6th IEEE RAS/EMBS International Conference on
Biomedical Robotics and Biomechatronics (BioRob)
June 26-29, 2016. UTown, Singapore

A Bio-inspired Algorithm for Identifying Unknown Kinematics from a Discrete Set of Candidate Models by Using Collision Detection

Dylan P. Losey, *Student Member, IEEE*, Craig G. McDonald *Student Member, IEEE*, and
Marcia K. O'Malley, *Senior Member, IEEE*

***Abstract*— Many robots are composed of interchangeable modular components, each of which can be independently controlled, and collectively can be disassembled and reassembled into new configurations. When assembling these modules into an open kinematic chain, there are some discrete choices dictated by the module geometry; for example, the order in which the modules are placed, the axis of rotation of each module with respect to the previous module, and/or the overall shape of the assembled robot. Although it might be straightforward for a human user to provide this information, there is also a practical benefit in the robot autonomously identifying these unknown, discrete forward kinematics. To date, a variety of techniques have been proposed to identify unknown kinematics; however, these methods cannot be directly applied during situations where we seek to identify the correct model amid a discrete set of options. In this paper, we introduce a method specifically for finding discrete robot kinematics, which relies on collision detection, and is inspired by the biological concepts of body schema and evolutionary algorithms. Under the proposed method, the robot maintains a population of possible models, stochastically identifies a motion which best distinguishes those models, and then performs that motion while checking for a collision. Models which correctly predicted whether a collision would occur produce candidate models for the next iteration. Using this algorithm during simulations with a Baxter robot, we were able to correctly determine the order of the links in 84% of trials while exploring around 0.01% of all possible models, and we were able to correctly determine the axes of rotation in 94% of trials while exploring $< 0.1\%$ of all possible models.**

I. Introduction

Body schema—adaptive representations of the body which change over time—are hypothesized to guide movements and actions in humans and other animals. Hoffmann *et al*. [1] argue that robots which are capable of autonomously identifying unknown kinematics can also be thought to possess body schema, since these continuously updating models help robots move correctly despite changes in their structure. In particular, identifying kinematics in real time helps robots compensate for unexpected failures, deal with unstructured environments, and adapt to mechanical modifications; plus, it reduces expenses associated with the robot's re-programming and maintenance [2]. Robotic "body schema" (i.e., kinematic identification processes) also provide a platform to explore the cognitive development of human body representations, particularly in regards to infant growth and tool usage [3]. Lastly, considering the connection between body schema and disabilities caused by strokes [4]—as well as the correlation from body schema to prosthetic embodiment [5]—the study of body schema acquisition methods may afford insights for better rehabilitation devices and prosthetic designs.

Several techniques have been proposed to automatically identify unknown kinematic information, some of which resemble biological notions of body schema. Hollerbach and Wampler [6] used closed kinematic chains and joint position sensors in order to calibrate continuous Denavit-Hartenberg parameters with least squares optimization. Hersch *et al*. [7] demonstrated an online gradient descent approach which employed cameras to determine the translation and rotation between successive links; their results were improved by Martinez-Cantin *et al*. [8], who reintroduced recursive least squares and added intelligent testing. Sturm *et al*. [9] developed Bayesian networks—in conjunction with external sensors and visual markers—so as to both identify the robot's kinematic structure from scratch and directly relate action signals to manipulator pose. Bongard *et al*. [10] programmed machines which can learn their own morphology through self-modeling before compensating for damages; multiple models constantly compete to explain preceding sensor data, and the most likely body schema is used to generate desired behaviors. From a more biologically-inspired perspective, Fuke *et al*. [11] built cross-modal maps to create representations for robotic segments invisible to the camera, and Saegusa *et al*. [12] described a framework by which action generation engenders understanding of the robot's kinematics and interaction capabilities.

While the listed methods can identify unknown kinematics in most situations, they cannot be straightforwardly applied when there exists only a discrete set of possible models. Although uncommon, this situation arises with robots composed of interchangeable, modular components which can be assembled into a finite variety of configurations [13]. Consider, for example, the relative order of modules a and b; since a is attached either before or after b with respect to the robot's base frame, any model must choose one of these two discrete options. Interestingly, even when the kinematics and controllers for each individual module are known, there are still several aspects of the overall kinematics—e.g., module order, relative axis of rotation, resultant shape—which can only be described by a discrete set of candidate models.

