Memetic Evolution for Generic Full-Body Inverse Kinematics in Robotics and Animation

Sebastian Starke, *Student Member, IEEE*, Norman Hendrich, *Member, IEEE*, and Jianwei Zhang, *Member, IEEE*

***Abstract*—In this paper, a novel and fast memetic evolutionary algorithm is presented which can solve fully constrained generic inverse kinematics with multiple end effectors and goal objectives, leaving high flexibility for the design of custom cost functions. The algorithm utilizes a hybridization of evolutionary and swarm optimization, combined with the limited-memory-Broyden–Fletcher–Goldfarb–Shanno with bound constraints algorithm for gradient-based optimization. Accurate solutions can be found in real-time and suboptimal extrema are robustly avoided, scaling well even for greatly higher degree of freedom. The algorithm provides a general framework for bounded continuous optimization which only requires two parameters for the number of individuals and elites to be set, and supports adding additional goals and constraints for inverse kinematics, such as minimal displacement between solutions, collision avoidance, or functional joint relations. Experimental results on several industrial and anthropomorphic robots as well as on virtual characters demonstrate the algorithm to be applicable for solving complex kinematic postures for different challenging tasks in robotics, human-robot interaction and character animation, including dexterous object manipulation, collision-free full-body motion, as well as animation post-processing for video games and films. Implementations are made available for Unity3D and robot operating system.**

***Index Terms*—Artificial intelligence, character animation, collision avoidance, evolutionary computation, full-body motion, inverse kinematics, optimization, robotics.**

Manuscript received December 18, 2017; revised June 27, 2018; accepted August 12, 2018. Date of publication August 29, 2018; date of current version May 29, 2019. This work was supported in part by the German Research Foundation (DFG) and in part by the National Science Foundation of China (NSFC) through the Project Crossmodal Learning, TRR 169. The work of S. Starke was supported by the Principal's Career Development Ph.D. Scholarship from the University of Edinburgh. *(Corresponding author: Sebastian Starke.)*

S. Starke is with the School of Informatics, University of Edinburgh, Edinburgh EH8 9AB, U.K. (e-mail: sebastian.starke@ed.ac.uk).

N. Hendrich and J. Zhang are with the Faculty of Mathematics Computer Science and Natural Sciences, TAMS Group, University of Hamburg, 20146 Hamburg, Germany.

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org provided by the author. This consists of a video which demonstrates the performance of the proposed memetic evolutionary algorithm for generic full-body inverse kinematics. Experiments are shown for different application examples in robotics and animation, including full-body motion, dexterous manipulation, teleoperation, collision-avoidance, adaptive foot-placement and animation post-processing. This video is a MP4 file that can be watched with VLC and all other modern video players. This file is 36.8 MB in size.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2018.2867601

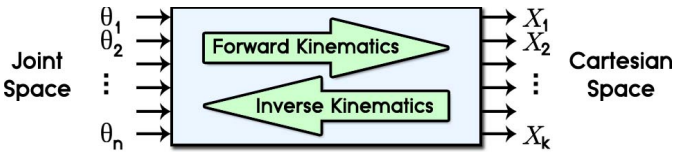

Fig. 1. Relation between forward and inverse kinematics.

I. Introduction

KINEMATICS outlines one of the most fundamental aspects in the general field of motion, covering the transformation of objects with respect to position and orientation, as well as velocity and acceleration, but regardless of mass, force, and torque. It has particular relevance for various applications in robotics and computer graphics, including the design and control of robots, visualization and simulation, as well as for creating and post-processing animations of virtual characters.

A. Articulated Kinematic Geometry

The geometry of an articulated kinematic system can be described by a set of kinematic chains, each consisting of a consecutive set of links and joints from the root to the end effectors. Each end effector results in a certain Cartesian configuration $\mathcal{X}$ given a specific joint variable configuration θ. The alignment of the joint axes define the either translational or rotational motion of the joints, and all together the degree of freedom (DOF) which often indicates the computational complexity of the whole kinematic system. The robotic design of industrial manipulators usually involves a six DOF serial kinematic chain to reach full Cartesian poses in position and orientation. Manipulators with higher DOF can be described as hyper-redundant, mostly providing a continuous range of joint configurations to reach identical poses [1], [2].

Particular challenges arise for human-like anthropomorphic geometries, which have multiple connected kinematic chains, and who are directly controlled by their preceding joints, such as two arms affected by the configuration of the pelvis and torso. The fundamental relation to map between joint and Cartesian space is given by forward and inverse kinematics, and is visualized in Fig. 1.

B. Forward Kinematics

The computation of forward kinematics (1) relates to the mapping from joint to Cartesian space. Using the joint variables $\theta_{1,\ldots,n}$ to recursively calculate the coordinate

Fig. 2. Proposed algorithm solving generic full-body inverse kinematics for different geometries and application scenarios.

transformations along the kinematic chains gives the resulting end effector transformations $\mathcal{X}_{1,\dots,k}$. There exist numerous ways to realize the transformations through using either homogeneous matrices or quaternion-vector-based representations. In particular, choosing quaternions should be preferred in order to avoid the related singularity and gimbal lock issues of Euler angles, and also requires less operations to be computed for each frame transformation

$$f(\theta) = \mathcal{X}_{1,\dots,k}. \tag{1}$$

C. Inverse Kinematics

In contrast to forward kinematics, obtaining a solution for inverse kinematics (2) is not straightforward, and zero up to infinite solutions can exist in joint space to satisfy a Cartesian configuration for position and orientation. The further presence of joint limits requires suboptimal extrema to be avoided, which typically turns inverse kinematics into a nonconvex optimization problem. Finally, including additional objectives and constraints to prefer particular neighboring or collision-free solutions, as well as to implement nonlinear couplings between joints in a soft-constraint fashion is not generally available for existing methods. However, handling such structures and complex posture specifications is essentially required to solving full-body motion on humanoid robots, advances in dexterous manipulation with anthropomorphic hands, as well as for creating more realistic motion of animated characters

$$\theta = f^{-1}\left(\mathcal{X}_{1,\dots,k}\right). \tag{2}$$

While both domains have their own algorithms and methods developed and applied for their specific needs, their benefits and limitations in terms of computational cost, robustness, scalability, and flexibility to control complex geometries tend to remain mutually exclusive. Finally, the question remains how to integrate additional goals which specify the solution quality for a particular task, such as for grasping, interaction or natural motion. The motivation of this paper is to propose an efficient solution for inverse kinematics that combines multiple objectives into a single-objective problem, and can serve the various aspects in performance requirements both in robotics and character animation on generic kinematic geometries. Although general approaches like evolutionary algorithms are well suited for such demands, they have so far only been applied relatively scarcely to solving inverse kinematics problems. In this paper, a memetic evolutionary approach combining the characteristic strengths of hybrid biologically inspired and gradient-based optimization is presented. As demonstrated in Fig. 2, our algorithm can be successfully applied to different types of kinematic geometries and application scenarios.

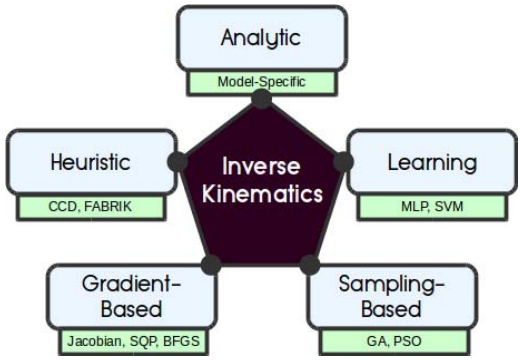

Fig. 3. Algorithmic methodologies for inverse kinematics.

II. Related Work

Although inverse kinematics is well researched on serial kinematic chains, many methods still seem rather limited in solving full-body postures on more complex geometries or while incorporating multiple constraints. In particular, no general solution to this problem could yet be found. The various sophisticated approaches originate from very different research areas, and which can be divided as shown in Fig. 3.

A. Analytic

Given a particular kinematic geometry, analytic methods can be constructed specifically to provide closed-form expressions. The obtained solutions are exact, and can also return all existing joint configurations for a desired reachable Cartesian pose. The IKFast [3], [4] module is a recent implementation which automatically generates the algebraic equations. Although these methods are often significantly faster than related approaches, they are typically only available for rather simple geometries, as the complexity rapidly increases with each additional DOF. Hence, they are rather used for serial industrial manipulators and impractical for solving multiple and highly articulated chains simultaneously. In this context, larger focus has been paid to iterative and numerical methods.

B. Heuristic

Heuristic methods for inverse kinematics aim to iteratively estimate the required joint updates using geometric calculations. The cyclic coordinate descent (CCD) [5], [6] is a traditional and one of the most popular related algorithms. It is simple to implement and only requires little computational cost per iteration since each joint along the kinematic chain is updated independently, iterating from the last to the first link. Therefore, it is predominantly applied in computer graphics and for animation tasks where motion on multiple characters must be generated in real-time. The calculations can be done by using dot and cross product operations between the positions of the end effector and the current joint p_{ee} and p_i with respect to the target position p_t, where (3) gives the angular change $\Delta\theta_i$ of the current joint about the vector $\vec{n}$

$$\Delta\theta_i = \cos^{-1}\left(\frac{p_{\text{ee}} - p_i}{||p_{\text{ee}} - p_i||} \cdot \frac{p_t - p_i}{||p_t - p_i||}\right) \tag{3}$$

$$\vec{n} = \frac{p_{\text{ee}} - p_i}{||p_{\text{ee}} - p_i||} \times \frac{p_t - p_i}{||p_t - p_i||}. \tag{4}$$

Note that this method only optimizes the position of the end effector and becomes more complex if solving an orientation goal is also desired. Additionally, it can be observed to produce unrealistic motion as it tends to overestimate particular joints along the kinematic chain. Since CCD operates locally on each joint, it is rather difficult to efficiently combine multiple kinematic chains, and working in Cartesian space can finally cause joint limits to be violated without further post-processing.

