A Versatile Inverse Kinematics Formulation for Retargeting Motions onto Robots with Kinematic Loops

Christian Schumacher*,1, Espen Knoop*,1, Moritz Bächer*,1

***Abstract*—Robots with kinematic loops are known to have superior mechanical performance. However, due to these loops, their modeling and control is challenging, and prevents a more widespread use. In this paper, we describe a versatile Inverse Kinematics (IK) formulation for the retargeting of expressive motions onto mechanical systems with loops. We support the precise control of the position and orientation of several end-effectors, and the Center of Mass (CoM) of slowly walking robots. Our formulation safeguards against a disassembly when IK targets are moved outside the workspace of the robot, and we introduce a regularizer that smoothly circumvents kinematic singularities where velocities go to infinity. With several validation examples and three physical robots, we demonstrate the versatility and efficacy of our IK on overactuated systems with loops, and for the retargeting of an expressive motion onto a bipedal robot.**

***Index Terms*—Kinematics; Parallel Robots; Methods and Tools for Robot System Design**

I. Introduction

ROBOTS consisting of kinematic loops have several advantages over designs that consist of a fully actuated kinematic tree: they have superior stiffness, allow the placement of actuators where there is space, and often exhibit an advantageous payload-to-weight ratio [1]. However, while there are clear mechanical advantages, kinematic trees remain a popular choice in legged robots and related systems due to the complexity of the design, control, and simulation of mechanisms containing loops.

For robots with loops, or parallel robots, designs are commonly made of submechanisms for which, for example, a closed-form analytical model exists, and inverse kinematics can be formulated by interfacing between submodules. However, this modeling forgoes the true potential of robots with loops, because for every type of submechanism a custom module has to be written from scratch [1], unnecessarily restricting the mechanical design of custom robots.

In this paper, we propose a versatile inverse kinematics that enables the kinematic control of general robotic systems with (1) position and orientation objectives to precisely control the motion of end-effectors or bodies, and (2) a CoM objective to keep the CoM within the support spanned by the contact of the robot with the ground. See Fig. 1 for an illustrative example of our pipeline.

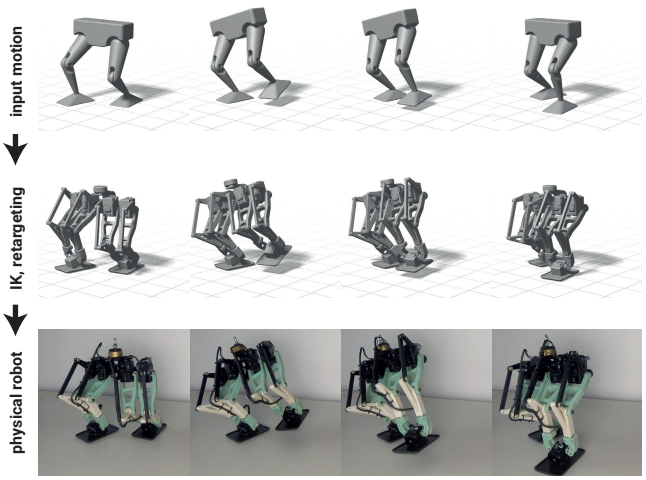

Fig. 1. Starting from an input motion, in this case an animated walking cycle (top), our pipeline solves the IK problem of retargeting this motion onto robots with arbitrary kinematic structures (middle). By including a CoM objective in the IK, we obtain a stable gait that can be executed on a physical robot (bottom). Note that a naive motion retargeting, without the CoM objective, would cause the robot to fall over.

Our approach combines a *constraint-based* formulation for mechanical joints with an *objective-based* formulation for IK targets. This means that the returned IK solution is as close as possible to the specified targets, in a least-squares sense, while we guarantee that the mechanical constraints of the robot are satisfied exactly. Moreover, we safeguard against kinematic singularities at which velocities go to infinity.

We compare our IK formulation to an implicit approach where actuated degrees of freedom are solved for directly, with a first-order optimality constraint on forward kinematics. In contrast to our formulation, the mechanical system disassembles if not all IK targets can be met. Moreover, in our IK formulation, we only need derivatives, and hence building blocks, from a multibody kinematics implementation, making it straightforward to add inverse kinematics support to existing forward kinematics codes.

With validation examples, we demonstrate that our IK works on mechanical systems containing all common types of active and passive joints. With three physical robot examples, we demonstrate how (1) a rich dancing motion can be transferred

Manuscript received: Oct, 15, 2020; Revised Dec, 22, 2020; Accepted Jan, 05, 2021.

This paper was recommended for publication by Editor Lucia Pallottino upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by Disney Research.

*All authors contributed equally.

[1]All authors are with Disney Research, Stampfenbachstrasse 48, 8006 Zurich, Switzerland. firstname.lastname@disney.com

Digital Object Identifier (DOI): see top of this page.

over to a 6-DoF pair of legs with a single end-effector objective, (2) an expressive walking cycle can be retargeted onto a bipedal robot with complex loops, with the help of our CoM objective, and (3) how we can safeguard against velocity singularities when retargeting motions from a digital to a physical arm with fewer degrees of freedom. Our IK can hence be understood as a first-order approximation of a dynamics model, without having to deal with force and torque transfer, or frictional contact.

