PASO: An Integrated, Scalable PSO-based Optimization Framework for Hyper-Redundant Manipulator Path Planning and Inverse Kinematics

Thomas Collins and Wei-Min Shen

***Abstract*— Hyper-redundant manipulation is the use of a hyper-DOF robotic system to accomplish tasks such as picking, placing, reaching, and exploring in challenging environments. Hyper-redundant manipulation involves both finding collision-free configuration-space paths (the *path planning* (PP) problem) and transforming a given workspace target pose into a configuration space goal (the *inverse kinematics* (IK) problem). Traditional, Jacobian-based IK approaches become computationally infeasible as the number of DOF increases. Existing hyper-DOF manipulator PP approaches either require extensive global preprocessing (e.g., Probabilistic RoadMaps) or spend considerable time on costly nearest neighbor computations (e.g., Rapidly-exploring Random Trees). We propose an integrated, scalable optimization framework called PASO (PAth planning with Swarm Optimization) that uses Particle Swarm Optimization in a divide-and-conquer fashion to efficiently produce approximate solutions to both the hyper-DOF PP and IK problems together. Implicitly-defined C-space waypoints are used to bias C-space sampling. We present promising experimental results showing PASO's ability to scale to much larger manipulators (120 DOF, so far) than reported in the literature on current integrated position and orientation IK and PP solvers for general serial hyper-DOF manipulators in 3D workspaces.**

I. Introduction

Hyper-dimensional (hyper-DOF) robotic manipulators have a tremendous number of potential uses: cleaning pipes, exploring crevices, aiding astronauts with space station repairs, etc. Their high level of kinematic redundancy gives them the flexibility to perform their primary tasks while avoiding collisions with obstacles, avoiding self-collisions, and possibly even meeting secondary constraints relevant to the task (e.g., favoring smooth curves).

We are therefore interested in the following question: how do we plan a collision-free and self-collision-free path – a sequence of joint configurations, each of which is collision-free and self-collision-free – for a hyper-DOF robotic manipulator that leads it from a starting joint configuration to a joint configuration that achieves the goal pose (specified in workspace), given a forward kinematics model and a collision detector? We consider this dual *inverse kinematics* (IK) and *path planning* (PP) problem for serial, hyper-DOF manipulators.

The general hyper-redundant manipulation problem is extremely difficult (the motion planning portion alone is known to be PSPACE-complete with respect to the number of DOF [1]). Combining existing methods may not be sufficient because they are either computationally too expensive for high-DOF (e.g., traditional IK approaches), require extensive pre-processing (e.g., global planners based on a Probabilistic Roadmap[2]), or require a costly nearest neighbor search (e.g., local approaches such as Rapidly-exploring Random Trees [3]).

In this paper, we propose the PASO (PAth planning with Swarm Optimization) framework (Section IV), a novel integrated approach to efficiently solving the IK and PP problems for general serial hyper-DOF manipulators in 3D workspaces. Given a workspace target pose, PASO utilizes the powerful Particle Swarm Optimization (PSO) [4] algorithm to first solve the hyper-DOF IK problem for a goal joint configuration that achieves that pose, and then uses PSO to recursively subdivide the PP problem to a desired resolution. The set of collision-free and self-collision-free configurations selected to subdivide the subproblems are the output of the PP phase.

The contributions of this work are as follows:

- We propose a simple fitness function that, when minimized by PSO, solves the general hyper-DOF position and orientation inverse kinematics problem for an n-DOF serial manipulator in a 3D workspace, thus generalizing previous work on using PSO to solve the IK problem. We call the use of PSO with this new fitness function PSOIK. We show the extreme scalability of PSOIK by testing it on manipulators with up to 180 DOF (Section III).
- We propose the PASO optimization framework, a scalable integrated PP and IK solver (approximation) for general serial hyper-DOF manipulators in 3D workspaces. PASO is novel in terms of its extreme scalability (up to 120 DOF so far, with up to 150 obstacles) in solving the IK and PP problems together. It is also novel in its use of PSO to recursively subdivide the PP problem based on waypoints (Section IV). PSOIK forms an integral component of PASO.

We present promising experimental results generated in physics-based simulation in Section V. First, however, we review previous work in this area (Section II).