In our paper, we propose an algorithm specifically for identifying unknown kinematics when a discrete set of candidate models are given (see Fig. 1). While designing this algorithm we attempted to mimic biological body schema; in particular, given the significance of touch in human body

This work was funded in part by the NSF GRFP-0940902, and in part by the NSF IGERT-1250104. The authors are with the Mechatronics and Haptic Interfaces Laboratory, Department of Mechanical Engineering, Rice University, Houston, TX 77005. (e-mail: dlosey@rice.edu)

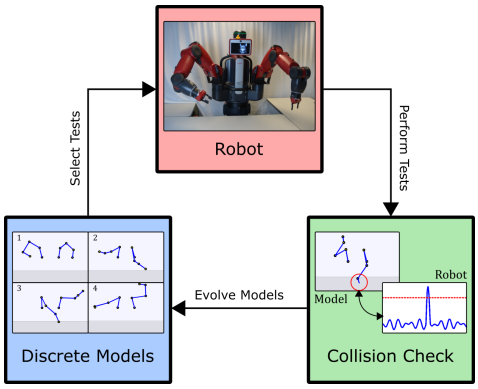

Fig. 1. Proposed method for obtaining unknown kinematic information given a discrete set of candidate models. We use model evolution, intelligent test selection, and collision detection sensors to efficiently identify unknown kinematics. Although intended for modular robots, this algorithm can also be applied to more traditional manipulators, such as the pictured Baxter.

schema acquisition [3], [5], [11], [14], [15], we analogously used collision detection sensors to determine the accuracy of candidate models. It is not necessary that these sensors locate where the collision has occurred—rather, they should simply report whether or not a collision is detected. We also leveraged evolutionary algorithms similar to those described by [10] in order to find the correct kinematic model without exploring all of the discrete candidates. Finally, we introduced a tunable sampling-based planner that considers the bound on position errors when selecting which tests best distinguish among candidate models. Section II more comprehensively describes our problem formulation by detailing the robot, workspace, collision detection, kinematic models, and two specific applications. In Section III we overview the proposed algorithm, while Section IV includes simulations implementing this algorithm on a model Baxter robot.

II. Problem Formulation and Applications

We here consider a serial robot composed of multiple, independently controlled, reconfigurable modules, where the kinematics of each individual module are known. The serial robot assembled from N modules has n total joints, $N \leq n$. Each joint may be either revolute or prismatic, where the joint positions are denoted by $q = [q_1, q_2, \ldots, q_n]^T$, and q is an element of configuration space $\mathcal{Q}$. Joint i is physically constrained by limits $q_{max,i}$ and $q_{min,i}$ and is measured using some on-board sensor—such as an encoder—with accuracy $\pm\Delta q_i$. A desired trajectory $\dot{q}_d(t)$ can be input; the independent, decentralized, module-level controllers then drive each joint along this reference path.

Collision detection senors within each module are used in order to determine whether the physical robot has collided with any boundary of the free configuration space. Alternatively, a fault detection scheme may be leveraged; for example, Geravand *et al.* [16] proposed a high-pass filter on joint torques, in conjunction with thresholding, to discern unexpected contacts without sensors. Let $c \in \{0, 1\}$ represent the output of our chosen collision detector, where $c = 1$ when a collision is perceived and $c = 0$ otherwise; if at q_0 the manipulator does not intersect any obstacle, we can monitor $c(t)$ to confirm that a path from q_0 to q is collision free. Note that obstacles include both objects in the workspace and intersections among the robot's modules. Since we are only using touch sensors, we here must assume that the workspace is known; in future work, we hope to lessen this assumption through integrating visual sensors.