II. Related Work

There is a large body of related work on inverse kinematics in robotics, and related fields such as, e.g., computer graphics [2]. While most IK formulations target serial or tree-like robots, inverse kinematics for serial-parallel hybrid robots, or mechanical systems that contain kinematic loops, are less common despite their desirable properties such as increased accuracy, stiffness, or payload capacity [1]. We target the latter category of robots with our IK.

a) Serial Robots: Tree-like robots are usually modeled as reduced or generalized coordinate formulations, and closed-form analytical [3], learning-based techniques [4], and Jacobian-based numerical approaches [5] are common. Targeting robotics control, Pechev [6] introduced an IK that does not require a matrix inversion (increased speed), more robustly handles kinematic singularities, and supports multiple end effectors. Jacobian-based formulations were also extended to handle joint limits (see, e.g., [7]), with Newton-based approaches handling them better in general [8]. Like [9], we enable the control of the CoM position of the robot, and like [6], our IK handles multiple targets gracefully. However, in contrast, our IK supports arbitrary mechanical systems, containing kinematic loops.

b) Parallel Robots: A large body of work has focused on parallel robots, or Parallel Kinematic Machines (PKMs). An extensive discussion is beyond the scope of this paper, and we refer to [10] for a comprehensive overview. For most studied PKMs, the IK problem is relatively simple [10] and is solved by dividing the mechanism into a number of simple kinematic chains which are then tackled analytically. A central challenges in the study of PKMs is the avoidance of workspace singularities [11]. More recent work on PKMs includes [12], where the kinematic analysis of an overactuated parallel robot is considered, and [13], where a formulation is provided which is applicable also to robots with redundant degrees of freedom. However, these approaches are not readily applicable to arbitrary robot topologies as considered here.

c) General Serial-Parallel Robots: To represent general robots with loops, maximal coordinates are commonly used [14]. As a recent survey on serial-parallel hybrid robots points out [1], the "kinematic analysis of generic series-parallel hybrid robots is an open problem", let alone inverse kinematics. For specific serial-parallel topologies, known solutions for building blocks (submechanisms) could be used (e.g., IK for a 6R manipulator [15], [16], [17]). Hybrid approaches where the loops in mechanical systems are cut, and consistency constraints are introduced, have also received attention (see, e.g., [18]). However, these mixed representations where reduced and maximal coordinates are combined, are very complex. In contrast, our kinematics and inverse kinematics is applicable to general mechanical systems that are modeled as multibody systems, represented with generalized coordinates.

Analytic IK solutions are also common (see for example [19, 20, 21]), however they are laborious, and do not generalize to arbitrary mechanisms.

There are relatively few examples in the literature of serial-parallel robots where the parallel elements cannot be treated as submodules in the serial chain. One such robot is the series-parallel biped presented by Gim et al [22]. We believe that one reason why such robots are not more widespread is their challenging inverse kinematics and the lack of suitable off-the-shelf IK solvers.

III. Forward Kinematics

Before delving into the specifics of our inverse kinematics, we summarize how kinematics simulations are performed. To motivate and illustrate our representation, we reference the four-bar linkage example in Fig. 2.

In what follows, we will first introduce the rigid body representation, and discuss constraints the actuators and joints impose. We will then discuss how the unknown states can be solved for after the actuators are stepped.

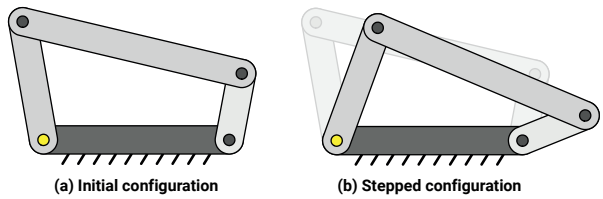

Fig. 2. Four-bar linkage in its initial (a) and a stepped configuration (b). The linkage is driven by a single actuator (yellow circle) and has three passive joints (black circles). One of the four bars is fixed relative to the world frame (dark grey).

A. Representing Rigid Bodies

We represent the position of a rigid body in global coordinates with its CoM $\mathbf{c} \in \mathbb{R}^3$. This simplifies the formulation of our CoM objective. Moreover, it means that our method could easily be integrated with common dynamics representations, where the use of frames centered at $\mathbf{c}$ is often preferable because they enable the decoupled treatment of the linear and angular motions in the Newton-Euler equations of the bodies.

To represent a body's orientation, we rely on 4-coordinate unit quaternions $\mathbf{q}$, as they do not suffer from singularities. This comes at the cost of enforcing unit length constraints.

Summarily, we represent the i-th component in an assembly with a 7-coordinate state vector $\mathbf{s}_i$, consisting of the position and orientation of the body, and stack them for an n-body assembly to represent its maximal coordinate state with the state vector $\mathbf{s} \in \mathbb{R}^{7n}$. For example, for the four-bar linkage in Fig. 2, the number of components is 4, and the state of the assembly is represented with a 24-vector $\mathbf{s}$. After stepping the actuators, we solve for the state of the assembly so that all constraints are fulfilled. Hence, this state is considered *unknown* in forward kinematics.

B. Representing Kinematic Constraints

Passive or active joints constrain the relative motion between pairs of bodies. To formulate constraints, we attach local frames that rigidly move with the bodies, to either body i and j (see Figs. 3 and 4). We then transform these two frames to global coordinates, and impose restrictions on their relative translations or rotations when solving for the unknown state for a stepped configuration.

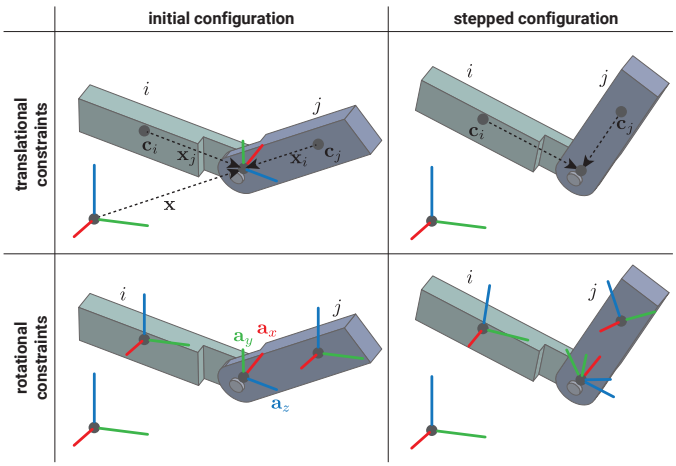

Fig. 3. A revolute joint in its initial (left) and a stepped configuration. To formulate constraints, we attach local frames to either body. After stepping the motors, the origin of frame i, transformed to global coordinates, has to equal the transformed origin of frame j (translational constraints). In addition, the transformed hinge axis $\mathbf{a}_x$ on body i has to remain orthogonal to the other two transformed axes, $\mathbf{a}_y$ and $\mathbf{a}_z$, of frame j at all times (rotational constraints).