II. Related Work

A. Path Planning

Path planning techniques for hyper-DOF manipulators can be classified either as *local*, if they incrementally step toward a goal configuration from a starting configuration by localized search (see, e.g., [5], [6]); or *global*, if they search for collision-free paths on precomputed graphs of the manipulator's C-space (see, e.g., [7], [8]).

Thomas Collins and Wei-Min Shen are with Information Sciences Institute, The University of Southern California, Los Angeles, U.S.A. collinst@usc.edu, and shen@isi.edu

Local methods such as Rapidly-exploring Random Trees (RRTs) [3] spend significant time computing nearest neighbors and have difficulty finding solutions in high dimensional spaces. In [9], a standard RRT was unable to find solutions to planar manipulator PP problems with more than 20 DOF in any reasonable amount of time. Global methods, particularly those based on Probabilistic RoadMaps (PRMs) [2] and cell decompositions [10], require extensive pre-computation that is often exponential in the number of DOF [11]. Hybrid global and local approaches have been successful [12], [13], but their applicability to manipulation has been limited primarily to lower-dimensional C-spaces than those considered here. All the above techniques also generally require separate IK algorithms to translate the goal into C-space. OMPL, available at http://ompl.kavrakilab.org, provides high-quality implementations of many such algorithms.

Finally, we note that PSO has been shown to be an effective path planning methodology [14], [15], particularly for mobile robots and manipulators with little redundancy. Its applicability to the path planning of manipulators with complex and general kinematic structures and hyper-redundancy (dozens to hundreds of DOF) is still an open question.

B. Inverse Kinematics

Classical approximation approaches to solving the IK problem, such as those based on the Jacobian, suffer from numerical issues around singularities and do not scale well with the number of DOF [16].

The limitations of Jacobian-based approximations have led to a plethora of alternative solution strategies, the most relevant to the present work being those based on sequential Monte Carlo methods [17] and optimization approaches, such as Particle Swarm Optimization (PSO). The sequential monte carlo methods of [17] were shown to be highly effective for solving the hyper-DOF IK problem, but the path planning problem was not addressed. It was assumed that the arm could go directly from the starting configuration to the configuration output by the Monte Carlo IK solver.

PSO has been used to solve the position inverse kinematics problem for a particular 6 DOF manipulator in [18] and in order to generate biped gaits in [19]. A statistical analysis of the application of PSO to the inverse kinematics problem was presented in [20], but only 2 DOF planar manipulation was considered. Also, PSO was used to solve the IK problem for 7-DOF redundant manipulators in [21]. In this work, we generalize the use of PSO to the hyper-DOF position and orientation IK problem of an n-DOF manipulator in the form of PSOIK (Section III). This generalization can take into account and avoid self-collisions, an important aspect of manipulation not previously considered in such work.

For a thorough analysis of the multitude of IK techniques applicable to redundant and hyper-redundant manipulators consult [16].

C. Integrated Path Planning and Inverse Kinematics

Several integrated path planning and inverse kinematics solution frameworks have been proposed for redundant and hyper-redundant manipulators. For example, in [6], an RRT-based IK and PP solver is proposed for redundant manipulators. However its efficiency is only evaluated for manipulators with 7-DOF, and its scalability is unclear.

In [9], Shkolnik and Tedrake present a powerful IK and PP framework for hyper-redundant manipulators which augments the basic RRT algorithm to grow trees in a low-dimensional task space (W-space) rather than a high-dimensional configuration space (C-space). The manipulator Jacobian pseudoinverse is used to bias the sampling toward the W-space goal. In this way, the dimensionality of the nodes in the tree stays roughly constant, regardless of the number of manipulator degrees of freedom. The resulting TS-RRT has been shown to scale up to 1500 DOF manipulators. However, the inverse kinematics problem was not solved directly, the Jacobian pseudoinverse was used, which causes irregularities in the RRT generated, particular near the edges of the reachable workspace, the manipulators tested resided only in planar workspaces and self-collisions were not considered. Unfortunately, as the DOF of the manipulator increases, the likelihood of self-collisions greatly increases, severely limiting the available actions of the manipulator. In many cases, this may have a large negative impact on the effectiveness of searching for a path in W-space rather than C-space.