Forward and backward reaching inverse kinematics (FABRIK) [7], [8] is a more recent algorithm which has quickly gained high relevance in character animation. Instead of traversing a single direction along the chain, it consists of two consecutive phases in an iterative forward and backward reaching mode. Let $\lambda_i = (d_i/r_i)$ be defined by the initial and new distances d_i and r_i between two joint positions p_i and p_{i+1} with $i = 1, \ldots, n-1$ and p_n being the target position of the end effector, then the forward (5) and backward (6) calculations iteratively optimize the required joint locations $p_{1,\ldots,n}$ to reach a desired position for the end effector

$$p_i = (1 - \lambda_i)p_{i+1} + \lambda_i p_i \tag{5}$$

$$p_{i+1} = (1 - \lambda_i)p_i + \lambda_i p_{i+1}. \tag{6}$$

Thus, articulated postures are solved through finding points on lines rather than calculating rotational joint updates. This technique is extremely fast, scalable, and provides several advantages over CCD, as it can efficiently solve multiple end effectors simultaneously. However, extending it to support orientational constraints and joint limits again makes it rather complex to deploy for more challenging needs, and inapplicable for robotics where solutions in joint space are preferred. In more detail, FABRIK differs from other methods in terms that it avoids optimizing particular joint configurations in order to find a posture to reach the target.

Implementations for different variants of CCD and FABRIK are provided by [9] and [10], and are used in modern graphics engines such as Unity3D [11], Unreal [12], and Maya [13].

C. Gradient-Based

In robotics, gradient-based methods for nonlinear optimization are among the most widely applied techniques for solving inverse kinematics. As they require approximating first- or second-order derivatives, they are more computationally expensive than the previous introduced heuristic algorithms. However, they can directly operate in joint space which eases the use of joint limits, and are slightly more flexible to include additional constraints as well as for solving full-pose goals.

Most popular approaches are based on computing the Jacobian (7), which is a matrix of first-order partial derivatives of each joint variable which gives a linear approximation of the resulting end effector velocities in Cartesian space [14]

$$J(\theta)_{ij} = \left(\frac{\delta\mathcal{X}_i}{\delta\theta_j}\right). \tag{7}$$

The challenge then becomes to find an appropriate update of the whole joint variable configuration $\theta' = \theta + \Delta\theta$ such that the error vector $\vec{e}$ is minimized as smoothly and quickly as possible. The most simple and computationally cheap version is to use the transpose method with α defining a small factor to move in direction of the gradient. This solution generates smooth motion, but usually has slow convergence

$$\Delta\theta = \alpha J^T \vec{e}. \tag{8}$$

Using the pseudoinverse (9) provides a significantly faster convergence, but becomes unstable in near-singular configurations which can cause jittery motion and no solutions to be found

$$\Delta\theta = J^T\left(JJ^T\right)^{-1}\vec{e}. \tag{9}$$

A way to avoid such issues is by using the damped least squares method (10). However, choosing the damping constant λ is not trivial in order not to slow down the optimization

$$\Delta\theta = J^T\left(JJ^T + \lambda^2 I\right)^{-1}\vec{e}. \tag{10}$$

An extended version of the Jacobian [15] was used to solve inverse kinematics on humanoid robots, and [16] successfully used it for real-time character animation tasks. An implementation is available by the Orocos KDL framework [17], which is commonly being used with robot operating system (ROS) [18]. Nevertheless, a major drawback of these methods is that they largely suffer from multiple local extrema, which can cause no acceptable solution to be found at all. Improvements could be achieved by introducing heuristic restarts from random initial seeds [19], and combining with sequential quadratic programming returned good results [20] on serial kinematic chains of various industrial and humanoid robots.

Another well-known algorithm is the Broyden–Fletcher–Goldfarb–Shanno (BFGS) [21] which originates from the family of Newton approaches to approximate the Hessian matrix for solving nonlinear optimization problems. Although it is more costly, a particular advantage of this method (11) is that it allows a custom objective function Ω to be minimized by additionally using its gradient, which makes it flexibly applicable and suitable for complex inverse kinematics [22]. It provides smooth convergence and has been observed to

perform notably well under nonsmooth optimization problems [23] compared to Jacobian methods, and does also not suffer from singular configurations

$$\Omega(x+\sigma) \approx \Omega(x) + [\nabla\Omega(x)]^T\sigma + \frac{1}{2}\sigma^T H_\Omega(x)\sigma. \tag{11}$$

D. Sampling-Based

Sampling-based methods, such as genetic algorithms (GAs) [24] and particle swarm optimization (PSO) [25], provide a general and yet efficient technique for bounded nonlinear optimization. They offer maximum flexibility in terms of objective function design, and perform well even under high-dimensional and nonconvex search spaces. In particular, they are more robust in avoiding suboptimal extrema than previous introduced methods, do also not suffer from singularities, and directly operate in joint space through encoding joint variable configurations as individuals. In [26], GA were successfully applied to find multiple existing inverse kinematics solutions on industrial manipulators. A memetic variant of differential evolution (DE) [27] was proposed in [28] for solving more complex anthropomorphic geometries. Further, Aguilar and Huegel [29] demonstrated their computational performance increase when using a parallel implementation, and Stollenga *et al.* [30] were able to generate task-specific motion trajectories with multiple constraints on the iCub humanoid robot. In [31] and [32], PSO was successfully used to solve inverse kinematics on serial kinematic chains, and Collins and Shen [33] demonstrated its scalability on hyper-redundant serial manipulators with up to 120 DOF. However, although such methods can typically achieve a fast and robust initial convergence, the total time to obtain high accuracy is not able to compete with one of gradient-based methods unless further heuristics are used.

E. Learning

Learning methods have also been applied to approximate the inverse kinematics function between joint and Cartesian space. Once this function is learned, solutions can be generated very quickly and repeatedly. However, since inverse kinematics typically includes multiple solutions for identical Cartesian queries, the learning algorithm might interpolate between multiple training samples. As a result, the learning error often remains too large for many practical applications. Several research using artificial neural networks reports achievable accuracies of approximately 10^{-2}m/rad [34], [35] for lower DOF manipulators, whereas 10^{-3}–10^{-5}m/rad is usually requested in robotics. Though, Almusawi *et al.* [36] recently modified the input to additionally use the current joint variable state of the robot and reported considerably better accuracy. Similar results could also be achieved using support vector machines with an additional spatial decomposition method [37]. However, these methods require individual training for each kinematic geometry and for any change in objectives and constraints, and thus are difficult to use for dynamic tasks.

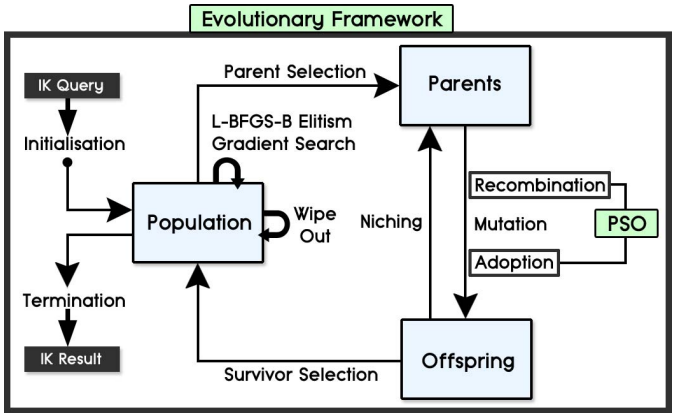

Fig. 4. Scheme of memetic evolutionary algorithm.

III. Algorithmic Approach

The various methods discussed in the previous section all have their particular strengths, but are either suffering from local extrema, high computation time, low scalability, or being difficult to extend for multiple or different objectives. Hence, our intention of solving inverse kinematics generically aims to serve the different aspects, which can be formulated as follows.

1) *Success:* An existing solution for a given accuracy under the defined objectives and constraints can also be found within a specified amount of time.
2) *Accuracy:* The solution shall be as precise as required, typically with an error below 10^{-3} m/rad.
3) *Time:* The solution shall be found as fast as possible, preferably within a very few milliseconds.
4) *Continuity:* The distance between solutions in joint as well as Cartesian space shall be as minimal as possible.
5) *Adaptivity:* Robustness, scalability, and fast convergence can be maintained for varying kinematic geometries.
6) *Flexibility:* The algorithmic methodology allows adding further objectives and constraints, and can be extended in order to fulfil task-specific requirements.

In order to meet these challenges, biologically inspired optimization algorithms can provide robustness and scalability, while gradient-based methods can offer a fast and accurate convergence—which combined result in a memetic evolutionary optimization strategy. Previous work in [38] proposed a hybridization of GA and PSO, but was limited to serial inverse kinematics. Further research in [39] and [40] extended the algorithm to support multiple concurrent goals for solving full-body postures, but was specifically designed for traditional inverse kinematics with pure Cartesian objectives for position and orientation. The contribution of this paper is a generalization to serve a high variety of goals and constraints by enabling the algorithm to use the same objective function for the evolutionary and the additional gradient-based optimization—and thus serving the last desired aspect for flexibility.

The scheme of the proposed algorithm is shown in Fig. 4, which represents an evolutionary framework into which the gradient-based and swarm optimization are directly integrated. Since inverse kinematics constitutes a continuous optimization problem, GA and PSO suggest being combined into a single method. This hybridization is implemented by the additional

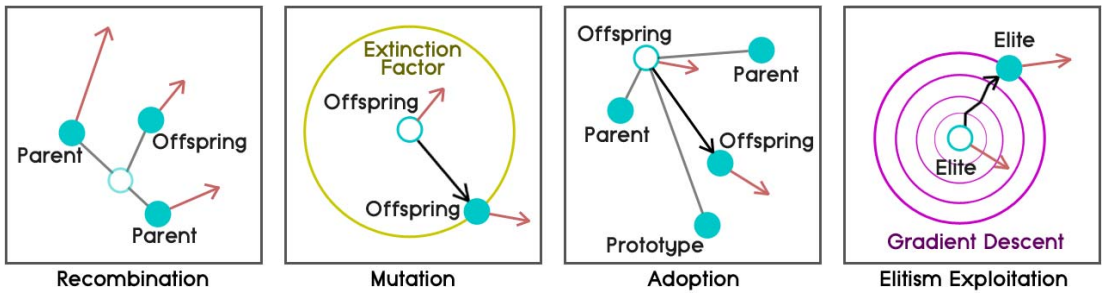

Fig. 5. Visualization of the evolutionary phases.