a) Passive Joints: Without loss of generality, it is convenient to assume that all bodies, and the global coordinate frame, have the same orientation in the initial configuration. To see why this is advantageous, it is best to study a specific example: If the orientations of bodies i and j are set to the identity, the three frame axes $\mathbf{a}_x$, $\mathbf{a}_y$, and $\mathbf{a}_z$ are the same in global and local body coordinates (Fig. 3, bottom row, left). Moreover, the initial location of the joint in global coordinates, $\mathbf{x}$, can be transformed to local coordinates, $\mathbf{x}_i$ and $\mathbf{x}_j$, by subtracting the initial positions of either body (top row, left). Assuming the revolute axis to point in the direction of the x-axis in either local frame, we formulate the constraint for a revolute joint by asking the transformed revolute axis on body i to remain orthogonal to the transformed y- and z-axes of body j (bottom row, right)

$$\boldsymbol{\mathcal{J}}(\mathbf{s}_i, \mathbf{s}_j) = \begin{bmatrix} \mathbf{R}(\mathbf{q}_i)\mathbf{x}_i + \mathbf{c}_i - \mathbf{R}(\mathbf{q}_j)\mathbf{x}_j + \mathbf{c}_j \\ (\mathbf{R}(\mathbf{q}_i)\mathbf{a}_x) \cdot (\mathbf{R}(\mathbf{q}_j)\mathbf{a}_y) \\ (\mathbf{R}(\mathbf{q}_i)\mathbf{a}_x) \cdot (\mathbf{R}(\mathbf{q}_j)\mathbf{a}_z) \end{bmatrix} = \mathbf{0}$$

where $\mathbf{R}$ is the rotation matrix that we extract from the unit quaternion with a variant of the Euler-Rodrigues formula.

Note that this particular formulation with dot products between pairs of transformed axes for rotational constraints, leads, in general, to the minimal number of constraints: A revolute joint constrains all but one rotational degree of freedom between the two bodies. Hence, the minimal number of constraints is 5. Common joint types, such as Cartesian, cylindrical, fixed, prismatic, spherical, and universal joints, can be formulated analogously.

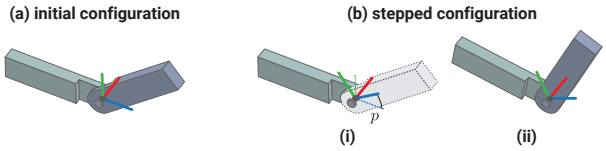

Fig. 4. A Revolute actuator in its initial (a) and stepped (b) configuration. By first rotating the local y-axis on body i by p about the local hinge axis $\mathbf{a}_x$ (i), and asking this axis, after transforming it to global coordinates, to remain orthogonal to the transformed z-axis of the frame on body j (ii), we complement the revolute joint constraints to form an actuator.

b) Active Joints: By parameterizing the remaining degrees of freedom, we can complement every joint type with less than 6 constrained degrees of freedom (all but the fixed joint) to form an actuator. For example, for a revolute actuator (Fig. 4), we parameterize the remaining rotational degree of freedom about the revolute axis with a parameter p. To form a constraint, we first rotate the y-axis about the revolute axis in local coordinates of body i (i), before transforming it to global coordinates and asking it to remain orthogonal to the transformed z-axis of body j (ii)

$$\boldsymbol{\mathcal{A}}(p, \mathbf{s}_i, \mathbf{s}_j) = \begin{bmatrix} \boldsymbol{\mathcal{J}}(\mathbf{s}_i, \mathbf{s}_j) \\ (\mathbf{R}(\mathbf{q}_i)\,\mathbf{R}[p, \mathbf{a}_x]\,\mathbf{a}_y) \cdot (\mathbf{R}(\mathbf{q}_j)\mathbf{a}_z) \end{bmatrix} = \mathbf{0}.$$

To rotate the y-axis, we apply the rotation matrix $\mathbf{R}[p, \mathbf{a}_x]$ that represents a rotation of p about the revolute axis in local body coordinates of i. A prismatic actuator, and also less common actuators with more than one actuated degree of freedom, can be formulated analogously.

c) Unary Constraints: So far, we have discussed binary constraints that constrain the relative motion between pairs of bodies. However, to guarantee that the constraint Jacobian is full rank, we need to constrain the rigid body motion of the *overall* assembly. To this end, we fix at least one body in space with 6 constraints of the form

$$\boldsymbol{\mathcal{U}}(\mathbf{s}_i) = \begin{bmatrix} \mathbf{c}_i - \overline{\mathbf{c}} \\ \mathrm{Im}(\mathbf{q}_i) \end{bmatrix} = \mathbf{0}$$

where the position of the body is kept at its initial position $\overline{\mathbf{c}}$, and the initially zero imaginary part of the unit quaternion that represents the bodies orientation, is kept zero (see dark grey component in Fig. 2).

C. Solving for the Kinematic Motion

To solve for the kinematic motion, we collect all actuation parameters in a parameter vector $\mathbf{p}$, and all joint, actuator, and unary constraints, together with n unit length constraints of the form $\mathbf{q}_i \cdot \mathbf{q}_i - 1$, in a constraint vector

$$\boldsymbol{\mathcal{C}}_{\text{FK}}(\mathbf{p}, \mathbf{s}(\mathbf{p})) = \mathbf{0}$$

that we ask to remain zero after setting $\mathbf{p}$ to new values. Because the state of the assembly changes whenever we make changes to parameters, there is an implicit dependence, $\mathbf{s}(\mathbf{p})$, between the two.