A similar approach is presented in [22], where again an RRT is grown in W-space rather than C-space. The random sampling is guided by the Maximum Reachable Area (MRA) of the manipulator at any W-space point. This approach also does not solve the inverse kinematics problem directly and has many of the same drawbacks as TS-RRT above. Additionally, this method was only experimentally tested on planar manipulators with up to 20 DOF.

In contrast, PASO can provide approximate answers to both the IK and PP problems directly for any serial hyper-DOF manipulator in 3D workspaces, and singularities are avoided altogether by avoiding the use of the manipulator Jacobian.

III. PSOIK: PSO for Inverse Kinematics

Particle Swarm Optimization (PSO) is a swarm-based optimization algorithm that has been shown to be quite effective in solving difficult and high-dimensional optimization problems in a number of diverse domains [24], [25]. The basic idea of PSO is that a swarm of m particles, each n-dimensional, perform an independent search in the space of possible n-dimensional solutions. Each particle i is a point x_i in this search space with a certain velocity v_i and has an associated fitness given by the objective function (F) value at that point $F(x_i)$. Particles move around in this search space randomly but sampling is biased toward a random weighted average of the best position achieved by any particle in the swarm g and the best position achieved by each particle individually, p_i. This focuses random searches on areas of the search space where a global optimum is expected to be.

At each iteration $t+1$, every dimension $j = 1, 2, ..., n$ of particle i is updated according to the following two

Fig. 1. SuperBot [23] manipulators ranging from 30 to 180 DOF (10-60 SuperBot modules) were used to test PSOIK. Illustrated here are a 30 DOF manipulator (left), a 90 DOF manipulator (middle), and a 180 DOF manipulator (right).

DOF (n)	Avg. Runtime (seconds)	Avg. $posError$	Avg. $orientError$	Avg. Iterations
30	1.57	0.00046	0.00314	275.84
60	3.36	0.00034	0.00260	383.165
90	7.46	0.00036	0.00210	533.055
120	15.47	0.00034	0.00554	697.945
150	22.11	0.00036	0.00515	764.065
180	37.03	0.00032	0.00237	947.255

Fig. 2. PSOIK Results. $posError$ and $orientError$ are as defined in Equation (3). Avg. Iterations is the average number of PSO iterations performed before the fitness threshold of $h = 0.001$ was met. We see that for all manipulator sizes tested, PSOIK was able to consistently converge on a solution below the threshold h. PSOIK never converged to a solution that was not collision-free and self-collision-free.

equations:

$$v^j_{i,t+1} = v^j_{i,t} + c_1 R^j_{1_{i,t}}(p^j_{i,t} - x^j_{i,t}) + c_2 R^j_{2_{i,t}}(g^j_t - x^j_{i,t}) \quad (1)$$

$$x^j_{i,t+1} = x^j_{i,t} + v^j_{i,t+1} \quad (2)$$

In equation (1), c_1 and c_2 are constants (algorithm parameters), $R^j_{1_{i,t}} \sim U(0,1)$ and $R^j_{2_{i,t}} \sim U(0,1)$. Particle positions and velocities are often clipped to keep them within certain bounds. This process continues until a certain fitness threshold h is reached or a maximum number of iterations N is performed. When the algorithm terminates at iteration k ($k \in 1, 2, ..., N$), the solution returned by the algorithm is g_k, the current global best particle in the swarm.

To apply PSO to IK problems for any serial manipulator with any number of DOF, we propose to let n be the number of DOF of the manipulator in question and search directly in the space of possible manipulator joint angles using a new Equation (3) below as a novel fitness function (to be minimized):

$$F(\mathbf{q}) = w_p posError + w_o orientError + w_1 collisions(\mathbf{q}) \quad (3)$$

The function F maps a set of joint angles in $\mathbb{R}^n$ (where n is the number of manipulator DOF) to a real number, which is the fitness of joint angles $\mathbf{q}$. $posError$ is the magnitude of the position error between the end effector pose corresponding to $\mathbf{q}$ (given by the forward kinematics model) and the target end-effector pose. $orientError$ is the magnitude of the vector of Euler angles representing the rotational difference between the pose of the end-effector corresponding to $\mathbf{q}$ and the target end-effector pose. $collisions$ is a collision detector function that returns 1 if $\mathbf{q}$ causes a collision or self-collision, and 0 otherwise. w_p, w_o and w_1 are nonnegative weights trading off the relative importance of each error term. It is easy to see that arbitrary constraints could be added to this fitness function simply by adding more nonnegative weighted terms $w_2, w_3, ...$ to be minimized. Our new PSOIK approach is simply the integration of Equation (3) into any PSO variant.