Adoption phase in combination with the *Recombination* phase, and which integrates a momentum term for the individuals. The momentum is gradually updated within each evolutionary phase which modifies the genes. For the *Mutation* phase, an adaptive extinction operator was designed which reduces the required number of parameters, and can be directly applied to varying problem dimensionalities. The further *Niching* phase aims to explore different regions in the search space simultaneously, along with the *Exploitation* phase which explicitly seeks to improve successful individuals. For this, the limited-memory-BFGS with bound constraints (L-BFGS-B) algorithm is used. BFGS was originally designed for unconstrained optimization with an expensive calculation of the Hessian matrix. The L-BFGS-B has improved computational performance in approximation of the Quasi-Newton method [41], and allows incorporating bound constraints [42], which is required in order not to violate joint limits. Finally, a *Wipe Out* criterion is used to heuristically detect whether all niches are stuck in suboptimal extrema, in which case a partial reinitialization is performed. The different phases for the evolutionary algorithm are visualized in Fig. 5, and will be discussed in more detail throughout the next sections.

A. Joint Variable Encoding

The basic strategy to solving inverse kinematics through evolution is to encode a joint variable configuration θ as the genotype x of an individual. This can be denoted as (12), which uses a real-valued gene representation assigning one gene for each joint giving rise to a n-dimensional search space

$$\theta \mathrel{\hat{=}} x = \left(x_1 \mid x_2 \mid x_3 \mid \ \ldots \ \mid x_{n-1} \mid x_n\right). \tag{12}$$

Further, incorporating joint limit is an essential task when operating on real robots or to achieving realistic postures on virtual characters. Thus, each gene is constrained (13) to the lower and upper limits of its particular joint, and which gets clipped in case of exceeding its bounded search space domain

$$\theta_{i_{\min}} \leq x_i \leq \theta_{i_{\max}} \quad \forall i = 1, \ldots, n. \tag{13}$$

More specifically, each evolved individual directly represents a valid joint variable configuration, which is an advantage over related methods that often require costly post-processing in order not to violate joint limits. The resulting kinematic posture can then be obtained by forward kinematics using (1).

B. Objective Function

The design of the objective function outlines one of the most crucial challenges in evolutionary optimization. Especially if multiple goals shall be solved concurrently, this can become very difficult as the error terms need to be formulated in a sensible relation to each other. In addition, the combination of those should preferably not cause strong fluctuations in the resulting optimization landscape, but rather define smooth transitions. With this in mind, the fitness ϕ of an individual under the objective function Ω that shall be minimized (14) is evaluated using the root-mean-square-error (RMSE) over all single objectives whose loss terms are calculated by $\mathcal{L}$. Each of those can further be weighted by w to achieve customizability for task-specific needs—such as preferring higher accuracy in position over orientation while forcing a joint to keep a particular value

$$\phi = \Omega(x) = \sqrt{\frac{1}{k}\sum_{j=1}^{k} w_j \mathcal{L}_j^2(x)}. \tag{14}$$

Note that by using a squared error metric to combine the different goals into a single-objective optimization problem, the aim is to transform the initial linear problem into a quadratic which introduces smoothness, and eventually leads to a near-convex solution for the optimization. For reference, a performance comparison for both fitness measures is shown in Table II. Information on a proper choice of weights for the objectives will be discussed in the following section.

C. Design of Objectives

All objectives (15) are formulated in a way such that they yield zero error for an optimal solution. Clearly, the most relevant objectives for inverse kinematics are given by position and orientation, together defining a full Cartesian pose. However, while orientation errors are limited, position errors to the specified targets can grow arbitrarily large, and which also depend on the size of the kinematic model. Thus, a normalization is performed, where $d = ||\mathcal{Y}^{\text{pos}} - \mathcal{X}^{\text{pos}}||$ denotes the Euclidean distance between the positions of the target $\mathcal{Y}$ and the resulting transformation $\mathcal{X}$ within the kinematic model, L is the constant length of the kinematic chain, and λ is the variable distance between the root and $\mathcal{X}$ under the evaluated joint variable configuration. Multiplying by π finally yields a position error between $[0, \pi]$, similarly to the orientation error which

is obtained by the quaternion dot product between $\mathcal{X}$ and $\mathcal{Y}$. Thus, the position objective will adapt accordingly to different body sizes, and allows the user to keep equal weights for both position and orientation objectives without requiring to retune the algorithm for different kinematic geometries. Assigning a weight of 1 can be considered as default. Next, sometimes the user desires to have the head or eyes of a character looking at a particular point while moving the rest of the body, or have the camera of a robot arm tracking an object while manipulating it. Such tasks can be solved simultaneously by adding an objective to minimize the angular error between

$$\mathcal{L} = \begin{cases} \frac{\pi d}{\sqrt{(L+d)(\lambda+d)}} & \text{Position} \\ 2\cos^{-1}(\mathcal{X}^{\text{quat}} \cdot \mathcal{Y}^{\text{quat}}) & \text{Orientation} \\ \cos^{-1}(V \cdot (\mathcal{Y}^{\text{pos}} - \mathcal{X}^{\text{pos}})) & \text{Look At} \\ \begin{cases} \infty & \text{if } d - d_{\min} \leq 0 \\ \frac{1}{d - d_{\min}} & \text{else} \end{cases} & \text{Distance} \\ \sum_{i=1}^{n} |x_i^* - x_i| & \text{Displacement} \\ |\theta_i^* - x_i| & \text{Joint Value} \\ |x_i - h(x)| & \text{Joint Function} \end{cases} \tag{15}$$

the direction to the target position and a specified viewing direction V. The weight of this objective can be chosen equally to those for position and orientation. Further, collision avoidance is an issue which is often neglected by generic inverse kinematics solvers. Using the proposed method, self-collision as well as collision with objects in the environment can be avoided through defining an objective which forces the optimization not to fall under a specific distance threshold $d_{\min}$ between two particular points. More specifically, this technique prevents spherical collisions by returning an infinite error if the distance d becomes too small, but acts as a soft-constraint otherwise and is fast to compute. The weight for this objective should usually be comparatively small. Furthermore, a displacement objective is designed which aims to prefer solutions with small distances in joint space. This objective seeks the average variation of x to the current solution x^* to be minimal, and thus avoids noisy or jerky movements. A similar technique can also be applied to single genes, trying to maintain a desired value θ^* for a particular joint. Finally, it is also possible to consider functional relations $x_i = h(x)$ among joints. For example, this can be helpful to describe realistic motion, such as for the little finger of the hand affected by the joint configurations of the other fingers. The weights for those objectives should be chosen comparatively small, i.e., 0.001 to 0.01 depending on the user-specific needs and the total amount of objectives added to the system. In general, all objective weights can be set depending on the preferences of the user. This is a particular advantage of this method as it allows to assign a higher importance for particular parts of the body when solving inverse kinematics. For example, when performing manipulation, the position objectives for the finger tips can be given a higher weight than other optional objectives on the wrist or elbow. This improves the accuracy at the important body parts, and leaves more flexibility for the rest of the body in finding a suitable posture. Similarly, the poses of the feet to get in contact with the ground can be given a specifically higher importance.

D. Initialization

For an inverse kinematics query, the algorithm is initialized according to (16), using $\varphi - 1$ randomly generated individuals where φ denotes the population size, and $U_{[a,b]}$ describes a uniformly distributed random value between a and b. Thus, all genes are randomly sampled between the particular joint limits, and one further individual is created from the currently assigned joint variable configuration θ used as the seed state

$$x^1 = \theta \quad x_i^{2,\ldots,\varphi} = U_{[\theta_{i_{\min}}, \theta_{i_{\max}}]} \quad \forall i = 1, \ldots, n. \tag{16}$$

E. Recombination

The recombination phase creates new offspring from two parents each by using a rank-based selection, which aims to maintain diversity and to avoid dominating individuals. Then, each gene is calculated by a weighted average of both parent genes, where $I_c(a, b)$ is a linear interpolation between a and b weighted by c. In addition, a small amount of the momentum term g of both parents is added, which aims to let offspring immediately dive a little deeper into the direction which likely caused improvement for their parents. This method represents the first step of the introduced hybridization of GA and PSO

$$R : \begin{cases} x_i = I_{U_{[0,1]}}\left(x_i^{\mathcal{P}_1}, x_i^{\mathcal{P}_2}\right) + g_i \\ g_i = U_{[0,1]}\ g_i^{\mathcal{P}_1} + U_{[0,1]}\ g_i^{\mathcal{P}_2}. \end{cases} \tag{17}$$

F. Mutation

For mutation, an operator is required which remains adaptive to varying kinematic geometries whose DOF defines the search space dimensionality. Thus, a higher DOF requires smaller while a lower DOF needs higher probabilities for the mutation rate to cause effective changes along with the mutation strength. A simple technique would be using an inverse-proportional probability for the number of genes, but which is not adaptive to the current progress in optimization. Instead, an extinction operator was designed which uses a factor ξ (18) to adaptively control mutation rate and strength

$$\xi = \frac{\phi + \phi_{\min}\left(\frac{i-1}{\varphi - 1} - 1\right)}{\phi_{\max}} \tag{18}$$

$$M : \begin{cases} p = \frac{\xi^{\mathcal{P}_\varnothing}(n-1)+1}{n} \\ x_i' = x_i + U_{[-1,1]}\frac{\xi^{\mathcal{P}_1} + \xi^{\mathcal{P}_2}}{2}\left(\theta_{i_{\max}} - \theta_{i_{\min}}\right) \\ g_i' = g_i + \left(x_i' - x_i\right). \end{cases} \tag{19}$$

This factor measures the relative success of an individual within the whole population regarding its rank and fitness, and yields a normalized value between [0, 1]. It is then possible to define the mutation phase as (19), where p is the mutation rate between $[(1/n), 1]$, and the change is calculated by a variable random offset using the average extinction of both parents and the domain size. This operator performs small variations for better individuals, but still allows larger changes to be sampled for worse individuals. We experimented with several versions for choosing the offset. However, we observed this method to produce a good tradeoff in fast and robustly finding solutions. A particular advantage is that this operator can adapt automatically to different population sizes, without the need of being tuned again for every kinematic setup.

G. Adoption

The adoption phase is the second step of the GA and PSO hybridization. In this, each individual is updated (20) according to the average of its parents $P_{\varnothing}$ and a further prototype individual P_* which is similarly selected by its rank

$$A: \begin{cases} x_i' = x_i + I_{U_{[0,1]}}\left(U_{[0,1]}\left(x_i^{\mathcal{P}_{\varnothing}} - x_i\right), U_{[0,1]}\left(x_i^{\mathcal{P}_*} - x_i\right)\right) \\ g_i' = g_i + \left(x_i' - x_i\right). \end{cases} \tag{20}$$

In particular, a randomly weighted combination of both directions is added to the current gene with the aim to adopt promising characteristics of successful individuals. Hence, this method directly follows the idea of PSO to update particles.