The serial robot's forward kinematics are formulated as

$$X = f(q, \phi) \tag{1}$$

where $X \in SE(3)$ is a transformation from the base frame to the end-effector frame of the robot, specifying the robot's shape. Candidate model $\phi \in \Phi$ is a guess of the unknown kinematic information; for applications with M discrete options for ϕ, the set of all possible candidate models is given by $\Phi = \{\phi_1, \phi_2, \ldots, \phi_M\}$, and the superscript * indicates a correct model. Thus, $f : \mathcal{Q} \times \Phi \rightarrow SE(3)$ serves as the "body schema" mapping between discrete candidate models and robot pose. When the joint position q is given, we can now compute the expected robot pose $f(q, \phi_i)$ corresponding to an arbitrary candidate model ϕ_i. Next, we check if this proposed robot shape collides with any obstacles by running a collision detection algorithm; if a collision is found, we let $c_{m,i} = 1$, otherwise we let $c_{m,i} = 0$, where $c_{m,i}$ indicates the output of the collision check for model ϕ_i. Repeating this process for multiple candidate models, we obtain a vector a predicted collisions, c_m, where each entry in the vector is associated with a particular candidate model, and specifies whether that model predicts a collision. Finally, by comparing the collision sensor, c, and the vector of predicted collisions, c_m, over a sufficiently small interval in $\mathcal{Q}$, we can assess the relative accuracy of our candidate models.

A. Applications

We will focus on two example applications: identifying the order of the modules with respect to a home frame, and identifying the axis of rotation of each module with respect to the previous module. When determining the relative ordering, we seek find the sequence in which to multiply the N known module transformations such that the resultant forward kinematics match the actual serial robot. Let $X_i \in SE(3)$ be the kinematics of module i, and let candidate model ϕ be a permutation of the integers 1 to N, denoted $\phi = perm(1, 2, \ldots, N)$. In this case, we can re-write (1) as

$$f(q, \phi) = X_{\phi_1}(q) X_{\phi_2}(q) \ldots X_{\phi_N}(q) \tag{2}$$

Since the set Φ must accommodate all permutations of the sequence $(1, 2, \ldots, N)$, there are $M = N!$ discrete options for ϕ, and therefore $N!$ potential robot poses at a given joint position. Of course, this makes it impractical to examine every candidate model as the number of modules increases.

Identifying the axis of rotation (or translation) of each module with respect to the previous module or a global frame is a more common kinematic identification problem, and is explicitly addressed in the continuous case by [6], [7], [9]. For this work, however, we consider the unique situation where there are only finite discrete choices for each axis of

rotation, as dictated by the module's geometry and mechanical attachments. Given that the global frame is assigned such that a principal axis aligns with the known axis of rotation of the robot's first module, let us assume that, in some home position, the remaining modules' axes of rotation are parallel to either the x, y, or z-axis of our global frame. Emulating [17], we here denote an axis of rotation as $\omega \in \mathbb{R}^3$, $\|\omega\| = 1$. Due to our assumption, $\omega \in \{[1,0,0]^T, [0,1,0]^T, [0,0,1]^T\}$, and the candidate model ϕ is a matrix of N predicted axes of rotation, $\phi = [\omega_1^*, \omega_2, \ldots, \omega_N]$, where the first axis of rotation is known. Retaining our previous notation, for this application we can re-write (1) to be

$$f(q, \phi) = X_1(q, \phi_1^*) X_2(q, \phi_2) \ldots X_N(q, \phi_N) \qquad (3)$$

Since the geometry from base to tip of individual modules is known, specifying orientation offsets between consecutive modules is sufficient to formulate the forward kinematics. Hence, the set Φ is composed of all $M = 3^{N-1}$ options for the $3 \times N$ matrix ϕ, again suggesting that it might be impractical to examine every candidate model.

III. Proposed Algorithm

Our goal is to develop a method with which to efficiently identify ϕ^*, the correct candidate model that contains the unknown discrete kinematic information, given the modular robotic system described by Section II. In cases with few possible models, the correct candidate model can be found by comparing the collision sensor data to the collisions predicted by every $\phi \in \Phi$; as M increases, however, it becomes computationally infeasible to explicitly examine all discrete candidate models. Both the discontinuity of Φ (the discrete candidate model space) and nonlinearities in c_m (the collision checking logic) prohibit the use of gradient descent or least squares estimation techniques, such as those employed by [6]–[8], [11]. Moreover, since the type of unknown model information—and therefore the form of ϕ—may vary among applications, the proposed search procedure should readily adapt to different situations.