In order to empirically test the efficacy of PSOIK, the authors performed a number of experiments on simulated SuperBot [23] manipulators with 30, 60, 90, 120, 150, and 180 DOF in the ReMod3D [26] simulator in a 3D workspace using the canonical (original) PSO variant. 200 runs were done per manipulator size. Each run consisted of selecting an end-effector pose guaranteed to be in the reachable task space of the manipulator (uniformly at random), running PSOIK on that input using the fitness function defined in Equation (3) (with $w_p = 1$, $w_o = 0.3$, and $w_1 = 1000$), and recording the running time, fitness, position error, orientation error, and number of iterations performed to compute the output solution. Between 200 to 350 particles and 1500 to 3000 maximum iterations were used (depending on manipulator size). h was set to 0.001. The results are illustrated in Figure 1, and summarized in Figure 2. All IK solutions computed by PSOIK were collision and self-collision-free.

Algorithm 1: PASO Algorithm

Input: F_P : C-space waypoint function; δ : maximum step size (path resolution); $depthLimit$: recursive depth limit; W_{target} : W-space target point; C_{start} : C-space starting point; H : collision detector;

Output:

$success$: true if PASO succeeded, false otherwise

P: path in C-space

C_{target}: solution to IK problem for W_{target}

Function `PASO()`
  $P := \emptyset$;
  $P := P \cup C_{start}$;
  `/* W-Space goal to C-space */`
  C_{target} := PSOIK(W_{target});
  $success$:= PASOR(C_{start}, C_{target}, 0);
  $P := P \cup C_{target}$;
  return *success, P, C_{target}*;

Function `PASOR(`c_{start}, c_{goal}, *depth*`)`
  $error := \|c_{start} - c_{goal}\|$;
  if *no collisions along waypoints from c_{start} to c_{goal} to resolution δ* **then**
    return *true*;
  if *error* $<$ *tol* **then**
    return *true*;
  if *depth* $>$ $depthLimit$ **then**
    return *false*;
  else
    `/* Compute target waypoint */`
    $c_{way} := F_P(c_{start}, c_{goal})$;
    `/* Optimize toward` C_{way} `using PSO with Equation (4) */`
    c_{split} := PSO(c_{way});
    if *c_{split} is in collision* **then**
      return *false*;
    $P := P \cup c_{split}$;
    return *PASOR(c_{start}, c_{split}, depth + 1) && PASOR(c_{split}, c_{goal}, depth + 1)*;

IV. PASO: Applying PSO To Path Planning in Hyper-DOF Spaces

Given a target end-effector pose in workspace (W-space), the PASO framework solves the hyper-DOF IK problem using PSOIK, generating a self-collision-free and collision-free configuration space (C-space) target with respect to which C-space path planning can occur. Path planning is then solved in C-space using a recursive, divide-and-conquer approach, where the splitting of the problem is achieved using implicitly-defined sets of waypoints. These waypoints represent a desired default path for the manipulator to follow if self-collisions and collisions do not prevent the execution of a path through these points.

More formally, given a forward kinematics model, a collision detector H, a starting joint configuration, C_{start}, a target end-effector position and orientation in workspace, W_{target}, and a function F_P that, given a two manipulator configurations $start$ and $goal$, produces a single waypoint C_{way} approximately halfway between $start$ and $goal$, the PASO framework works as follows:

Solve the hyper-DOF IK problem: W_{target} is converted into manipulator configuration space (C-space) by finding a self-collision-free and collision-free manipulator configuration C_{target} that achieves W_{target}. This is done using PSOIK.