H. Exploitation

To improve the speed and continuity of convergence, the exploitation phase uses the L-BFGS-B algorithm to enhance potential solutions which are given by the elites. As input, the same objective function Ω as for the evolution is used, together with its approximated gradient $\nabla\Omega$ of partial derivatives through iteratively modifying each gene by a small value (21)

$$\nabla\Omega_i = \left(\frac{\delta\Omega}{\delta x_i}\right). \tag{21}$$

The update can then be denoted as (22). Thus, elite individuals do not only survive, but are also exploited through gradient-based optimization. In more detail, this technique allows the population to exploit multiple local extrema simultaneously, which can be controlled by the chosen number of elites

$$E: \begin{cases} x' = \text{L-BFGS-B}(x, \Omega, \nabla\Omega) \\ g' = U_{[0,1]}g + \left(x' - x\right). \end{cases} \tag{22}$$

I. Niching

Niching is a common technique in evolutionary optimization to let the population discover multiple solutions. Inverse kinematics usually involves multiple local extrema, so niching can significantly improve the robustness of the algorithm. A preselection scheme is used, which removes any parent from the mating pool Ψ whose offspring scored a better fitness value. This can be formulated as (23), and encourages the population to explore different paths simultaneously during optimization instead of forming particular clusters that work independently or try to maintain a distance to each other. Note that in case the mating pool becomes empty, a random offspring is created

$$N: \begin{cases} \Psi \setminus x^{\mathcal{P}_1} & \text{if } \Omega(x) < \Omega\left(x^{\mathcal{P}_1}\right) \\ \Psi \setminus x^{\mathcal{P}_2} & \text{if } \Omega(x) < \Omega\left(x^{\mathcal{P}_2}\right). \end{cases} \tag{23}$$

J. Wipe Out

Although the evolutionary phases are designed to avoid suboptimal extrema, it is still possible that all niches can get stuck in such. In those cases, performing a reinitialization of the population is likely to achieve an overall faster convergence instead of waiting for a successful exploration to occur. The condition (24) for this is met if the current solution x^* could not be replaced by the fittest individual x^1, and if no further fitness improvement could be achieved for any elite individual x^{ε} within the set of elites $\mathcal{E}$

$$W{:}\phi^1 \geq \phi^* \ \wedge \ \phi^{\varepsilon\prime} \geq \phi^{\varepsilon} \ \ \forall\, \varepsilon \in \mathcal{E}. \tag{24}$$

The reinitialization is then performed as described in (16), where the current solution x^* is reintegrated into the new population with $x^1 = x^*$. This achieves a partial reinitialization which does not lose information about the best solution that could be found so far. In particular, this can be considered as an advantage of multimodal sampling-based methods over unimodal gradient-based techniques for optimization, where the latter would need to start from scratch after being reinitalized.

K. Termination

The algorithm finally terminates if either all objectives are satisfied or a specified time limit was exceeded. Instead of using a fitness threshold to check for convergence, defining individual termination conditions for each objective ensures that a minimum accuracy for each particular goal is achieved if the evolution was successful. Those are defined as thresholds in Euclidean or angular space, i.e., 10^{-4} m and 10^{-3} rad, or 5–10 ms when using a time limit. Those conditions are particularly relevant for position, orientation, and direction goals since the fitness values does not directly relate to the actual errors. For the collision-avoidance, the defined value for the distance objective is used. For the remaining objectives, an termination or failure condition is optional. However, note that even if the algorithm did not converge, the best approximate solution that could be found so far can be returned.

L. Pseudocode

The pseudocode of the complete Bio-IK algorithm is shown by Algorithm 1, which again summarizes how the single evolutionary phases are integrated and how the population is evolved.

M. Computational Improvements

Usually, the evaluation of fitness values is the most computationally demanding part in evolutionary optimization. In context of the proposed algorithm, multiple forward kinematics queries must be resolved in order to iteratively optimize the inverse kinematics solution. However, most joint variable configurations are only slightly modified within each generation, and repeated recalculation of the full equations would be inefficient. For this, an optimized forward kinematics tree data structure [43] was developed which decomposes the nonstatic kinematic geometry into a linked list of joints, and avoids redundant transformations and calculations by storing and reusing results from previous queries. Furthermore, the algorithm allows to be parallelized on the CPU to improve computation of independent operations. For each elite individual, an additional thread is reserved to perform the L-BFGS-B exploitation, which is rather costly compared to the other tasks. The remaining population is evolved on the main thread, which must be performed sequentially because of the niching phase which introduces mutual dependency between parents and offspring. In particular, the elitism exploitation on the

Algorithm 1: Bio-IK Algorithm

```
Input : Population Size, Number of Elites, Seed
Assign Seed as Solution;
Initialize Population;
⟩⟩ Incorporate Solution;
⟩⟩ Create PopulationSize − 1 Random Individuals;
⟩⟩ Evaluate and Sort Individuals by Fitness;
⟩⟩ Calculate Extinction Factors;
⟩⟩ Try Update Solution;
while Not Terminated do
    Assign whole Population to the Mating Pool;
    for All Elite Individuals do
        Perform Exploitation using L-BFGS-B;
    end
    for All Non-Elite Individuals do
        if Mating Pool is not empty then
            Select Parents and Prototype from Mating Pool
            and Create new Individual by Recombination,
            Mutation and Adoption;
            Constrain Genes to Dimension Bounds;
            Evaluate Fitness;
            Remove worse Parents from Mating Pool;
        else
            Create Random Individual;
            Evaluate Fitness;
        end
        Add Individual to Offspring;
    end
    Select Elites and Offspring as the new Population;
    Sort Individuals by Fitness;
    Calculate Extinction Factors;
    Try Update Solution;
    if Wipe Out Criterion Fulfilled then
        Reinitialise Population;
    end
end
Return Solution;
```

single threads continues iterating until the main thread has finished evolving individuals so that no computation time remains unused. Thus, the population size implicitly controls the number of exploitation steps within each generation.

N. Available Implementations

1) Unity3D (C#): The Unity3D implementation [44] consists of several modular components which can be added to the transform hierarchy where desired. The software automatically detects the defined kinematic geometry, provides user-friendly custom inspectors and parameter visualization for each component, and includes an API to control the algorithm although it can also entirely be used without any programming.

2) ROS (C++): The ROS implementation [45] is written as a native IK plugin for MoveIt!, and can be directly exchanged for any other existing IK plugin. In contrast to KDL and TRAC-IK, our BioIK plugin can solve multiple goals at once.

IV. Experiments and Results

The experiments were conducted using an ASUS ROG G-751 notebook with 2.6-GHz processor cores, and the available implementation of the algorithm in Unity3D [44] was used. All statistical results were evaluated over 10.000 inverse kinematics independent queries from random initial seeds, leading to unsignificantly small deviations between the results. This section first discusses the parameter selections, then demonstrates the performance on serial and anthropomorphic geometries, and finally shows the algorithm used for different application scenarios in robotics and character animation.

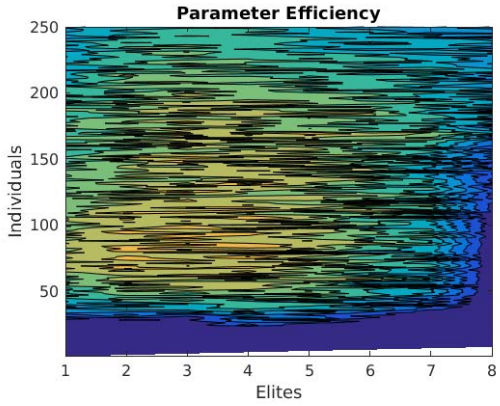

Fig. 6. Parameter efficiency on a 15 DOF serial manipulator.

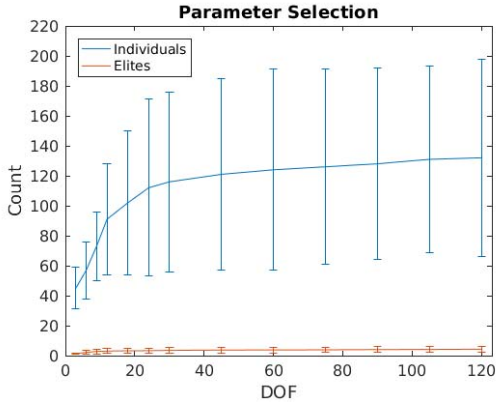

Fig. 7. Suitable parameter selections for increasing DOF.

A. Parameters

The only required parameters for the algorithm are given by the population size and the number of elites. Fig. 6 shows the normalized distribution of efficiency which is measured by the achieved fitness divided by a fixed optimization time of 10 ms.

Investigating these distributions for varying DOF on serial manipulators, Fig. 7 suggests that choosing at least two elite individuals should be preferred. The results further show that increasing the population size has larger impact for smaller DOF, and seems to converge for increasing DOF. In general, it was observed that parameter selections between 50 and 150 individuals with 2–5 elites performed well in the experiments.