Note that there are often more constraints than unknowns, even though we formulate the joint and actuator constraints

with the minimal number of equations. Especially assemblies that contain planar subassemblies result in overconstrained systems, for example the four-bar linkage in Fig. 2: The four bodies amount to $4 \times 7 = 28$ unknown states. Revolute joints add 5 constraints, revolute actuators add 6, and we have 4 quaternion unit-length constraints, and one grounded component, for a total of 31 constraints. Only because all revolute axes point in the same direction, the motion is kinematically feasible.

Forward kinematics, in its general form, is therefore a nonlinear least squares problem

$$\min_{\mathbf{s}} f_{\text{FK}}(\mathbf{s}) \quad \text{with} \quad f_{\text{FK}}(\mathbf{s}) = \frac{1}{2}\boldsymbol{\mathcal{C}}_{\text{FK}}(\mathbf{s})^T \boldsymbol{\mathcal{C}}_{\text{FK}}(\mathbf{s}) \tag{1}$$

that we can solve with Gauss-Newton steps

$$\left(\frac{\partial \boldsymbol{\mathcal{C}}_{\text{FK}}(\mathbf{s}_k)}{\partial \mathbf{s}}\right)^T \frac{\partial \boldsymbol{\mathcal{C}}_{\text{FK}}(\mathbf{s}_k)}{\partial \mathbf{s}} \mathbf{d}_k = -\left(\frac{\partial \boldsymbol{\mathcal{C}}_{\text{FK}}(\mathbf{s}_k)}{\partial \mathbf{s}}\right)^T \boldsymbol{\mathcal{C}}_{\text{FK}}(\mathbf{s}_k)$$

where $\mathbf{d}_k$ is the k-th Newton direction along which we perform standard backtracking line search [23]. Because the translational and rotational parts of the constraints are scaled differently, the constraint Jacobian can be conditioned unfavorably. To counteract this ill-conditioning, we scale all rotational constraints (e.g., the first three constraints in $\boldsymbol{\mathcal{J}}$) with the average distance between pairs of passive or active joints. Because f_{FK} is well-behaved, and minimizations are started from previous solutions which are close to the optimum, the Gauss-Newton algorithm is the preferred algorithm. However, for remote starting points, the Levenberg-Marquardt algorithm may outperform the Gauss-Newton method [23].

IV. Inverse Kinematics

For IK, the problem is reversed. Instead of solving for the state after updating the actuation parameters $\mathbf{p}$, we seek to find optimal actuation parameters, and indirectly also the assembly state, such that a single or several end effectors (e.g., feet of a legged robot) follow a desired trajectory.

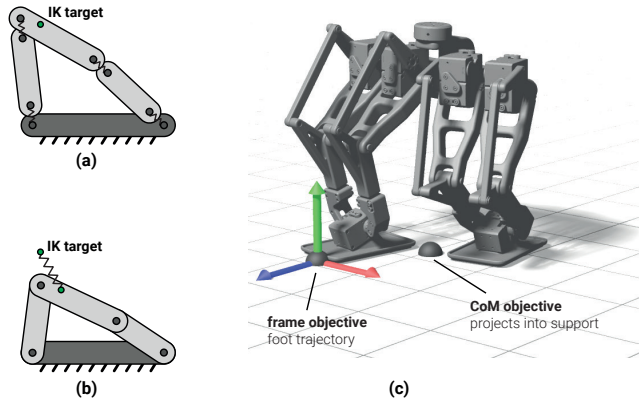

Fig. 5. If we ask for an IK target that cannot be reached with the physical robot (a), we favor a solution where the distance to the IK target is minimal while the robot remains in an assembled state (b). In addition to position and orientation objectives, we provide the user with control over the CoM location (c).

A. Desiderata

A problem that we seek to address within the scope of this paper, is to safeguard against parameter configurations that are not physically feasible. For example, for the four-bar linkage in Fig. 5 (a), a full rotation of the revolute actuator is not possible without causing a disassembly. To make our IK a versatile tool in different robotic animation tasks, we further add the support of a sufficiently complete set of IK targets to our list of desiderata. In addition to tracking end effector positions and orientations, we add a CoM objective. This means we can ensure that the CoM projects into the support spanned by dynamically changing contact areas with the ground (Fig. 5 (b)), enabling our pipeline to be used for quasi-static walking (Fig. 1).

Moreover, if actuation configurations are physically feasible, targets should be satisfied exactly. Otherwise, if not all of the targets can be satisfied simultaneously, the user should have control over their relative importance.

B. Specifying IK Targets

To achieve the second goal, it is best to formulate IK targets as objectives rather than hard constraints.

To formulate IK objectives, we attach frames to individual bodies i (Fig. 5, frame objective). Frames are specified in global coordinates, with respect to the initial configuration. The initial position of the frame, $\mathbf{x}$, is transformed to local body coordinates by subtracting the initial position of body i, resulting in the position $\mathbf{x}_i$. Because all orientations of all bodies are set to the identity in the initial configuration, the three global frame axes $\mathbf{a}_x$, $\mathbf{a}_y$, and $\mathbf{a}_z$ are the same in local coordinates.

a) Frame Objectives: To penalize differences between the current position of a frame, and a target position $\hat{\mathbf{x}}$, we minimize *position objectives* of the form

$$f_{\text{pos}}(\mathbf{s}) = \frac{1}{2}\|\mathbf{R}(\mathbf{q}_i)\mathbf{x}_i + \mathbf{c}_i - \hat{\mathbf{x}}\|^2.$$

With our *orientation objectives*

$$f_{\text{ori}}(\mathbf{s}) = \sum_{a \in \{x,y,z\}} (1 - (\mathbf{R}(\mathbf{q}_i)\mathbf{a}_a) \cdot \hat{\mathbf{a}}_a),$$

we penalize differences between frame and target orientations.