Solve the hyper-DOF PP problem: The problem has now been converted into C-space, setting up a PP problem between manipulator configurations C_{start} and C_{target}. First, using F_P, waypoints are recursively computed between C_{start} to C_{target} until the distance between any two consecutive waypoints is lower than C-space resolution δ and each is tested by collision detector H. If no collisions or self-collisions exist, planning is successfully completed to resolution δ. Otherwise, we propose to run PSO optimization with the following new fitness function (to be minimized):

$$F_C(\mathbf{q}) = \|\mathbf{q} - C_{way}\| + w_1 collisions(\mathbf{q}) \quad (4)$$

Note that the above fitness function optimizes in C-space, not W-space, $collisions(\mathbf{q})$ is as defined in Section III, and arbitrary constraints with nonnegative weights $w_2, w_3, ...$ could again be added. The above PSO optimization attempts to find a self-collision-free and collision-free C-space *split point* as close to C_{way} as possible. This divides the PP problem into two subproblems, one from C_{start} to C_{split} and the other from C_{split} to C_{target}, both of which are recursively subdivided in the same way. This recursion continues until either a recursive depth limit d is reached or $\|C_{start,i} - C_{target,i}\| < \delta$ for subproblem i is achieved. The computed split points are precisely the path output by the PASO framework. At each step of the recursion, the waypoints from $C_{start,i}$ to $C_{goal,i}$ are tested to see whether or not recursive subdivision using PSO is necessary. In the output path, if two configurations are separated by a distance greater than δ, the waypoints between these two configurations (to resolution δ) are guaranteed to be self-collision and collision-free. The algorithm fails if any split point chosen is in self-collision or collision with any obstacle or if the recursive depth limit is exceeded. Otherwise, the PP problem is successfully solved to resolution δ.

As an example of how to define waypoints implicitly in PASO, consider a PP subproblem from C_{start} to C_{goal}. The first waypoint in this case might simply be

$$C_{way} = C_{start} + 0.5(C_{goal} - C_{start}) \quad (5)$$

Similarly, for any sub-subproblem i of the original PP subproblem from $C_{start,i}$ to $C_{goal,i}$, waypoint $C_{way,i} = C_{start,i} + 0.5(C_{goal,i} - C_{start,i})$. In this case, each waypoint cuts the PP subproblem exactly in half (in terms of Euclidean distance), creating two new subproblems. See Algorithm 1 for an outline of the full PASO framework.

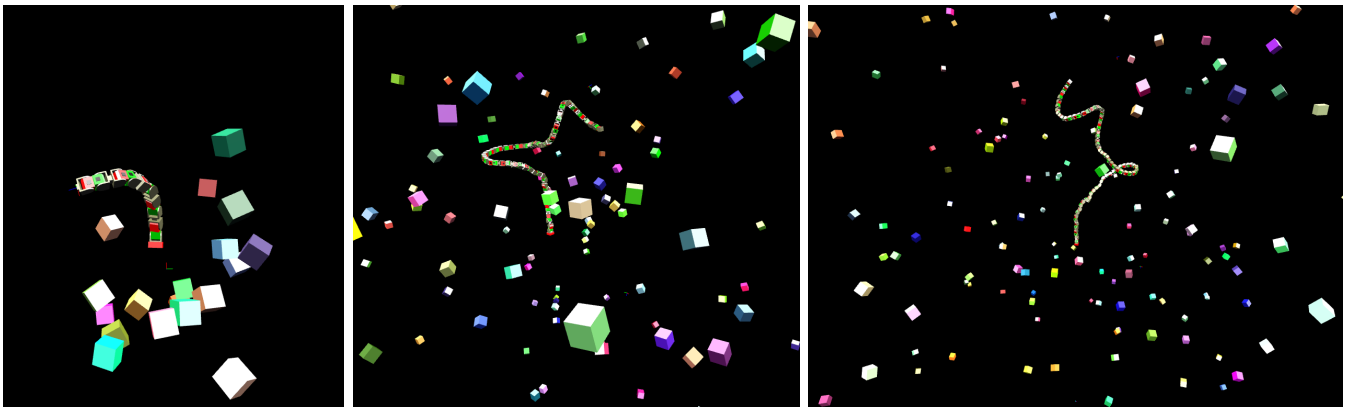

Fig. 3. The experimental setup in ReMod3D for the PASO evaluation showing the size and distribution of random obstacles. This image shows a 15 DOF (left), a 60 DOF (middle), and a 120 DOF (right) SuperBot manipulator executing collision-free and self-collision-free paths computed by PASO.