B. Serial Manipulators

Initial experiments in order to measure the performance of the algorithm were made on some common robotic arm serial manipulators shown in Fig. 8. The robot models were imported using a unified robot description format parser for Unity3D. Those were chosen by different geometric

TABLE I
PERFORMANCE OF THE ALGORITHM COMPARED TO RELATED GRADIENT-BASED AND EVOLUTIONARY APPROACHES FOR SOLVING INVERSE KINEMATICS ON SERIAL MANIPULATORS. THE RESULTS SHOW THE AVERAGE COMPUTATION TIMES AND SUCCESS RATES OVER 10.000 QUERIES

Manipulator (Full-Pose)	DOF	Jacobian-T (Joint Limit Termination)	Jacobian-DLS (Joint Limit Termination)	L-BFGS-B (Joint Limit Termination)	SGA (1s Timeout)	DE (1s Timeout)	Bio-IK (1s Timeout)	Bio-IK (10ms Timeout)
PR2 Arm	7	4.06 ms (25.27%)	0.34 ms (28.61%)	0.36 ms (37.68%)	334.53 ms (36.24%)	197.65 ms (66.89%)	1.02 ms (100%)	0.91 ms (99.47%)
Baxter Arm	7	8.73 ms (36.92%)	0.59 ms (9.78%)	0.44 ms (43.19%)	362.18 ms (39.22%)	171.39 ms (56.84%)	1.34 ms (100%)	1.12 ms (99.16%)
KuKA LBR iiwa	7	5.27 ms (88.71%)	0.49 ms (68.64%)	0.41 ms (74.33%)	297.87 ms (61.14%)	129.42 ms (87.39%)	0.85 ms (100%)	0.74 ms (99.93%)
KuKA KR120 R2500	6	3.05 ms (58.77%)	0.36 ms (59.62%)	0.39 ms (67.74%)	398.49 ms (52.21%)	231.87 ms (77.17%)	1.33 ms (100%)	1.09 ms (99.21%)
UR5	6	7.39 ms (88.14%)	0.79 ms (77.81%)	0.36 ms (86.97%)	323.84 ms (68.53%)	159.11 ms (89.75%)	0.94 ms (100%)	0.91 ms (99.38%)
Fanuc M-10iA	6	6.92 ms (45.78%)	1.34 ms (55.83%)	0.35 ms (49.66%)	416.81 ms (42.67%)	203.64 ms (76.91%)	1.26 ms (100%)	1.07 ms (98.96%)

Fig. 8. Robotic arms used for serial inverse kinematics.

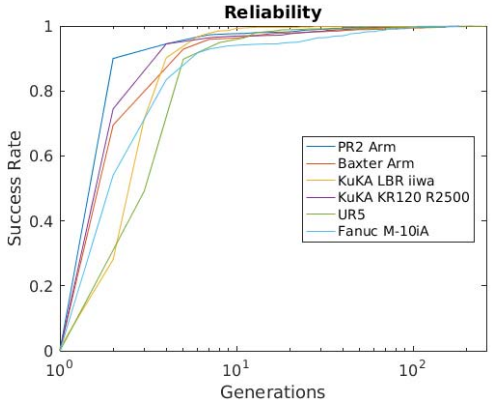

Fig. 9. Success rate with respect to the number of generations over 10.000 reachable targets for the robot models in Fig. 8.

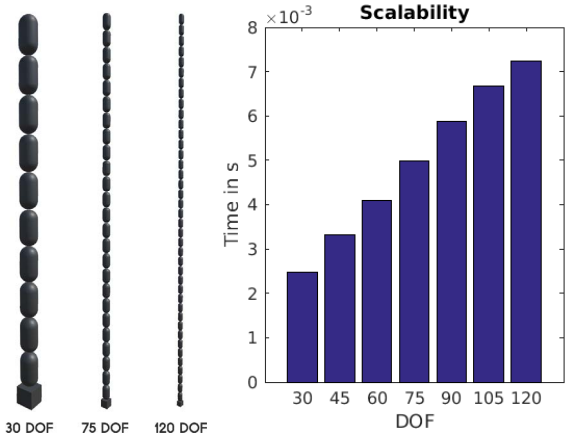

Fig. 10. Performance on hyper-redundant manipulators.

complexity regarding solution manifolds and joint limits, which often cause traditional algorithms get stuck in suboptimal extrema.

The teal spheres show the targets to be optimized, which were generated from random valid joint configurations using the FK equations (1). The algorithm was then used to find a solution which minimizes the translational and rotational error to an accuracy of 10^{-4} m and 10^{-3} rad. Fig. 9 visualizes the averaged reached success rate over the number of evolved generations. The graph shows that all 10.000 goal configurations for all tested robots could be successfully evolved after approximately 10^2 generations. However, most solutions can typically be expected to be found in much fewer generations.

Next, Table I gives an overview of the actual computation time which is required to find a solution, and also compares the algorithm to existing methods for inverse kinematics with respect to time and success. For the popular Jacobian and L-BFGS-B methods, the optimization was terminated if the algorithm began violating joint limits, in which case no valid solution would be found. For the evolutionary SGA and DE methods, a time limit of 1 s was specified by which a solution had to be found. While the gradient-based approaches converge much faster than the evolutionary methods, the overall reliability of the former to find a solution significantly varies for the different robot geometries, but which is more stable for the latter although being too slow. Finally, Bio-IK has similar computation time like gradient-based methods requiring approximately 1 ms, but outperforms them with success rates of over 99% using a specified time limit of 10 ms.

TABLE II
PERFORMANCE OF THE ALGORITHM IN SOLVING FULL-BODY POSTURES WITH MULTIPLE CARTESIAN OBJECTIVES ON THE NASA VALKYRIE ROBOT. THE RESULTS SHOW THE AVERAGE COMPUTATION TIMES AND SUCCESS RATES FOR THE CHOSEN PARAMETER SELECTIONS OVER 10.000 QUERIES

NASA Valkyrie Setup	DOF	Objectives	Parameters (Individuals / Elites)	Time and Success for ME Loss (10ms timeout)	Time and Success for RMSE Loss (10ms timeout)
Torso to Right Wrist (Position & Orientation)	10	2	90 / 2	1.79 ms (90.23%)	0.87 ms (99.96%)
Right Leg (Position & Orientation)	6	2	75 / 2	1.48 ms (93.36%)	0.63 ms (99.98%)
Torso to Left and Right Wrist (Position & Orientation)	17	4	100 / 3	5.97 ms (24.72%)	2.07 ms (98.31%)
Torso to Right Wrist (Position & Orientation) with Right Elbow (Position) and to Head (Look At)	13	4	110 / 3	5.31 ms (18.11%)	2.23 ms (98.82%)
Torso to Left and Right Wrist (Position & Orientation) and to Head (Look At)	20	5	130 / 3	6.53 ms (7.94%)	2.67 ms (97.98%)
Torso to Right Wrist (Orientation) and to Thumb, Index, Middle and Ring Fingers (Position)	23	5	150 / 3	6.22 ms (11.61%)	3.24 ms (97.69%)
Torso to Left and Right Wrist (Position & Orientation) and to both Feet (Position & Orientation)	32	8	160 / 4	8.43 ms (1.27%)	5.64 ms (95.29%)
Torso to Left and Right Wrist (Position & Orientation) and to both Feet (Position & Orientation) and to Head (Look At)	35	9	180 / 4	9.13 ms (0.48%)	6.22 ms (94.56%)

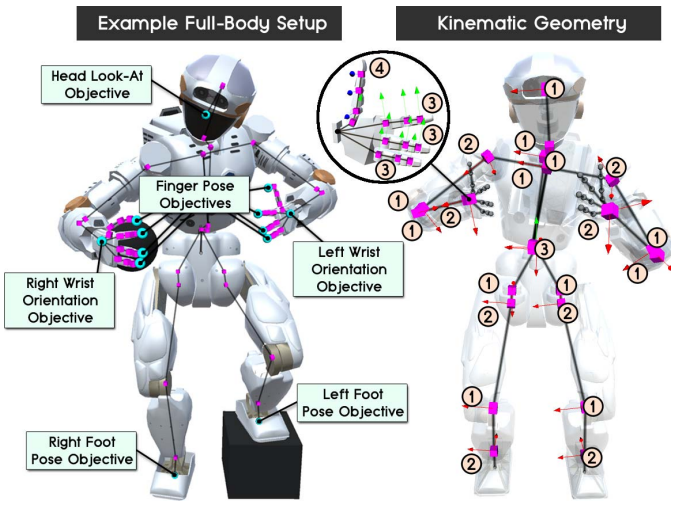

Fig. 11. Geometry of the 35/61 DOF NASA Valkyrie humanoid robot (without/with the 13 DOF hands).

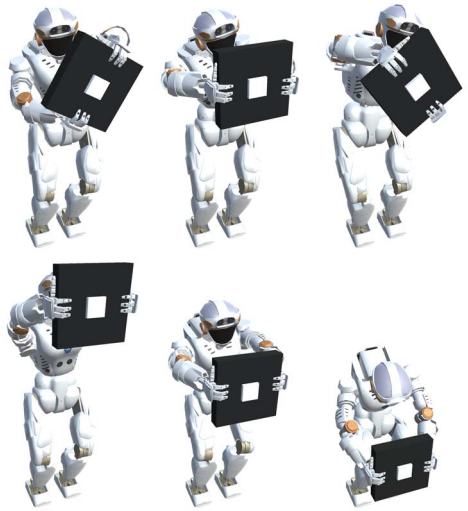

Fig. 12. Full-body motion on the NASA Valkyrie robot while lifting and rotating an object in real-time.

Fig. 10 shows the average required time for 100% success rates on hyper-redundant manipulators with up to 120 DOF, for which a slightly sublinear relation between time and DOF can be observed. Solving such structures is often difficult for Jacobian methods due to the typical singularity issues.

C. Humanoid Robots and Full-Body Motion

A more challenging problem for inverse kinematics is given by fully articulated humanoid robots with multiple kinematic chains, and who have joints which dependently control multiple end effector poses. Fig. 11 shows the NASA Valkyrie robot of which both arms are affected by the pelvis and torso configuration, and which further also affect the finger poses of the anthropomorphic hands. For such geometries, it is required to find a single solution whose resulting posture globally satisfies the multiple goals, and wherefore related methods discussed in Section II are often not applicable anymore. Table II lists the required computation time and resulting success rates for different inverse kinematics setups and objective combinations. The algorithm maintains a fast and robust convergence even for solving multiple goals simultaneously and high DOF. Especially when adding further objectives, it was observed that using the RMSE loss in (14) is able to achieve significant improvements for the optimization—making the algorithm considerably faster and applicable for robustly solving complex full-body postures.

Fig. 12 shows some keyframes of a full-body motion for lifting and rotating an object which could be evolved in real-time. The wrist and finger poses were defined on the surface of the object and used to generate the motion of the robot. Similarly, an objective for the viewing direction of the head was defined at the object center. Further position and orientation objectives were defined for both feet so stick to the ground. The required Cartesian configuration of the pelvis as the root of the hierarchical geometry was then also evolved by the algorithm using a translational prismatic joint.