Together, these two objectives can be used to formulate *frame objectives*. With 3-component weights $\mathbf{w}_{\text{pos}}$ and $\mathbf{w}_{\text{ori}}$ that vary per IK target, and weigh the relative importance of the position error along the three global coordinate axes, and the error about the three local frame axes, we provide the user with a fine-grained control over tradeoffs between the individual objectives. By setting a subset of weights to zero, only a single axis, or two axes objectives can be formulated.

b) CoM Objective: To enable control of the CoM of the robot, we penalize differences between a user-specified target $\hat{\mathbf{c}}$ and the mass weighted sum of the positions of the individual bodies

$$f_{\text{CoM}}(\mathbf{s}) = \frac{1}{2}\left\|\left(\sum_{i=1}^{n} \frac{M_i}{M}\mathbf{c}_i\right) - \hat{\mathbf{c}}\right\|^2.$$

where M is the sum of all body masses M_i. Like for our position objectives, we introduce a 3-weight vector $\mathbf{w}_{\text{CoM}}$ to separately control the error along the three global coordinate axes. For example, for the Walker example, we disable the vertical component of this objective.

C. Solving for Actuation Parameters

For forward kinematics, we set actuation parameters to new values, then solve for the state vector that minimizes the objective f_{FK}. At the minimum, the gradient of the forward kinematics objective is zero.

a) Implicit Formulation: Hence, a first inverse kinematics formulation that comes to mind is enforcing first-order optimality of the forward kinematics as constraint, then evaluating the IK objective with the implicitly defined state that minimizes f_{FK}

$$\min_{\mathbf{p}} f_{\text{IK}}(\mathbf{s}(\mathbf{p})) \quad \text{s.t.} \quad \nabla_{\mathbf{s}} f_{\text{FK}}(\mathbf{p}, \mathbf{s}(\mathbf{p})) = \mathbf{0}. \tag{2}$$

This problem can be solved with a standard quasi-Newton method by enforcing the constraints implicitly. To this end, forward kinematics is first performed whenever the IK objective, or its gradient, is evaluated. To compute the analytical gradient, we then apply the chain rule and make use of the implicit function theorem to compute the sensitivity of the state with respect to the parameters: in a small neighborhood around the FK solution, the derivative of the first-order optimality constraint remains zero, and hence

$$\nabla_{\mathbf{p}} f_{\text{IK}} = \frac{\partial f_{\text{IK}}}{\partial \mathbf{s}} \frac{\mathrm{d}\mathbf{s}}{\mathrm{d}\mathbf{p}} \quad \text{with} \quad \frac{\mathrm{d}\mathbf{s}}{\mathrm{d}\mathbf{p}} = -\left(\frac{\partial^2 f_{\text{FK}}}{\partial \mathbf{s}^2}\right)^{-1} \frac{\partial^2 f_{\text{FK}}}{\partial \mathbf{p} \partial \mathbf{s}}.$$

In evaluations of the IK gradient, we need, besides the Hessian of the FK objective, the Jacobian of f_{FK} with respect to the state and actuation parameters. All necessary derivatives can be computed with a symbolic differentiation package. To keep our formulation general, we take symbolic derivatives of each individual constraint type, then assemble the gradients, Jacobians, and Hessians for a specific assembly, at runtime.

However, the first-order optimality constraint does *not* guarantee that the FK objective evaluates to zero, meaning that the FK constraints $\boldsymbol{\mathcal{C}}_{\text{FK}}$ equal zero. When setting the actuation parameters to values that are not physically feasible, FK returns the minimal norm solution. If we ask for IK targets that are not physically feasible, this implicit formulation will converge to a solution where all IK targets are fulfilled, but the robot is in a disassembled state (see Fig. 5 (a)).

b) State-based Formulation: To fulfill our first desideratum, we instead rely on a state-based formulation, and read off actuation parameters in a post-processing step. To this end, we replace every active joint with its corresponding passive joint. More formally, we replace all constraints $\boldsymbol{\mathcal{A}}$ that depend on the actuation parameters $\mathbf{p}$ with the corresponding passive joint constraints $\boldsymbol{\mathcal{J}}$ in $\boldsymbol{\mathcal{C}}_{\text{FK}}$, resulting in the reduced constraint vector $\boldsymbol{\mathcal{C}}_{\text{IK}}$. For example, for our four bar linkage, we replace the revolute actuator with a revolute joint, reducing the overall number of constraints from 31 to 30.

This procedure turns our robots into "puppets" that would collapse under gravity. This is precisely what we want, because (1) it enables us to move individual components with IK objectives without resistance from actuators, and (2) we can formulate an optimization problem directly on states

$$\min_{\mathbf{s}} f_{\text{IK}}(\mathbf{s}) + R(\mathbf{s}) \quad \text{s.t.} \quad \boldsymbol{\mathcal{C}}_{\text{IK}}(\mathbf{s}) = \mathbf{0} \tag{3}$$

where we *enforce* the robot to remain in an assembled configuration with a set of equality constraints, and $f_{\text{IK}}(\mathbf{s})$ sums up all IK objectives. Because the IK objectives may not fully prescribe the robot state, we add a regularizer $R(\mathbf{s})$ to "choose" a solution in this subspace.

