low}}, q_{j,\text{high}}]$. This can be achieved by clamping each dimension $j$ within these bounds at each iteration immediately after it is updated. Each dimension $j$ of each particle can be initialized uniformly at random in $\text{limits}_j = [q_{j,\text{low}}, q_{j,\text{high}}]$.

### IV. Experimental Results

A. PSO for High-DOF IK

In our previous work in [15], we developed a high-fidelity, physics-based 3D simulator capable of simulating hundreds of physically-connected robot modules. This simulator provided us with an appropriate testbed to evaluate PSO as a high-DOF IK solver because, by connecting robot modules one on top of another, we could build manipulators with incrementally increasing numbers of DOF. Due to its general kinematic structure and dexterity, we elected to use simulated SuperBot.
modules. Each SuperBot module can itself be viewed as a 3 DOF manipulator (see Figure 5 for a visualization). Figure 4 gives the DH parameters of a SuperBot module when viewed as a manipulator where its front dock is the base and its back dock is the end-effector. By connecting \( n \) SuperBot modules on top of one another front-to-back, we generate a 3\( n \)-DOF manipulator whose end-effector is the back dock of the topmost module. Since the modules have identical kinematics, the forward kinematics model of the entire manipulator is obtained by multiplying the individual transformations for each module.

The base of the manipulator — the front dock of the bottommost module — is anchored in simulation to a small rectangular platform (red) suspended in midair (see Figure 1) such that the manipulator can move freely unless it collides with itself or the platform.

In order to evaluate PSO as a high-DOF IK solver, we used two popular variants, Constriction factor PSO (PSO-Co) [5] and Bare Bones PSO (BB-PSO) [4], to optimize our fitness function (Equation 7) on manipulators with 9, 15, 30, 60, 90, 120, 150, and 180 DOF (3-60 SuperBot modules). For each algorithm variant and manipulator size combination, we selected a random self-collision-free manipulator configuration uniformly at random, computed the pose of the end-effector at that configuration, and fed that end-effector pose to the PSO IK solver being evaluated as the target pose. This ensured that the optimization problem had a valid solution with zero error. This process was repeated 200 times for each algorithm variant and manipulator size combination.

The results are tabulated in Figure 2 and visualized in Figures 3(a), 3(b), and 3(c). In Equation 7, \( a_x \) was set to 1, \( a_y \) was set to 0.3 (based on extensive experimentation by the authors) and \( a_z \) was set to 1000. 300 particles were used with a maximum iteration count of 3000 for manipulators with 30-180 DOF. The particle counts and maximum iteration counts for 9 and 15-DOF manipulators were set differently as they performed poorly with these parameters. Since different parameters were used for these manipulators, their results are
separated in the table with lines and omitted from the graphs in Figures 3(a), 3(b) and 3(c). For 15-DOF manipulators, 500 particles were used with a maximum iteration count of 750. For 9-DOF manipulators, 1000 particles were used with a maximum iteration count of 500. The fitness threshold $h$ was set to 0.001 (sub-millimeter precision with orientation visually indistinguishable from the target pose), meaning that the algorithm run was terminated when a solution with fitness less than 0.001 was found. In Constriction Factor PSO, $c_1$ and $c_2$ were each set to 2.05 and $\chi$ was set to 0.7298, as suggested in [14]. Each column in Figure 2 is an average over 200 runs. $P_{\text{error}}$ and $O_{\text{error}}$ are as defined in Equations 5 and 6. The Fitness column is the average fitness of the final solution returned by the algorithm at the termination of each run. The Iterations column is the average number of iterations performed by the algorithm before a solution with fitness lower than threshold $h = 0.001$ was found. Note that a self-colliding configuration could never reach this threshold.

The collision function $C(q)$ defined in Equation 7 was implemented using geometric overlap queries on bounding boxes assigned to each module and the red platform. If any pair of these bounding boxes overlapped at configuration $q$, 1 was returned (indicating self-collision). Otherwise, 0 was returned.

On manipulators with 30 or more DOF, Constriction Factor PSO consistently outperformed Bare Bones PSO along every metric tested, including average runtime. However, for the 9-DOF and 15-DOF manipulators, Bare Bones PSO performed significantly better on every metric tested. This inversion persisted for every combination of particle population size and maximum number of iterations tested (only the results of the best combination are given in the table in Figure 2). This provides evidence that Bare Bones PSO may offer an advantage over Constriction Factor PSO for solving the IK problem of lower-DOF manipulators. However, more testing of different manipulator kinematic structures is needed to confirm if this holds in general.