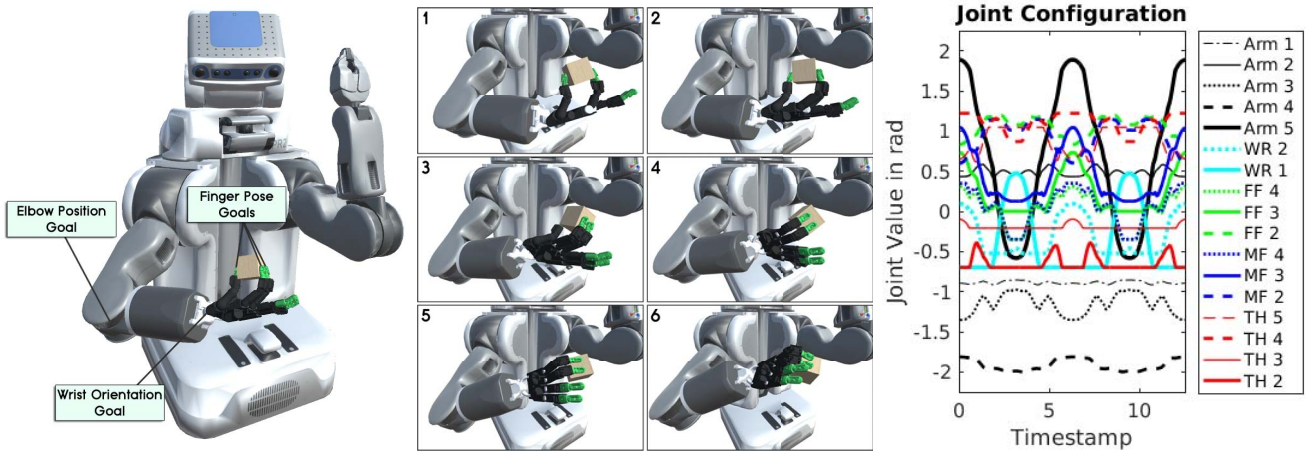

Fig. 13. Dexterous object manipulation scenario using the PR2 with the Shadow Dexterous Hand attached. The curves visualize the generated motion in joint space for the arm, wrist (WR), first finger (FF), middle finger (MF), and thumb (TH).

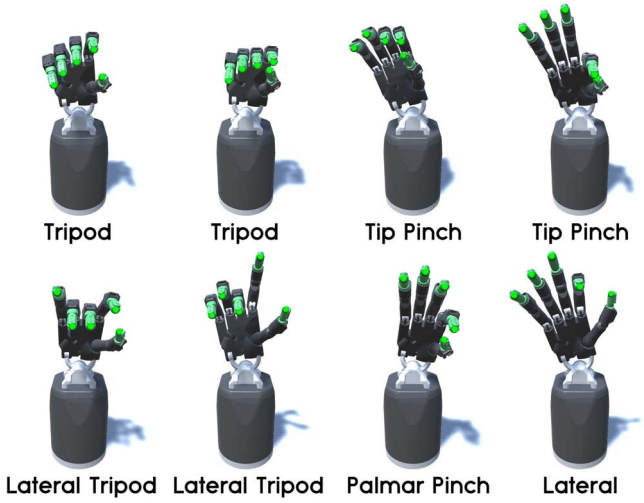

Fig. 14. Solving inverse kinematics for different grasp types of real-human hand data on the Shadow Dexterous Hand.

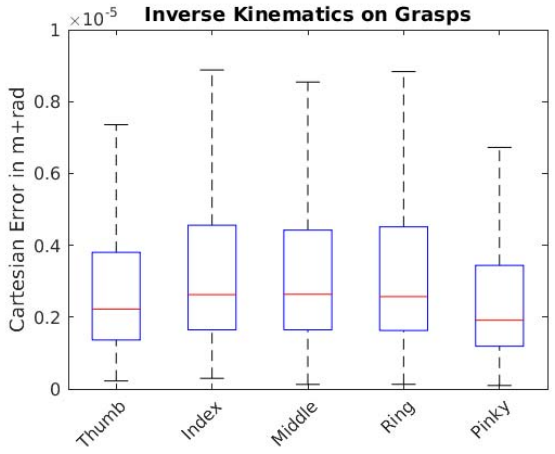

Fig. 15. Statistical accuracy for reconstructing 444 real-human grasp configurations using the Shadow Dexterous Hand.

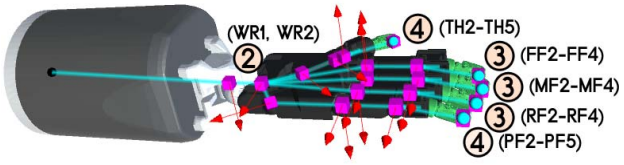

Fig. 16. Geometry of the 19 DOF Shadow Dexterous Hand.

D. Anthropomorphic Grasps and Dexterous Manipulation

The algorithm was further investigated on anthropomorphic robotic hands for solving grasps and dexterous manipulation scenarios. Such tasks are often difficult since all fingers are again directly controlled by the preceding joints of the wrist and the arm of a robot. In addition, manipulation with real human hands typically involves rolling the fingers slightly on top of the surface instead of keeping a particular pose for the individual finger tips. From a computational perspective, this requires handling particular objectives as soft-constraints, such as having an appropriately lower weight for the orientation than for the position objective. For the experiments, the Shadow Dexterous Hand shown in Fig. 16 was used, which consists of a 2 DOF wrist, 4 DOF thumb and pinky fingers, and 3 DOF index, middle, and ring fingers.

Dexterous manipulation requires solving grasps which ideally incorporate the wrist configuration for all fingers—solving the hand as a whole kinematic system rather than each finger independently. Fig. 14 demonstrates the proposed algorithm applied to different exemplary grasping types for the hand. Those were recorded from data of 444 real-human grasps [46], and the algorithm was then used to find a solution which resembles the original finger poses of the hand, despite of varying link lengths and joint limits between the humanoid and robotic hand. All different tested grasp types could be successfully evolved within a few milliseconds.

When using a fixed optimization time of 10 ms, Fig. 15 shows the achieved accuracy as the sum of errors in position and orientation of the single fingers, along with the statistical deviation between the minimum and maximum values over reconstructing all 444 prepared grasp configurations. It demonstrates that the memetic evolution can be used to obtain solutions with a pose accuracy between 10^{-5} m and 10^{-4} rad. In addition, it can be observed that it was possible to obtain a slightly better accuracy for the thumb and pinky fingers which

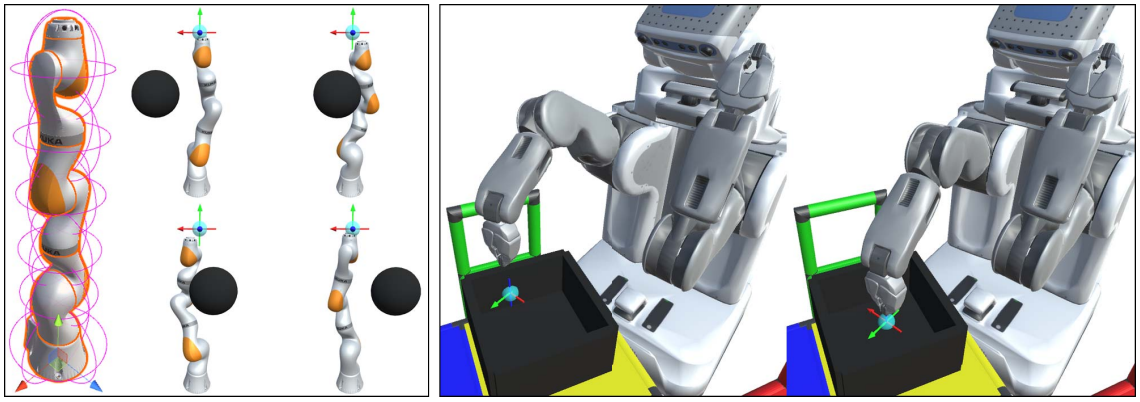

Fig. 17. Real-time collision avoidance using distance objectives on the KuKA LBR iiwa (left) and PR2 robot (right).

have one DOF more than the other fingers—which can be reasoned by a higher kinematic flexibility and thus a larger solution manifold in the fitness landscape.

Next, the Shadow Dexterous Hand was attached to the right arm of the PR2 robot shown in Fig. 13, having 17 DOF in total. In the experiment, a wooden cube was rotated around all three axes, and the motion of the robot from the arm to the thumb, index, and middle fingers should be evolved. In addition, a further goal was defined for the head camera to track the center of the object. A grasping pose on the surface of the cube was defined for each finger while using a significantly lower weight for orientation with 0.1 than for position with 10.0. Two weighted soft-constraints for the wrist orientation with 0.5 and elbow position with 0.25 were specified to encourage the optimization to evolve into a reasonable grasping posture. Note that both intermediate goals for the elbow and wrist do not necessarily need to be reached exactly, which is controlled by the weights to offer some amount of flexibility for the arm configuration while solving the grasp. Thus, not only the wrist motion of the hand, but also the whole arm configuration is incorporated. The rightmost figure visualizes the configuration of the single joints covering two full manipulation cycles of the object. All joints of the robot arm can be observed to be adjusted over time, while also accounting for joints running into their lower or upper limits. In such cases, the evolution performs slightly higher updates for the joints that can still be moved, and enables to find a solution despite of some joints reaching their joint space limits.

E. Collision Avoidance

Collision avoidance with obstacles as well as with the robot itself is an issue which is often neglected for generic inverse kinematics solvers. However, it has particular relevance in order to provide valid solutions in Cartesian space. Those can then be used for collision-free trajectory generation or for procedural animation in modern video games. The proposed algorithm handles collision avoidance by using objectives which ensure to maintain a minimum distance between specified points—resulting in spherical colliders. Those can be used to approximate collision geometries, as shown in Fig. 17 on the KuKA LBR iiwa robot. As a result, the robot arm smoothly avoids the black sphere obstacle while reaching for the teal sphere target. A more complex experiment was conducted using the PR2 robot moving the gripper to a pose goal within an open box. Similarly, the arm and box geometries were approximated through multiple distance objectives. When placing the target within a solid part of the box, the algorithm still tries to reach the target as close as possible without resulting in a collision. Finally, when placing the target within free space inside the box, a solution can be found with safety distances to the nearest collision points.