In order to find ϕ^* without having to check every $\phi \in \Phi$, we utilized bio-inspired evolutionary algorithms [18]. More specifically, we modified the hybrid coevolution process described by Bongard and Lipson [19]: alternately developing candidate models and informative tests. A population of models is here evolved based on relative fitness. The test which best differentiates among the current model population is carried out by the robot, and the models which accurately explain the collected data then serve as parents during the next iteration. Within this generic algorithm outline, however, we introduced specialized mechanisms that evolve candidate models and select test trajectories while considering both our robot's collision detection capabilities and the discrete set of candidate models, as shown in Fig. 2.

A. Candidate Model Evolution

Let $\Phi_m(t) \subseteq \Phi$ denote the population, or set, of $\mu + \lambda$ total candidate models which we are examining at iteration t, where $\mu + \lambda \leq M$. Although discrete versions of particle swarm optimization [20] or artificial bee colony [21] algorithms could alternatively be used to decide how the candidate model population changes over time, we here implement classical evolution strategies [18] to better investigate our algorithm's baseline performance. After each test trajectory carried out by the robot, the fitness vector h of the candidate model population is updated as shown in [19]

$$h = h_{prev} + \left| c \cdot \vec{1} - c_m \right| \qquad (4)$$

where c is the scalar output of the collision detection sensor, $\vec{1}$ is the all-ones vector of dimension $\mu + \lambda$, and c_m is the vector of collision predictions corresponding to each candidate model $\phi \in \Phi_m$. Because fitness h is error-based, it is inversely related to performance; we should therefore evolve Φ_m to minimize the elements of h.

When selecting the μ candidate models from $\Phi_m(t)$ which will produce offspring, we (a) select the fittest model as a parent, (b) select the $\mu - 1$ remaining parent models using deterministic tournament selection, and (c) ensure no candidate model is selected to become a parent more than once per iteration. The chosen parents then produce λ children through application-specific mutation operators with a low probability of random seeding (examples in Section IV). Proposed children are compared to the model bank—comprised of the candidate models already examined—and are retained only if they have not been previously considered. Finally, both parent and children models build the next population of candidate models, $\Phi_m(t+1)$, yielding a $(\mu + \lambda)$ evolution strategy [18]. Thus, the number of candidate models in Φ_m is constant throughout the evolutionary process.

Because we have changed the original algorithm to only allow candidate models a single introduction into Φ_m, it is important that candidate models which accurately predict the robot's collision sensor data are not inadvertently rejected. Indeed, since we guaranteed that the best candidate model in $\Phi_m(t)$ is also a member of $\Phi_m(t+1)$, we can prevent the most accurate models from being lost during the evolutionary process. One motivation for preventing discrete models from being re-entered into the candidate population is that it would be inefficient to keep examining the same poorly performing kinematics while ignoring untested candidate models, especially when better candidates have already been identified. A second motivation is to escape local minima; by ensuring that previously unconsidered candidate models are introduced at every iteration, we prevent the candidate model population from converging until ϕ^* is found.

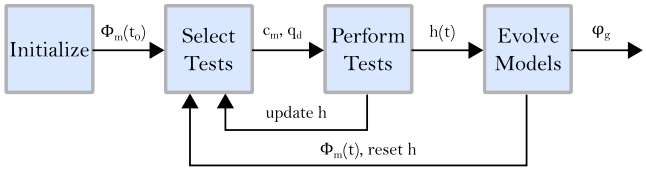

Fig. 2. Outline of proposed algorithm. Several test trajectories may be selected and performed within the inner loop before advancing to evolve candidate models; here h is a vector of candidate model fitnesses, t indicates the iteration count of the outer loop, and ϕ_g denotes the best candidate model currently identified by the search procedure.