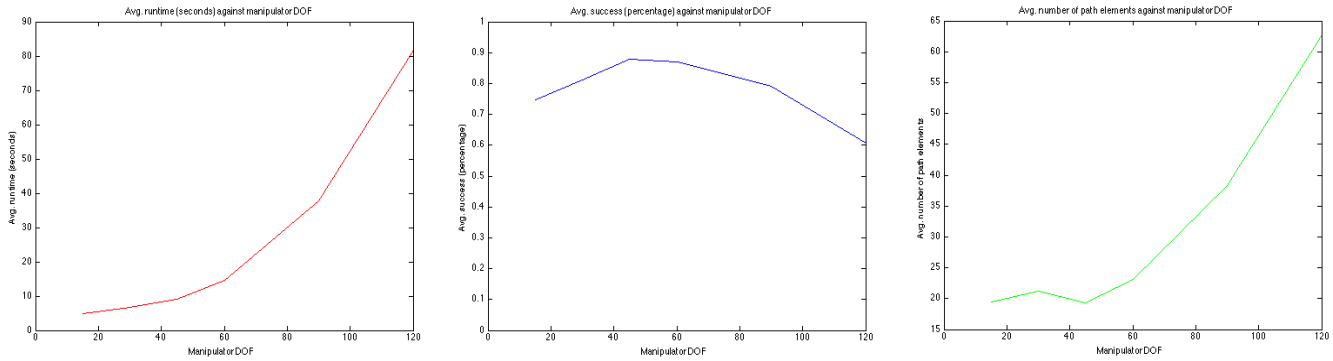

Fig. 4. Summary of PASO experimental results. On the left, we plot expected runtime (averaged over 1000 runs per manipulator size) of PASO against the number of manipulator DOF. In the middle, we plot the ratio of successful PASO runs to total runs per manipulator size against manipulator DOF. On the right, we plot the average number of path elements in a planned path against manipulator DOF.

V. PASO Evaluation and Analysis

To evaluate PASO, we tested it with manipulators made of simulated SuperBot [23] modules with 15, 30, 45, 60, 90, and 120 DOF in the ReMod3D [26] simulator. For each manipulator size, we ran PASO 1000 times. For each run, a random starting joint configuration and a random W-space target guaranteed to correspond to at least one collision and self-collision-free joint configuration were chosen as C_{start} and W_{target}, respectively. In other words, for each run, there was guaranteed to be at least one manipulator configuration in C-free that achieved W_{target} exactly. PASO then solved the IK problem for C_{target} and attempted to plan a path to the desired resolution $\delta = 9.0$ (very fine-grained) between C_{start} and C_{target}. Waypoints were generated using Equation (5). The running time, path size, and convergence were analyzed for each run. Let n be the number of degrees of freedom of the manipulator. Then, for each n, $1.25n$ obstacles were spread out uniformly at random throughout the reachable task space of the manipulator. Each obstacle was a cube whose sides were approximately the size of the length of a manipulator link. The 1000 runs for each manipulator size were distributed equally across 5 randomly generated environments (200 runs per environment instance). 200 particles and an iteration limit of 1500 were used for PSOIK. Recursive subdivision PSO swarms (those using Equation 4) were given 50 particles and a maximum iteration count of 100. The results are summarized in Figure 4. The experimental setup is illustrated in Figure 3.

It is very important to mention that, though the starting manipulator configuration was in C-free and W_{target} was guaranteed to correspond to at least one manipulator configuration in C-free, there was no guarantee a path in C-free existed between C_{start} and the result of the IK phase, C_{target}. Some failures of the algorithm may have been unavoidable due to the lack of connectivity of the C-space.

It is clear from PASO's runtime results in Figure 4(left) that it is a very scalable framework. Even when planning paths for a manipulator with 120 DOF in an environment with 150 obstacles, PASO only required 80 seconds on average to terminate. The quadratic nature of the runtime curve is due primarily to the collision detector, which explicitly checked every pair of manipulator links for collisions and also checked every manipulator link/obstacle pair for collisions. Collision detection was not the focus of this work, so this naïve implementation suited our purposes; however, it is likely that more sophisticated collision detectors could reduce the expected runtime drastically.