F. Motion Reconstruction

Further challenging applications are given by controlling a robot through teleoperation in human-robot interaction, or by remotely transferring motion on a game character in virtual reality. Such tasks require a fast and responsive real-time computation, and also the ability to solve highly articulated models. In addition, a robust handling of different link lengths and joint limits between the kinematic structure and the real human operator must be ensured to achieve plausible postures. An example of reconstructing human motion on the NASA Valkyrie robot model in virtual reality is shown in Fig. 18. Two position and orientation objectives were specified for the left and right wrist of the robot, along with a further orientation objective for the head, and were updated by the tracking information obtained from the HTC Vive controllers and headset. Hence, only a few goals were used trying to reconstruct a full-body motion on a complex humanoid robot while having the feet fixed on the ground. As can be observed, the algorithm successfully evolves similar looking postures on the robot as those of the operator. Smooth motion transitions while standing and turning or stretching the upper body, waving with hands or kneeling could be obtained in real-time.

G. Animation Postprocessing

Inverse kinematics is an essential method for generating or post-processing animations for virtual characters. A common application in video games is to adjust existing walking or running motion in real-time such that the feet do not disappear within the ground, but instead are properly placed on the surface. Fig. 19 visualizes the algorithm post-processing a running animation for the Kyle humanoid while traversing an uneven terrain. Two pose goals were defined at the bottom center of the feet, and initially controlled by the default position and orientation of the original animation. Then, downward projections from the knees to the foot goals were

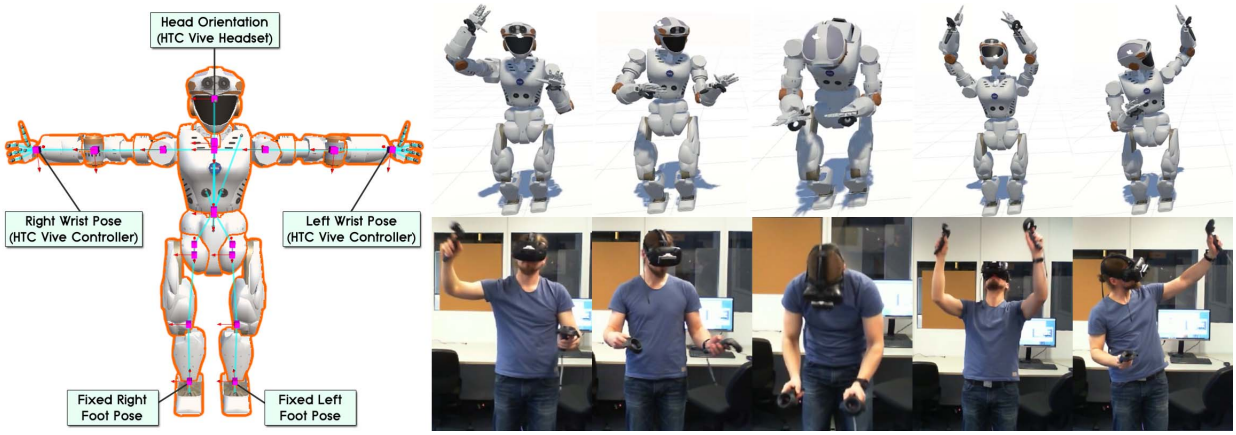

Fig. 18. Motion reconstruction on the NASA Valkyrie robot in virtual reality using the HTC Vive controllers and headset.

Fig. 19. Runtime post-processing of existing animations for terrain-adaptive foot placement using the Kyle humanoid.

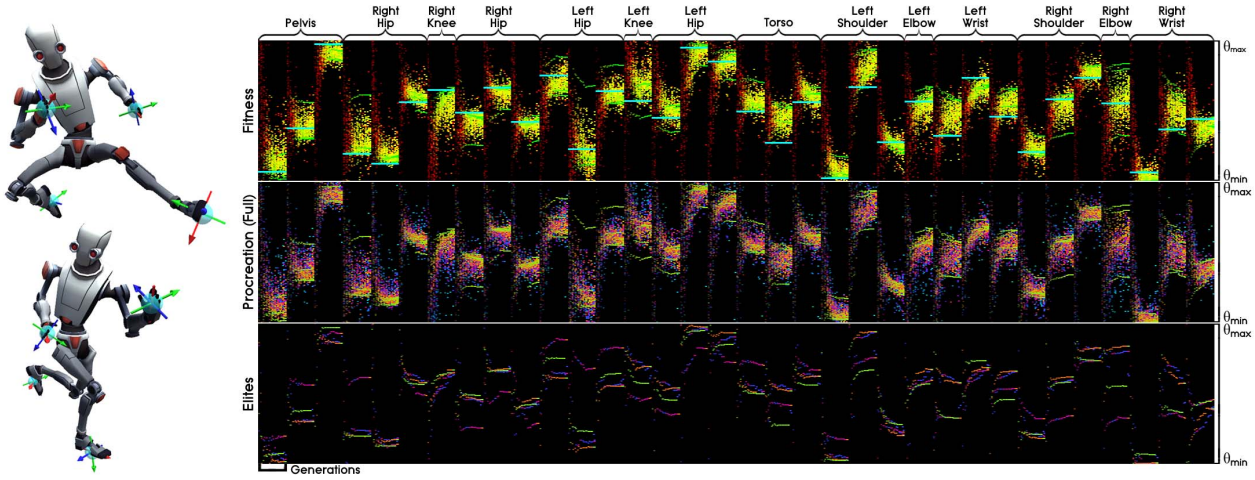

Fig. 20. Evolutionary algorithm for solving full-body inverse kinematics queries on the 34 DOF Kyle humanoid. Each column represents one joint dimension with its full optimization history until convergence from left to right. The color in the top row shows the normalized fitness values, interpolated between red (bad) and green (good). The middle and bottom row highlight the evolved optimization paths, where each color indicates one elite individual and its path through the search space.

used to correct them in case of intersection with the solid ground—such that the feet normal becomes orthogonally aligned to the ground surface. Note that all joints along the 7 DOF legs were used to correct the foot poses instead of only updating the orientation of the feet. Hence, the algorithm can be used to generate more realistic animations in real-time. In addition, animations of multiple characters can be resembled simultaneously—producing slight variations for each of them and thus a more natural manifold of concurrent animations.

H. Evolutionary Landscapes

The resulting underlying optimization of the algorithm is visualized in Fig. 20. For the experiments, different kinematic postures were create using the whole body of the Kyle humanoid by defining pose goals for the hands and feet. Each particle represents one joint value from the whole configuration over all dimensions of an individual. In the top row, the full population is shown with the color indicating the fitness of an individual. Although the algorithm converges,

it can be observed that a manifold of different solutions is maintained. In the middle row, the procreation of genes is shown to highlight the individuals which could produce offspring with better fitness values. The bottom row lastly shows the evolved joint configurations for the elite individuals, which demonstrates that the algorithm can track and find multiple solutions simultaneously. This property can be used to choose from a set of possible configurations in order to create collision-free trajectories between body postures, which is a common problem in robotics. In character animation, it allows the creator to choose from a set of similar postures for generating animation keyframes.

V. Limitations and Challenges

The algorithm was generally found being flexible to apply for different applications. In particular, further objectives could be designed and added quite easily as required. However, some difficulties were observed in finding proper weightings between those, which is not generally clear how to chose. In particular, setting higher weights for collision avoidance or intermediate position or orientation goals along the kinematic chains can result in not accurately solving end effector targets anymore. Although this is correct in terms of optimization, it might not always yield the desired behavior. Further, due to the probabilistic optimization it is possible that sudden changes between solutions might occur. Although this was only observed in very rare cases or around singularities, and could be avoided using a displacement objective to penalize large changes, we think this issue should be mentioned to be considered for real-world robotics applications. Finally, it was observed that the algorithm should be given 0.5–1 ms optimization time for animation post-processing in games whereas $\sim$16 ms (60 Hz) is the usual time limit to maintain real-time frame rates. This suggests that the algorithm is predominantly suited for fine control of some main characters rather than for controlling multiple less important characters at the same time, and for which computationally faster but less powerful methods are typically sufficient. Nevertheless, there should be very few cases in which both solving fine control of motion on multiple characters concurrently is actually required.

VI. Conclusion

This paper proposed a novel memetic evolutionary approach for solving inverse kinematics with multiple concurrent goals on fully constrained generic geometries. Since GA, PSO, and the L-BFGS-B algorithm can all operate on the same objective function, the characteristic strengths of biologically inspired and gradient-based optimization could be successfully combined into one method—performing both global and local search while offering maximum flexibility for the design of custom cost functions. The experiments demonstrated that the algorithm serves the different performance aspects in computational efficiency, robustness and scalability which are required in robotics as well as animation. In particular, similar speed of convergence as for popular gradient-based methods can be provided, but while achieving significantly higher success rates on all tested robot and character models. The algorithm was further demonstrated to be applicable for different challenging tasks, including full-body motion and dexterous manipulation with anthropomorphic hands, motion reconstruction in virtual reality, as well as for realizing terrain-adaptive foot placement of virtual characters through animation post-processing. Additionally, while collision avoidance is typically neglected for related generic inverse kinematics solvers, it was possible to successfully formulate an objective to efficiently integrate this functionality. Therefore, we believe evolutionary optimization to be a suitable methodology for solving complex inverse kinematics setups. Implementations of the algorithm are readily available in Unity3D (C#) and ROS (C++), and can be used for research in robotics, animation, virtual reality, and game development.

A. Future Work

So far, the algorithm was predominantly applied to solving pure kinematic postures, but not yet considering the dynamics of the system. Thus, further research can address the design of center-of-gravity objectives to integrate balancing while optimizing full-body postures. Although this paper mainly focused on hierarchical structures, future directions can also include defining objectives for parallel mechanisms that require solving separate kinematic chains controlling a single end effector. Another goal is to create a method or several heuristics to automatically select appropriate weights for combining such objectives. The algorithm will also be further investigated for dexterous manipulation tasks with anthropomorphic hands, collision-free trajectory generation, close character interaction, as well as for procedural animation.