B. Robotic Test Selection

We here describe our method for choosing a new desired joint position, q_d, which, when reached using the individual modules' controllers, causes differences in the candidate model population's fitness. Ideally, for some $\phi \in \Phi_m$, the robot pose $f(q_d, \phi)$ collides with an obstacle, while for other $\phi \in \Phi_m$, $f(q_d, \phi)$ lies in the free space. Then, regardless of whether a collision occurs while the robot carries out test trajectory q_d, evaluating (4) will cause the entries of h to change relative to one another, helping us determine which candidate models are the most accurate. To find a test trajectory q_d that satisfied these requirements, we used a straightforward sampling-based planner [22]. Let α be a tunable parameter that determines how many different q_d are proposed, and let q_d be randomly chosen such that $q_{min} \leq q_d \leq q_{max}$, where q_d is a sufficiently short distance `step_size` from the current joint position, q_0. Using the collision detection algorithm for every $\phi \in \Phi_m$, a collision prediction vector c_m can be assigned to each proposed q_d. Finally, the q_d that maximizes the variance in c_m among the γ most accurate candidate models, $\gamma \leq \mu + \lambda$, is selected as the next test trajectory.

While calculating the collision prediction vector c_m associated with a given q_d, joint sensor accuracy should be taken into consideration. Recall that the joints are measured using sensors with accuracy $\pm\Delta q$; during cases where Δq is large and/or the robot has many joints, these sensors may report that the robot has achieved q_d, when in actuality the robot's joint position is q. But then, for some $q \neq q_d$, we might find that c_m corresponding to q_d does not match c_m at q, leading to incorrect predictions. For serial robots, we can write

$$\|\Delta X\| \leq \|\Delta q\| \cdot \sigma_{max} \tag{5}$$

where σ_{max} is the greatest singular value of the robot's Jacobian matrix [17]. To account for the variety of possible poses conservatively bounded by (5), we employ stochastic collision checking over the probability density function (PDF) of the joint position sensors. Given the joint sensor PDF, β random joint positions q'_d are accordingly chosen around q_d. After applying collision checks for every combination of ϕ and q'_d, and then normalizing by β, $c_{m,i}$ now indicates the fraction of possible joint positions at q_d for which model ϕ_i predicts a collision. Moreover, as β (the number of q'_d tested) increases, the predicted collision probability approaches its true value. The resulting procedure used to select informative test trajectories while explicitly considering bounds on joint sensor accuracy is summarized in Algorithm 1.

When the test trajectory found by this algorithm is carried out by the robotic hardware, module-level controllers drive the robot from q_0 to q_d while c is recorded, where q is the actual joint position reached given the joint sensor reading q_d. Let q_0 lie in free configuration space; if no contact is detected during motion, then we update $q_0 = q_d$ and are ready to begin the next test. If, on the other hand, a collision is detected, we use the module-level controllers to drive the robot back to q_0 before preforming the next test. Remember that the distance from q_0 to q_d is small; this ensures that larger motions are composed of smaller steps, and comparisons of collision prediction vs. collision detection occur over these small increments.

Algorithm 1 Procedure to select a test trajectory which causes disagreement among candidate models

Input

Φ_m : the set of candidate models
q_0 : the robot's current configuration
α : the number of joint positions to examine
β : the number of joint variations to examine
γ : the number of candidate models to distinguish

Output

q_d : a desired position in joint space
c_m : a vector of probabilities where $c_{m,i}$ is the expectation with which model i predicts a collision at q_d

$S \leftarrow \emptyset$
for $i = 1$ to α **do**
  $q_{rand} \leftarrow$ a randomly chosen element of $\mathcal{Q}$ bounded by q_{min} and q_{max}
  $q_d \leftarrow$ progress q_0 by `step_size` along the straight line in $\mathcal{Q}$ between q_0 and q_{rand}
  $c_m \leftarrow \emptyset$
  for $j = 1$ to β **do**
    $q'_d \leftarrow$ a randomly chosen configuration selected from the joint sensor PDF about q_d
    for all $\phi_k \in \Phi_m$ **do**
      $X_k \leftarrow f(q'_d, \phi_k)$
      if `CollisionCheck`(X_k) **then**
        $c_{m,k} \leftarrow c_{m,k} + 1/\beta$
      end if
    end for
  end for
  $S \leftarrow S \cup \{(q_d, c_m)\}$
end for
return the element of S where c_m predicts the greatest variance among the γ best performing models

C. Interaction between Models and Tests

Unlike [19], here the fitness of new candidate models is not based on previous test results; instead, we re-initialize h after each iteration of evolution. Practically, calculating model fitness for a consistent number of test results helps the algorithm maintain a constant runtime. From a biological standpoint, it could also be argued that "active learning" (i.e., performing motions to examine kinematic models) more closely resembles both the agency and plasticity of body schema development than does an extensive memory (i.e., remembering previous motions to evaluate current models) [1]. Of course, updating candidate model fitness without using the entire collision detection dataset will likely increase the total number of physical experiments required before ϕ^* is identified; it may thus be reasonable to adjust this forgetting factor in response to testing cost, computational time, and study motivation.