A regularizer that leads to intuitive solutions keeps the current state close to the previous state $\bar{\mathbf{s}}$ of the robot

$$R_{\text{state}}(\mathbf{s}) = \frac{1}{2}\|\mathbf{s} - \bar{\mathbf{s}}\|^2.$$

This regularizer ensures that the problem is always well-posed, even if only a single IK objective is specified. Hence, this formulation provides an ideal tool to iteratively explore how to best transfer a motion over to a robot. Another advantage of the state-based formulation is that only building blocks from forward kinematics (only derivatives of constraints with respect to states) are needed, making it significantly easier to add IK support to an existing forward kinematics implementation. For optimization, we use standard line search Sequential Quadratic Programming (SQP) [23].

After optimization, we read off the actuation parameters from pairs of states of neighboring bodies. To disambiguate and return the "correct" angle, we compare extracted to previous angles for rotational degrees of freedom and add or subtract multiples of 2π to obtain a temporally-consistent trajectory.

D. Regularizing Velocities

As we illustrate with the retargeting of an arm motion onto our Boxer character [24] in Fig. 7, velocities of individual bodies can go to infinity even though we move the end-effector slowly and continuously in Cartesian space. While we could use our state regularizer to stay a safe distance away from such kinematic singularities, we would need to set the weight w_{state} to a high value, resulting in retargeting results that look like the robot is moving through a viscous fluid. Hence, a more fine-graded control is necessary.

While we could formulate standard actuator-centric constraints to keep the actuator velocities within the limits supported by the hardware, we favor a body-centric regularization: as soon as a hard constraint becomes active, the optimization "walks" along the constraint manifold until the next constraint becomes active, leading to undesired discontinuities in the retargeting result. With a regularizer, we provide the user with better control over the smoothness of the resulting motion, and distribute the "limits" among the affected components. By lowering the bound at which the regularizer becomes active, we can always find solutions where all actuator velocities remain in the required limits as we demonstrate in our Results section.

To penalize angular velocities from taking on too high values, we compute the dot products of the three frame axes of

the current orientation of the body (columns of $\mathbf{R}(\mathbf{q}_i)$), and the corresponding frame axes of its previous orientation (columns of $\mathbf{R}(\bar{\mathbf{q}}_i)$). If these dot products (compactly expressed with the trace operator) become too small, the regularizer

$$R_{\text{vel}}(\mathbf{s}) = \sum_{i=1}^{n} g_{\text{reg}} \left(3 - \text{tr}[\mathbf{R}^T(\bar{\mathbf{q}}_i)\mathbf{R}(\mathbf{q}_i)]\right) \tag{4}$$

becomes active where the function

$$g_{\text{reg}}(x) = \begin{cases} \ln\left(\frac{x}{t_{\text{vel}}}\right)^3, & \text{if } \; x > t_{\text{vel}} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

ensures that the regularizer, and its gradient, are only taking on non-zero values above a user-specified threshold t_{vel}. A similar regularizer could be formulated for body positions, if required. With a weight w_{vel}, we further control the importance of this regularizer relative to f_{IK} and R_{state}.

V. Results

We demonstrate our pipeline on three examples, highlighting different aspects and functionality of our framework. For all the examples, the input to our pipeline is a digital animation created using Autodesk Maya, from which we extract the relevant tracking frames with a custom script. We use a custom Solidworks plugin to extract the kinematic models directly from the CAD models. See Tab. I for key statistics for the three demos. Timings are reported for an Intel Core i7-6700K CPU. For all our motions, the input animation is sampled at 1200 frames per second. All of our robots use Dynamixel XH-430 actuators. Unless otherwise noted, regularization and objective weights are set to unity.

1) Validation: To demonstrate that our pipeline supports all common joint types, we created a 1-Dof example mechanism that incorporates all common joints. We attach an IK target to the end effector, and solve the IK on the mechanism. See Fig. 6 (left). This illustrates the generality of our method. We also created a serial chain combining 6 revolute and 6 prismatic actuators, Fig. 6 (right), and ran IK for a single frame attached to the end effector. Note that with 12 DoFs, a single frame is insufficient to fully constrain the state of the robot; our regularizer that pulls us towards the previous state makes the problem well-posed.

We also investigate how the IK performance changes with the input framerate, by subsampling the animation for the "dancer" robot. We find that as the framerate is reduced, there is only a slight increase in the computation time, see Fig. 6 (c). At lower framerates, our performance is significantly faster than real-time.

2) Boxer: Our pipeline supports arbitrary kinematic structures, and this of course includes serial robots. As a first example, we retarget an arm motion onto the 4-DoF arm of our Boxer robot [24]. The shoulder joint has 3 degrees of freedom, and can exhibit a kinematic singularity. This is also representative of standard robot arms. We place a single tracking frame at the robot end effector, tracking both position and orientation.

To demonstrate how our velocity regularizer avoids singularities, we generated an input animation that moves through

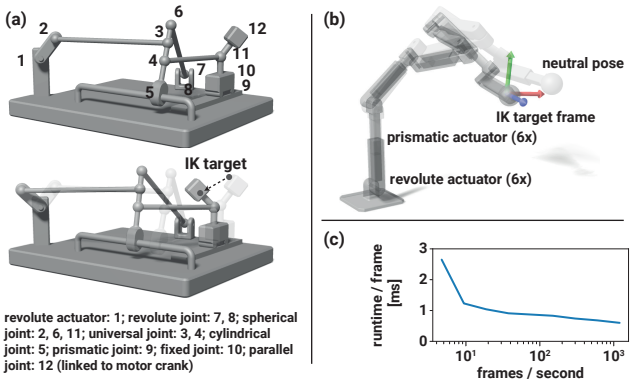

Fig. 6. **Validation.** (a) Test mechanism with all common joint types; when specifying a target motion on the end effector (bottom image) we can solve the resulting IK problem. (b) Serial robot with 6 revolute and 6 prismatic actuators, tracking a single frame at the end effector. (c) Computation time per frame for changing input framerate.