References

[1] B. Siciliano and O. Kathib, *Handbook of Robotics*. Heidelberg, Germany: Springer, 2008.
[2] R. Parent, *Computer Animation: Algorithms and Techniques*, 3rd ed. Waltham, MA, USA: Morgan Kaufmann, 2012.
[3] R. Diankov, "Automated construction of robotic manipulation programs," Ph.D. dissertation, Robot. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, 2010.
[4] R. Diankov, K. Sato, H. Yaguchi, K. Okada, and M. Inaba, *Manipulation Planning for the JSK Kitchen Assistant Robot Using OpenRAVE*, Univ. Tokyo, Tokyo, Japan, 2011.
[5] L.-C. T. Wang and C. C. Chen, "A combined optimization method for solving the inverse kinematics problems of mechanical manipulators," *IEEE Trans. Robot. Autom.*, vol. 7, no. 4, pp. 489–499, Aug. 1991.
[6] A. A. Canutescu and R. L. Dunbrack, "Cyclic coordinate descent: A robotics algorithm for protein loop closure," *Protein Sci.*, vol. 12, no. 5, pp. 963–972, 2003.
[7] A. Aristidou and J. Lasenby, "FABRIK: A fast, iterative solver for the inverse kinematics problem," *Graph. Models*, vol. 73, no. 5, pp. 243–260, 2011.
[8] A. Aristidou, Y. Chrysantho, and J. Lasenby, "Extending FABRIK with model constraints," *Comput. Animat. Virtual Worlds*, vol. 27, no. 1, pp. 35–57, 2016.
[9] *Root Motion*. (Jun. 24, 2018). [Online]. Available: http://root-motion.com
[10] A. Lansley, P. Vamplew, P. Smith, and C. Foale, "Caliko: An inverse kinematics software library implementation of the FABRIK algorithm," *J. Open Res. Softw.*, vol. 4, no. 1, p. e36, 2016.
[11] *Unity3D*. (Jun. 24, 2018). [Online]. Available: http://www.unity3d.com
[12] *Unreal Engine*. (Jun. 24, 2018). [Online]. Available: https://www.unrealengine.com
[13] *Maya*. (Jun. 24, 2018). [Online]. Available: https://knowledge.autodesk.com/support/maya

[14] S. R. Buss, *Introduction to Inverse Kinematics With Jacobian Transpose, Pseudoinverse and Damped Least Squares Methods*, Univ. California at San Diego, San Diego, CA, USA, 2004.
[15] G. Tevatia and S. Schaal, "Inverse kinematics for humanoid robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, San Francisco, CA, USA, 2000, pp. 294–299.
[16] B. Kenwright, "Real-time character inverse kinematics using the Gauss–Seidel iterative approximation method," in *Proc. Int. Conf. Creative Content Technol.*, 2012, pp. 63–68.
[17] *Orocos KDL*. (Jun. 24, 2018). [Online]. Available: http://www.orocos.org/kdl
[18] M. Quigley *et al.*, "ROS: An open-source robot operating system," in *Proc. IEEE ICRA Workshop Open Source Softw.*, 2009, p. 5.
[19] P. Beeson and B. Ames, "TRAC-IK: An open-source library for improved solving of generic inverse kinematics," in *Proc. IEEE RAS Int. Conf. Humanoid Robot.*, Seoul, South Korea, Nov. 2015, pp. 928–935.
[20] P. Beeson. (Jun. 24, 2018). *TRAC-IK GitHub*. [Online]. Available: https://bitbucket.org/traclabs/trac_ik.git
[21] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. Chichester, U.K.: Wiley-Interscience, 1987.
[22] J. Zhao and N. I. Badler, "Inverse kinematics positioning using nonlinear programming for highly articulated figures," *ACM Trans. Graph.*, vol. 13, no. 4, pp. 313–336, 1994.
[23] A. S. Lewis and M. L. Overton, "Nonsmooth optimization via quasi-Newton methods," *Math. Program.*, vol. 141, nos. 1–2, pp. 135–163, 2013.
[24] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: Univ. Michigan Press, 1975.
[25] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4. Perth, WA, Australia, 1995, pp. 1942–1948.
[26] S. Tabandeh, C. M. Clark, and W. W. Melek, "An adaptive niching genetic algorithm approach for generating multiple solutions of serial manipulator inverse kinematics with applications to modular robots," *Robotica*, vol. 28, no. 4, pp. 493–507, 2010.
[27] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, Dec. 1997.
[28] C. E. G. Uzcátegui, "A memetic approach to the inverse kinematics problem for robotic applications," Ph.D. dissertation, Dep. de Ingeniería de Sistemas yAutomática, Carlos III Univ. Madrid, Getafe, Spain, 2014.
[29] O. A. Aguilar and J. C. Huegel, *Inverse Kinematics Solution for Robotic Manipulators Using a CUDA-Based Parallel Genetic Algorithm* (LNCS 7094). Heidelberg, Germany: Springer, 2011, pp. 490–503.
[30] M. Stollenga *et al.*, "Task-relevant roadmaps: A framework for humanoid motion planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Tokyo, Japan, 2013, pp. 5772–5778.
[31] B. Durmus, H. Temurtas, and A. Gün, "An inverse kinematics solution using particle swarm optimization," in *Proc. 6th Int. Adv. Technol. Symp.*, 2011, pp. 193–197.
[32] R. Falconi, R. Grandi, and C. Melchiorri, "Inverse kinematics of serial manipulators in cluttered environments using a new paradigm of particle swarm optimization," *Int. Federation Autom. Control*, vol. 19, no. 3, pp. 8475–8480, 2014.
[33] T. Collins and W.-M. Shen, "PASO: An integrated, scalable PSO-based optimization framework for hyper-redundant manipulator path planning and inverse kinematics," Inf. Sci. Inst., Univ. Southern California, Los Angeles, CA, USA, Rep. ISI-TR-697, 2016.
[34] Y. Feng, W. Yao-Nan, and Y. Yi-Min, "Inverse kinematics solution for robot manipulator based on neural network under joint subspace," *Int. J. Comput. Commun. Control*, vol. 7, no. 3, pp. 459–472, 2012.
[35] P. Jha and B. B. Biswal, "A neural network approach for inverse kinematic of a scara manipulator," *Int. J. Robot. Autom.*, vol. 3, no. 1, pp. 52–61, 2014.
[36] A. R. J. Almusawi, L. C. Dülger, and S. Kapucu, "A new artificial neural network approach in solving inverse kinematics of robotic arm (Denso VP6242)," *Comput. Intell. Neurosci.*, vol. 2016, Jul. 2016, Art. no. 5720163. [Online]. Available: https://www.hindawi.com/journals/cin/2016/5720163/cta/
[37] A. Morell, M. Tarokh, and L. Acosta, "Inverse kinematics solutions for serial robots using support vector regression," in *Proc. IEEE Int. Conf. Robot. Autom.*, Karlsruhe, Germany, 2013, pp. 4203–4208.
[38] S. Starke, N. Hendrich, S. Magg, and J. Zhang, "An efficient hybridization of genetic algorithms and particle swarm optimization for inverse kinematics," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, Qingdao, China, 2016, pp. 1782–1789.
[39] S. Starke, N. Hendrich, and J. Zhang, "A memetic evolutionary algorithm for real-time articulated kinematic motion," in *Proc. IEEE Congr. Evol. Comput.*, 2017, pp. 2473–2479.
[40] S. Starke, N. Hendrich, D. Krupke, and J. Zhang, "Evolutionary multi-objective inverse kinematics on highly articulated and humanoid robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Vancouver, BC, Canada, 2017, pp. 6959–6966.
[41] J. Nocedal, "Updating quasi-Newton matrices with limited storage," *Math. Comput.*, vol. 35, no. 151, pp. 773–782, 1980.
[42] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM J. Sci. Stat. Comput.*, vol. 16, no. 5, pp. 1190–1208, 1995.
[43] S. Starke, N. Hendrich, and J. Zhang, *A Forward Kinematics Data Structure for Efficient Evolutionary Inverse Kinematics* (Mechanisms and Machine Science). Cham, Switzerland: Springer, 2017, pp. 560–568.
[44] *Bio IK for Unity3D*. (Jun. 24, 2018). [Online]. Available: https://www.assetstore.unity3d.com/en/#!/content/67819
[45] *Bio IK for ROS*. (Jun. 24, 2018). [Online]. Available: https://github.com/TAMS-Group/bio_ik
[46] A. Bernardino, M. Henriques, N. Hendrich, and J. Zhang, "Precision grasp synergies for dexterous robotic hands," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, Shenzhen, China, 2013, pp. 62–67.

Sebastian Starke (S'17) received the B.Sc. and M.Sc. (with Distinction) degrees in informatics from the University of Hamburg, Hamburg, Germany, in 2014 and 2016, respectively. He is currently pursuing the Ph.D. degree in character animation and artificial intelligence with the School of Informatics, University of Edinburgh, Edinburgh, U.K.

Before starting at the University of Edinburgh, in 2017, he was a Research Associate in the Transregio-SFB TRR 169 project on "Crossmodal Learning" in TAMS Group, University of Hamburg, where his research focused on robotics and evolutionary optimization. His current research interests include developing novel methods for character animation and control using deep learning.

Norman Hendrich (M'12) received the M.Sc. degree in physics and the Ph.D. degree in computer science from the University of Hamburg, Hamburg, Germany, in 1991 and 1996, respectively.

He is a Senior Lecturer with the Department of Informatics, University of Hamburg, where he currently acts as a Project Manager for the Transregio-SFB TRR 169 "Crossmodal Learning" with the TAMS Group. He has participated as a principal investigator in several collaborative European research projects. His current research interests include machine learning, service robotics, and dexterous manipulation with multifingered hands.

Jianwei Zhang (M'95) received the B.Eng. (with Distinction) and M.Eng. degrees from the Department of Computer Science, Tsinghua University, Beijing, China, in 1986 and 1989, respectively, the Ph.D. degree from the Institute of Real-Time Computer Systems and Robotics, Department of Computer Science, University of Karlsruhe, Karlsruhe, Germany, in 1994, and the Habilitation from the Faculty of Technology, University of Bielefeld, Bielefeld, Germany, in 2000.

He is a Professor and the Head of the TAMS Group, Department of Informatics, University of Hamburg, Hamburg, Germany. He has published about 300 journal and conference papers (winning four best paper awards), technical reports, four book chapters, and five research monographs. He has been coordinating numerous collaborative research projects of EU and German Research Council, including the Transregio-SFB TRR 169 "Crossmodal Learning." His current research interests include cognitive robotics, sensor fusion, dexterous manipulation, and multimodal robot learning.

Dr. Zhang is a life-long Academician of the Academy of Sciences, Hamburg. He is the General Chair of the IEEE MFI 2012, the IEEE/RSJ IROS 2015, and the IEEE Robotics and Automation Society AdCom from 2013 to 2015.