IV. Simulations

We simulated the algorithm outlined in Section III on the arms of a Baxter robot (Rethink Robotics). Although Baxter is not a modular robot—the intended application of our work—we thought it would be helpful to perform simulations using a robot with which many people are familiar so that our results could be more easily replicated. Both of Baxter's arms have 7 joints. We here considered each joint and subsequent link to be a module ($N = 7$ per arm), and then attempted to identify the order of the modules with respect to a home frame (order), and the axis of rotation of each module with respect to the previous module (axes), i.e., the example applications described in Section II-A. Simulations were performed in MATLAB (MathWorks); the robot's geometry and joint sensor parameters were derived from [23]. Our environment consisted of a table and wall (see Fig. 1), so collision checks monitored interactions with these planar obstacles as well as intersections among modules.

During the order simulations, we assumed that the N modules which belonged to each arm were known, but the relative order of those modules was unknown. Hence, there were $M = (N!)^2$ discrete candidate models $\phi \in \Phi$ for the unknown kinematics, and ϕ was a permutation of the module order along both arms. During the axes simulations, the first module's axis of rotation was known and aligned with a global frame. Then, there were $M = 3^{13}$ discrete candidate models $\phi \in \Phi$ for the remaining 13 axes of rotation, and each ϕ consisted of a different guess for these unknown kinematics. We used exchange operators to create mutations in the children of a parent candidate model; for the ordering simulations, each module position in ϕ had a $1/N$ chance of being swapped with another randomly chosen module position, and for the axes simulations, axes 2 through N in ϕ had a $1/(N-1)$ chance of being replaced with a new and different axis of rotation.

Candidate population size ($\mu+\lambda$) and α, the number of q_d proposed when choosing a new test trajectory, were varied between simulations. When $\alpha = 1$, only a single q_d was proposed, and thus each test trajectory was simply a random motion. When $\alpha > 1$, however, the test trajectory was selected using Algorithm 1 in order to best distinguish the fittest candidate models. Other simulation parameters—listed in Table I—were held constant throughout the simulations.

TABLE I

Parameters used when simulating the proposed algorithm

Trials	100
Max. Iterations / Trial	50
Children / Parent (λ/μ)	5
Tournament Size	$\mu+\lambda$
Test Traj. / Iteration, Order	30
Test Traj. / Iteration, Axes	10
Joint Variations (β)	5
Models to Differentiate (γ)	$\frac{1}{3}(\mu+\lambda)$

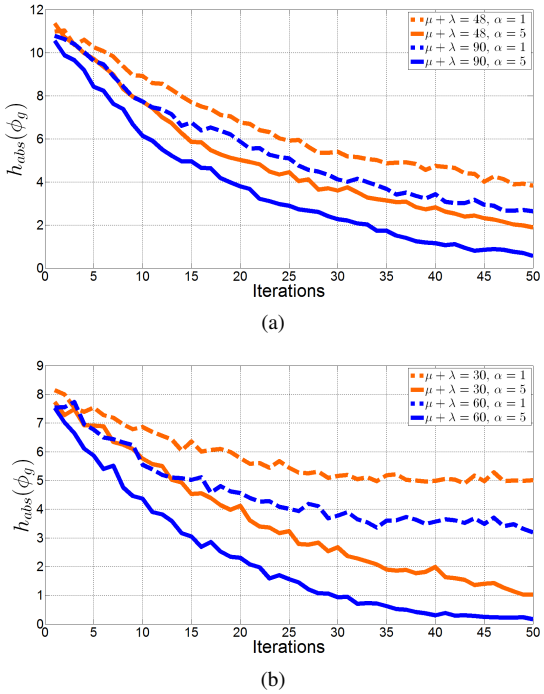

Fig. 3. Error of the fittest candidate model (averaged across all trials) as a function of iteration number. Applications: (a) identifying module ordering, (b) identifying module axes of rotation. Increasing α (solid lines) reduced the number of incorrectly identified elements when compared to completely random motions ($\alpha = 1$, dotted lines). Increasing population size (orange vs. blue) also improved the algorithm's performance.