In Figure 4(middle), we plot the ratio of successful PASO runs to the total 1000 runs per manipulator size. The initial climb from 75% (15 DOF) to approximately 88% (60 DOF)

makes very intuitive sense. We expect that as the number of manipulator DOF rises (i.e., the dimensionality of the search space becomes larger), self-collision-free and collision-free path(s) from the starting configuration to the goal configuration will lie increasingly close to the straight line in C-space between these configurations, making it easier for the PASO method (with the waypoints defined in Equation (5)) to find a viable solution. This is simply due to the fact that the arm has the increased dexterity (freedom along an increasing number of dimensions) necessary to avoid obstacles and itself on the way to its target configuration.

It is tempting to assume that this line of thinking should extend to higher and higher DOF in the sense that it should become easier and easier for PASO to find solutions as the dimensionality of the C-space rises. Unfortunately, this intuition is not justified due to the fact that manipulators of this size face an extremely large number of potential self-collisions. For example, a simple Monte Carlo simulation of the C-space of a 90 DOF SuperBot manipulator shows that approximately 60% of all manipulator configurations are in self-collision. This is an increase from approximately 42% for a 60 DOF SuperBot manipulator. This high probability of self-collision means that the arm may have to "unwind" itself far away from the straight line in C-space between the current configuration and the goal configuration in order to find a viable path. This can be overcome to some extent either by increasing the amount of computation in the recursive PSO subdivision swarms or by increasing δ, as a fixed resolution becomes more and more difficult to attain as the C-space dimensionality continues to increase.

Figure 4(right) plots the average size (number of elements) in the calculated PASO paths against the number of manipulator DOF. In all manipulator sizes tested, significant portions of the straight line in C-space can be followed in almost every path planning problem instance. This can be seen by comparing the average path length computed by the PASO framework presented in Algorithm 1 with a version of PASO that recursively subdivides the space using PSO without first checking if the waypoints are clear (that is, ignores lines 11 and 12 in Algorithm 1). This brute force version of PASO plans paths with an average of 69 elements for a 15 DOF manipulator, 125 elements for a 30 DOF manipulator, 240 elements for a 45 DOF manipulator, 426 elements for a 60 DOF manipulator, and 966 elements for a 90 DOF manipulator, given the same resolution $\delta = 9.0$. Using this brute force method, 120 DOF manipulators were too computationally expensive to test. The dramatic difference in the number of path elements illustrates the power of PASO in hyper-DOF contexts to find large sections of almost all planned paths in which it could follow the waypoints given in Equation (5) exactly. This provides empirical evidence that, despite its simplicity, Equation (5) computes reasonable waypoints in this PP context.

At this point, one of PASO's primary drawbacks is the requirement that it be supplied with a function that generates good waypoints. In this work, we proposed one such function (Equation (5)) and found that it works well for the problem at hand. However, other such functions may work better, and we are currently investigating automated methods for choosing high-quality waypoints and learning high-quality waypoint functions.

VI. Conclusions and Future Work

We have presented a novel divide-and-conquer optimization framework called PASO that efficiently solves the PP and IK problems in an integrated fashion for general serial hyper-DOF manipulators in 3D workspaces. PASO is based on the powerful Particle Swarm Optimization (PSO) algorithm. We first developed PSOIK, a generalized application of PSO to solving the IK problem of serial manipulators with any number of DOF. We then described how PSO could be used in a recursive, divide-and-conquer fashion to create a scalable PP framework for hyper-DOF manipulators. These two pieces together form the proposed integrated PP and IK framework called PASO. We presented the results of numerous experiments in physics-based simulation that illustrated the feasibility, scalability, and power of the proposed method.

For future work, we will continue to analyze the theoretical properties of PASO, particularly its completeness properties. We are also currently porting PASO completely to the GPU. Our initial results have shown dramatic speedup over the serial version presented in this work. Finally, we will assess the applicability of PASO to problems other than hyper-DOF manipulation. We have successfully used PASO to plan collision-free paths for simulated mobile robots, and will continue to investigate its generality.

References

[1] A. Hayashi, "Geometrical motion planning for highly redundant manipulators using a continuous model," Ph.D. dissertation, Austin, TX, USA, 1994.

[2] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566–580, 1996.

[3] J. Kuffner and S. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 2, 2000, pp. 995–1001 vol.2.

[4] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on*, 1995, pp. 39–43.

[5] C. Fragkopoulos and A. Graeser, "Sampling based path planning for high dof manipulators without goal configuration," in *International Federation of Automatic Control, 2011. Proceedings., 8th International Conference on*, 2011, pp. 11 568–11 573.