TABLE I
Key statistics

Example	$\#act$	$\#jnt$	$\#comp$	$targets$	t
Boxer	4	-	5	Fr (1)	0.14
Boxer (reg.)	4	-	5	Fr (1)	0.23
Dancer	8	4	12	Fr (1)	0.68
Walker	12	24	31	Fr (3), CoM	13.99

$\#act$: number of actuators; $\#jnt$: number of passive joints; $\#comp$: number of components; $targets$: the types and number of IK targets. Fr: frame, CoM: Center of Mass; t: algorithm runtime per time step (ms). For the Boxer, we report timings with and without the regularizer.

the kinematic singularity. In the animation, which uses a spherical joint at the shoulder, the angular velocities are smooth and bounded. However, a naive motion retargeting leads to discontinuous motor profiles with high peak velocities. By increasing the velocity regularizer weight, we are able to circumvent the singular configuration, and obtain motor profiles that are smooth and well within the velocity limits of the hardware. Fig. 7 shows the joint velocities over time for the animation, for the two cases.

We set the state regularizer weight to 2×10^{-4}. For the velocity regularizer, we set the weight to $1 \times 10^{-}4$, and the threshold to 1.11×10^{-5} (corresponding to 90% of the rated maximum actuator velocity). For the unregularized version, the target is tracked perfectly: mean positional error 0.02 mm; max error 3.89 mm (at first singularity), but the peak joint velocity is 147.04 rad/s which cannot be realized. With the regularized version, the peak joint velocity is 5.90 rad/s, the mean positional error for the sequence is 1.66 mm, with a peak error at the singularity of 24.03 mm.

We also refer to the supporting video, where the different motion sequences can be seen on both the simulated and physical robot; there is excellent agreement between the two.

3) Dancer: As an example of non-trivial parallel kinematics, we consider a pair of animatronic legs featuring a novel kinematic structure that allow it to perform highly expressive motions, in particular with the hips. The feet are fixed in place, so overall the mechanism forms a kinematic loop with 12 components and 12 joints. See Fig. 8 (left).

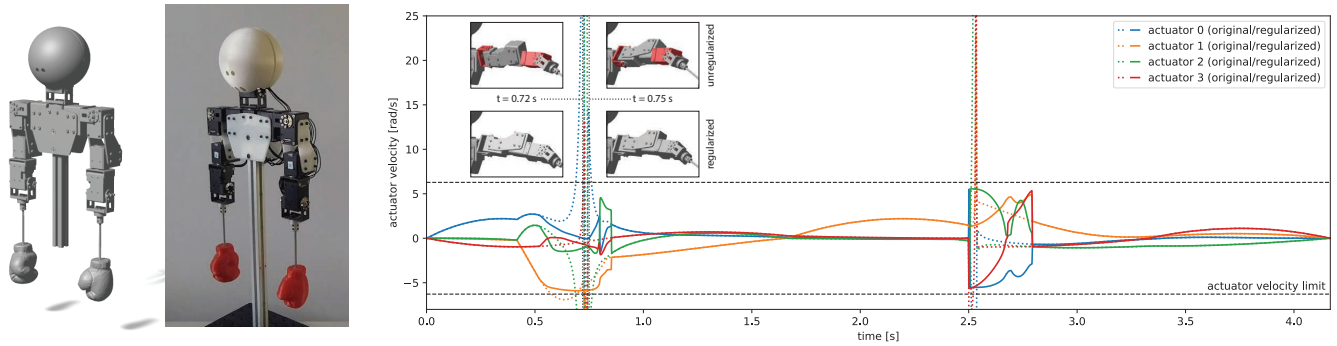

Fig. 7. **Boxer.** Left: the simulated and physical robot, shown in the neutral pose. Right: plot of the joint velocities over time, for the unregularized (dotted lines) and regularized (solid lines) versions. The inset views show the configuration change of the robot before and after the velocity spike, with motors in red indicating that the velocity limit is exceeded. See also the supporting video.

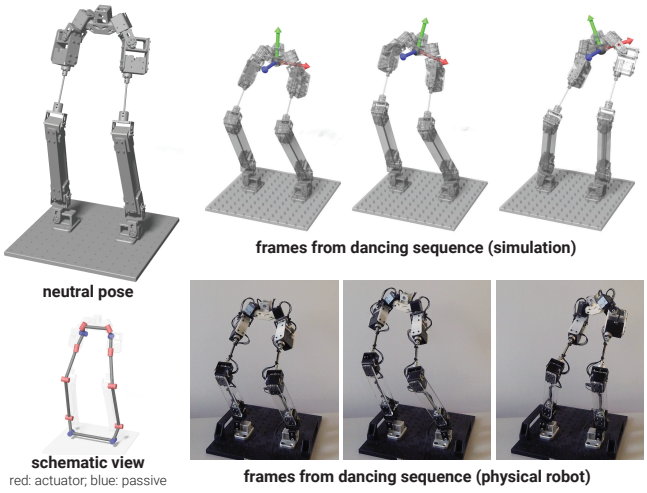

Fig. 8. **Dancer.** The robot features a novel kinematic structure (see schematic view) with 6 degrees of freedom, 12 joints, and 8 actuators, that allows for expressive hip motions. In the simulated dancing frames, the tracked frame on the pelvis can also be seen.

As an animation input, we use a dancing sequence. As the robot has 6 degrees of freedom, a single frame attached to the pelvis is sufficiently for uniquely specifying the robot motion. We place actuators at 8 of the joints, thus making the robot overactuated. This means that singularities in the operating range can be eliminated. Our IK formulation handles the overactuation without modification. We set the regularization weight to 1×10^{-6}.

Frames from the resulting sequence are shown in Fig. 8 (right), and we also refer to the supporting video. The IK tracks the animation input perfectly, with maximal positional and rotational objective errors of 2.46×10^{-5} mm and 1.22×10^{-5} rad respectively.