An iteration consisted of a series of test trajectories (30 during the ordering simulations, 10 during axes simulations) followed by candidate model evolution. Each simulation contained 100 trials, where individual trials were terminated after either ϕ^* was found or 50 iterations occurred. Trials were completed in less than 10 min on a destop computer.

Simulation results are depicted in Fig. 3 and listed in Table II. Data is presented in the form $mean \pm std$ where applicable. Here candidate model ϕ_g denotes the fittest candidate model identified during the final iteration of a trial (i.e., our best current guess of the unknown kinematics), and $h_{abs}(\phi_g)$ denotes the number of elements of ϕ_g which are incorrectly identified. Candidate model rate is the total number of candidate models examined divided by M, the total number of candidate models. Our results demonstrate that the proposed algorithm can be used to correctly identify ϕ^*, the unknown discrete kinematics, while exploring a small percentage of the discrete candidate model space. The unknown module ordering was successfully identified in 84% of trials while reducing the search space by 4 orders of magnitude, and the unknown module axes of rotation were successfully identified in 94% of trials while reducing the search space by 3 orders of magnitude.

We found that our proposed test selection procedure was more effective than purely random motions; increasing α from 1 to 5 increased success rates by an average of 45%, while also reducing both the number of candidate models examined and tests trajectories carried out by the robot. On the other hand, we note that when $\alpha = 5$, there was

TABLE II

SIMULATED PERFORMANCE OF THE PROPOSED ALGORITHM USING DIFFERENT CANDIDATE POPULATION SIZES AND TEST SELECTION METHODS

	$\mu+\lambda$	α	Success Rate (%)	$h_{abs}(\phi_g)$	Cand. Models	Cand. Model Rate (%)	Tests Performed	Total Test Length [rad]
Identifying Module Order	48	1	27	3.83 ± 2.86	1886 ± 328	0.0074	1387 ± 235	2311 ± 383
		5	49	1.88 ± 2.19	1698 ± 478	0.0067	1252 ± 348	2595 ± 735
	90	1	43	2.63 ± 2.81	3286 ± 780	0.0129	1291 ± 300	2152 ± 507
		5	84	0.55 ± 1.42	2585 ± 935	0.0102	1023 ± 368	2203 ± 814
Identifying Module Axes	30	1	13	5.01 ± 2.83	1210 ± 211	0.075	473.1 ± 81.4	785.3 ± 139.3
		5	65	1.03 ± 1.77	934 ± 338	0.059	368.1 ± 131.5	722.5 ± 262.6
	60	1	30	3.19 ± 2.78	2252 ± 565	0.141	441.4 ± 109.2	734.4 ± 182.8
		5	94	0.17 ± 0.80	1311 ± 600	0.082	259.5 ± 118.7	518.1 ± 244.7

consistently larger variability among trials in terms of models examined, tests performed by the robot, and total test trajectory length. Results also show that increasing the candidate model population size ($\lambda+\mu$) leads to smaller errors and less robotic motions, indicating a trade-off between the number of models examined and the number of test trajectories carried out by the robot.

V. CONCLUSION

In this paper we focused on autonomously identifying unknown robot kinematics when only discrete candidate models are present. To resolve this problem, we drew from the biological concepts of body schema and evolutionary algorithms, and used contact sensors to determine the accuracy of candidate models. Because there may be too many candidate models for each to be individually examined, we modified an existing co-evolutionary algorithm to efficiently guide our search, while still guaranteeing that the correct candidate model will be eventually identified. We also introduced a procedure to choose more useful test trajectories while explicitly considering joint accuracy limits. Simulations of the resultant algorithm on a Baxter robot with two different example applications validated both the discrete candidate evolution and test selection process. While this paper represents a first-pass solution, in future work we would like to implement our algorithm on modular robots, refine the described algorithm to deal with encountered hardware difficulties, and explore how different obstacle configurations affect algorithm performance.