[6] D. Bertram, J. Kuffner, R. Dillmann, and T. Asfour, "An integrated approach to inverse kinematics and path planning for redundant manipulators," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, May 2006, pp. 1874–1879.

[7] D. Xie and N. Amato, "A kinematics-based probabilistic roadmap method for high dof closed chain systems," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 1, 2004, pp. 473–478 Vol.1.

[8] A. Shukla, E. Singla, P. Wahi, and B. Dasgupta, "A direct variational method for planning monotonically optimal paths for redundant manipulators in constrained workspaces," *Robotics and Autonomous Systems*, vol. 61, no. 2, pp. 209 – 220, 2013.

[9] E. Shkolnik and R. Tedrake, "Path planning in 1000+ dimensions using a task-space voronoi bias," in *In IEEE International Conference on Robotics and Automation*, 2009.

[10] F. Lingelbach, "Path planning using probabilistic cell decomposition," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 1, 2004, pp. 467–472 Vol.1.

[11] J. F. Canny, *The Complexity of Robot Motion Planning*. Cambridge, MA, USA: MIT Press, 1988.

[12] B. Krogh and C. Thorpe, "Integrated path planning and dynamic steering control for autonomous vehicles," in *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, vol. 3, 1986, pp. 1664–1669.

[13] C. Warren, "Global path planning using artificial potential fields," in *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, 1989, pp. 316–321 vol.1.

[14] Y.-Q. Qin, D.-B. Sun, N. Li, and Y.-G. Cen, "Path planning for mobile robot using the particle swarm optimization with mutation operator," in *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, vol. 4, 2004, pp. 2473–2478 vol.4.

[15] A. Taherifar, A. Alasty, H. Salarieh, and M. Boroushaki, "Path planning for a hyper-redundant manipulator with lockable joints using pso," in *Robotics and Mechatronics (ICRoM), 2013 First RSI/ISM International Conference on*, 2013, pp. 224–229.

[16] S. Yahya, M. Moghavvemi, and H. A. F. Mohamed, "Redundant manipulators kinematics inversion," vol. 6, pp. 5462–5470+, 2011.

[17] N. Courty and E. Arnaud, "Inverse kinematics using sequential monte carlo methods," in *Articulated Motion and Deformable Objects*, ser. Lecture Notes in Computer Science, F. Perales and R. Fisher, Eds. Springer Berlin Heidelberg, 2008, vol. 5098, pp. 1–10.

[18] B. Durmus, H. Temurtas, and A. Gun, "An inverse kinematics solution using particle swarm optimization," in *Proc. of sixth International Advanced Technologies Symposium (IATS?11), Turkey*, 2011, pp. 193–197.

[19] N. Rokbani and A. M. Alimi, "Ik-pso, pso inverse kinematics solver with application to biped gait generation," *arXiv preprint arXiv:1212.1798*, 2012.

[20] N. Rokbani and A. Alimi, "Inverse kinematics using particle swarm optimization, a statistical analysis," *Procedia Engineering*, vol. 64, no. 0, pp. 1602 – 1611, 2013.

[21] H.-C. Huang, C.-P. Chen, and P.-R. Wang, "Particle swarm optimization for solving the inverse kinematics of 7-dof robotic manipulators," in *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, Oct 2012, pp. 3105–3110.

[22] J. Lee and S. eui Yoon, "Prot: Productive regions-oriented task space path planning for hyper-redundant manipulators," in *ICRA*, 2014.

[23] B. Salemi, M. Moll, and W.-M. Shen, "SUPERBOT: A deployable, multi-functional, and modular self-reconfigurable robotic system," Beijing, China, Oct. 2006.

[24] J. Sun, C. Lai, and X. Wu, *Particle Swarm Optimisation: Classical and Quantum Perspectives*, ser. Chapman & Hall/CRC Numerical Analysis and Scientific Computing Series. Taylor & Francis, 2011.

[25] R. Poli, "Analysis of the publications on the applications of particle swarm optimisation," *J. Artif. Evol. App.*, vol. 2008, pp. 4:1–4:10, Jan. 2008.

[26] T. Collins, N. O. Ranasinghe, and W.-M. Shen, "Remod3d: A high-performance simulator for autonomous, self-reconfigurable robots," Tokyo, Japan, Nov. 2013.