In fact, interestingly enough, both variants performed more poorly on 9 and 15-DOF manipulators than manipulators with 30 or more DOF. This is likely due to the fact that the number of solutions within the acceptable error tolerance increases as the number of DOF of the manipulator increases. On manipulators with 15 or fewer DOF, larger particle swarms with smaller maximum iteration counts performed much better than small particle swarms with large maximum iteration counts. For manipulators with 30 or more DOF, the inverse is true. This makes intuitive sense, as one would expect that larger particle swarms would help ensure the swarm evaluates many different parts of the fitness landscape to isolate the relatively few areas in which acceptable solutions can be found for lower-DOF manipulators. For higher-DOF manipulators, which likely have more potential solutions to each IK problem, a relatively small particle swarm can quickly isolate a few promising areas of the landscape and exploit these areas over large numbers of iterations to generate a solution of sufficient fitness.

Perhaps most encouraging about these results is the fact that the average position error, orientation error, and fitness of solutions using Constriction Factor PSO remained relatively flat, even as the manipulator DOF increased dramatically. Also encouraging was the fact that Constriction Factor PSO required fewer than 1000 iterations on average to find an acceptable solution, even for the largest manipulator sizes. This provides strong evidence that Constriction Factor PSO is a powerful high-DOF IK solver using our fitness function.

It is also important to note that neither algorithm variant ever converged to a solution that was not self-collision-free. This is an important result, as Monte Carlo simulations we have performed indicate that large percentages of all configurations of these high-DOF SuperBot manipulators are self-colliding (30% of all configurations for a 45-DOF SuperBot manipulator and 60% for a 90-DOF SuperBot manipulator). The ability of both variants of PSO to find valid IK solutions in these spaces is highly encouraging. Finally, we note that the runtime increases with manipulator DOF were due to the complexity involved in testing each pair of modules and the platform for geometric overlaps. More sophisticated and efficient self-collision testing (e.g., [16]) would dramatically reduce this running time.

B. PSO for IK on Real-World Hardware

The hardware setup used to validate the proposed PSO-based IK solution is visualized in Figure 6. It consists of a 6-DOF Manipulator (red) – composed of two autonomous SuperBot modules which communicate via message passing using IR – with an onboard camera and a docking interface at

![Figure 6. The hardware setup for 6D-docking experiments using SuperBot self-reconfigurable robots.](image)
the tip of its end-effector, a Reaction Wheel (blue) SuperBot module which rotates its middle motor to induce a counter-
rotation of the Freely Rotating Joint (green), thereby moving the non-stationary base of the manipulator, and a Target Module (orange) with an April Tag [17] (black). The entire system is a simulation of a microgravity environment.

A force controller running on the reaction wheel module autonomously moves the freely rotating joint until the manipulator's camera can detect the April Tag. Then, a PSO-based IK controller minimizing Equation 7 is used in a closed-
loop fashion to iteratively move the 6-DOF manipulator's docking interface into contact with the docking interface of the target module for final docking. At each step, the camera detects the April Tag, which gives the IK controller the relative position and orientation of the target module's docking interface (the target pose for the IK problem). The IK problem is then solved by the PSO-based IK controller with particles restricted to searching only a plus or minus 10 degree window surrounding each joint's current sensor reading (to reduce the effect of sensor and actuator noise).

The reaction wheel force controller is invoked whenever the manipulator's camera loses sight of the April Tag. This process continues iteratively, moving the end-effector of the manipulator closer to the target pose at each step, until the relative position and orientation of the target module's docking interface is within tight acceptable tolerances (5mm position accuracy, orientation error within 3 degrees on each axis). Physical docking with the target module is then initiated. Successful docking was achieved in approximately 86.7% of all tests (averaged over different starting distances, orientations, and displacements). $a_p$ was set to 1 and $a_o$ was set to 0.3. 1000 particles were optimized for a fixed number of iterations (500). Please see [18] for full details.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we generalized previous work to derive a full position function that, when optimized by PSO, solves the full position and orientation inverse kinematics problem for any serial manipulator while both respecting joint limits and avoiding self-collisions. We presented the first statistical analysis of two popular PSO variants (representing two important classes of PSO algorithms) as high-DOF IK solvers using our fitness function and also presented the first hardware-based validation of PSO-based IK on a precision manipulation task. Future work will evaluate additional PSO variants as IK solvers on real-world hardware and in simulation and will evaluate the effect of particle population size on IK solution quality. We also seek to combine this technique with statistical end-effector localization techniques to account for the accumulation of sensor and actuator noise in real-world manipulators, particularly those with high-DOF.

REFERENCES


[9] Z. F. Cai, D. P. Huang, and Y. Q. Liu, “Inverse kinematics of multi-


docking and manipulation with non-stationary-base using self-