REFERENCES

[1] M. Hoffmann, H. G. Marques, A. H. Arieta, H. Sumioka, M. Lungarella, and R. Pfeifer, "Body schema in robotics: A review," *IEEE Trans. on Autonomous Mental Development*, vol. 2, no. 4, pp. 304–324, 2010.

[2] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: A survey," *Cognitive Processing*, vol. 12, no. 4, pp. 319–340, 2011.

[3] M. Asada, K. Hosoda, Y. Kuniyoshi, H. Ishiguro, T. Inui, Y. Yoshikawa, M. Ogino, and C. Yoshida, "Cognitive developmental robotics: A survey," *IEEE Trans. on Autonomous Mental Development*, vol. 1, no. 1, pp. 12–34, 2009.

[4] J. Schwoebel and H. B. Coslett, "Evidence for multiple, distinct representations of the human body," *Journal of Cognitive Neuroscience*, vol. 17, no. 4, pp. 543–553, 2005.

[5] P. D. Marasco, K. Kim, J. E. Colgate, M. A. Peshkin, and T. A. Kuiken, "Robotic touch shifts perception of embodiment to a prosthesis in targeted reinnervation amputees," *Brain*, vol. 134, no. 3, pp. 747–758, 2011.

[6] J. M. Hollerbach and C. W. Wampler, "The calibration index and taxonomy for robot kinematic calibration methods," *The International Journal of Robotics Research*, vol. 15, no. 6, pp. 573–591, 1996.

[7] M. Hersch, E. Sauser, and A. Billard, "Online learning of the body schema," *International Journal of Humanoid Robotics*, vol. 5, no. 2, pp. 161–181, 2008.

[8] R. Martinez-Cantin, M. Lopes, and L. Montesano, "Body schema acquisition through active learning," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2010, pp. 1860–1866.

[9] J. Sturm, C. Plagemann, and W. Burgard, "Body schema learning for robotic manipulators from visual self-perception," *Journal of Physiology-Paris*, vol. 103, no. 3, pp. 220–231, 2009.

[10] J. C. Bongard, V. Zykov, and H. Lipson, "Resilient machines through continuous self-modeling," *Science*, vol. 314, no. 5802, pp. 1118–1121, 2006.

[11] S. Fuke, M. Ogino, and M. Asada, "Body image constructed from motor and tactile images with visual information," *International Journal of Humanoid Robotics*, vol. 4, no. 2, pp. 347–364, 2007.

[12] R. Saegusa, G. Metta, G. Sandini, and L. Natale, "Developmental perception of the self and action," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 183–202, 2014.

[13] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian, "Modular self-reconfigurable robot systems [grand challenges of robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 43–52, 2007.

[14] J. Medina and H. B. Coslett, "From maps to form to space: Touch and the body schema," *Neuropsychologia*, vol. 48, no. 3, pp. 645–654, 2010.

[15] H. E. Van Stralen, M. J. E. Van Zandvoort, and H. C. Dijkerman, "The role of self-touch in somatosensory and body representation disorders after stroke," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 366, no. 1581, pp. 3142–3152, 2011.

[16] M. Geravand, F. Flacco, and A. De Luca, "Human-robot physical interaction and collaboration using an industrial robot with a closed control architecture," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2013, pp. 4000–4007.

[17] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL, USA: CRC, 1994.

[18] F. Neumann and C. Witt, *Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity*. New York, NY, USA: Springer, 2010.

[19] J. C. Bongard and H. Lipson, "Nonlinear system identification using coevolution of models and tests," *IEEE Trans. on Evolutionary Computation*, vol. 9, no. 4, pp. 361–384, 2005.

[20] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. on Neural Networks*, 1995, pp. 1942–1948.

[21] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[22] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementation*. Boston, MA, USA: MIT Press, 2005.

[23] Rethink Robotics, "Baxter research robot: Hardware specifications," http://sdk.rethinkrobotics.com/wiki/Hardware_Specifications, 2015.