4) Walker: As an additional demonstration of our pipeline, we consider the serial-parallel bipedal walking robot presented by Gim et al [21, 22]. This robot features a novel 6-DoF leg mechanism where a single actuator is placed in the foot and the remaining 5 actuators are placed in the hips, reducing the moving mass of the leg.

In the original paper, the authors derive the forward and inverse robot kinematics analytically; while this can be done for the robot considered here, it is a laborious and time-consuming task. Moreover, if joint positions or orientations were to be perturbed, making joint axes non-parallel, analytic inverse kinematics would become significantly more challenging. Our approach, in contrast, is compatible with arbitrary kinematic structures and does not struggle with non-parallel joints.

For slow walking, a quasi-static approach is sufficient. The requirement for stable bipedal walking is then that the overall CoM of the robot must project into the support polygon spanned by the feet currently in contact with the ground. We take as input a walking animation (Fig. 1: top row). Note that the animated character has a "normal" joint configuration with hips, knees, and ankles; our pipeline takes care of retargeting this onto the serial-parallel leg mechanism.

We first perform a naive motion retargeting, by tracking frames on the two feet and on the pelvis. While the motion is visually close to the input animaton (max tracking error: 0.51 mm), the physical robot falls over immediately as the CoM does not project into the support. For this, the regularization weight is set to 1×10^{-2}.

Next, we add a CoM objective to the IK. The feet objective weights are set to $[100, 100, 100]$ and $[10, 10, 10]$ for the orientation and position, respectively. The orientation weight is high, as the size of the feet means that orientation errors cause dragging on the ground. The CoM position weight is set to $[10, 0, 10]$ (with y being the vertical direction), and the pelvis frame weight is set to $[0, 1, 0]$. The pelvis frame weight is set lower, as it is not critical for the robot's function.

In the single-support phase of the gait, we ask for the CoM to lie within the support of that foot, and in the dual-support phase we ask the CoM to move along a line connecting the two feet. We achieve a max positional tracking error of the frames of 1.11 mm, and a max tracking error for the CoM objective of 1.30 mm. Moreover, as shown in Fig. 9, the robot is able to walk without falling over. See also the supporting video for the full walking sequences in simulation and on the physical robot.

VI. Conclusion

We have devised a versatile inverse kinematics for the retargeting of expressive motions onto complex robots with kine-

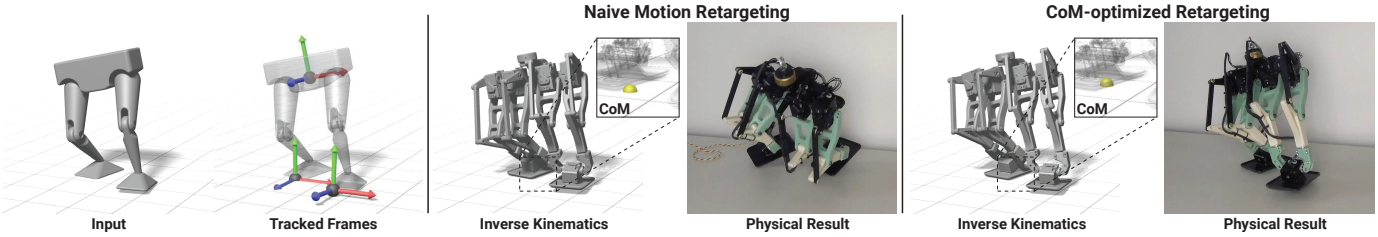

Fig. 9. **Walker.** Starting from a walking animation, we track frames attached to the pelvis and feet. A naive motion retargeting is visually close to the input, but the robot falls over as the CoM does not project into the support. With a CoM objective, we can successfully retarget the motion onto the walking robot.

matic loops. Compared to state-of-the-art approaches typically used for parallel and series-parallel robots, our formulation supports *arbitrary* mechanism topologies.

One limitation of our modeling is time complexity. For use in closed-loop control, our IK is ill-suited for high-DoF mechanical systems. An interesting future direction is the exploration of mixed representations where reduced coordinates are used for the minimum spanning tree of the robot, and a maximum coordinate representation for "closing" the loops [18]. Such hybrid formulations reduce computational complexity, and also ensure that joint constraints are satisfied. However, a challenge is that mechanisms that contain joints or actuators with more than a single translational or rotational degree of freedom (joints other than of revolute or prismatic type), may require additional cuts.

In our IK formulation, we have not explicitly treated all types of kinematic singularities. While we *regularize* all types of singularities in our IK formulation with the help of our regularizers, we cannot guarantee that the motion is feasible on the physical robot if singularities other than the velocity type we handle, are present in the workspace. For example, while we regularize velocities, we do not currently limit accelerations near singularities. An explicit treatment of *all* types of kinematic singularities is an exciting future direction.

ACKNOWLEDGMENT

We thank (alphabetically) Andréane d'Arcy-Lepage, Alfredo Ayala, Kyle Cesare, Michael Hopkins, Scott LaValley, Tony Martin, Tanner Rinke and Krishna Tamminana for insightful discussions, and Maurizio Nitti for the artistic input. The technical formulation proposed in the paper was developed in collaboration with Stelian Coros.

REFERENCES

[1] S. Kumar, H. Whrle, J. de Gea Fernndez, A. Mller, and F. Kirchner, "A survey on modularity and distributivity in series-parallel hybrid robots," *Mechatronics*, vol. 68, p. 102367, 2020.

[2] A. Aristidou, J. Lasenby, Y. Chrysanthou, and A. Shamir, "Inverse kinematics techniques in computer graphics: A survey," *Computer Graphics Forum*, vol. 37, no. 6, pp. 35–58, 2018.

[3] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd ed. Pearson, 2004.

[4] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics," in *2001 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, 2001, pp. 298–303.

[5] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," University of California, Tech. Rep., 2009.

[6] A. N. Pechev, "Inverse kinematics without matrix inversion," in *2008 IEEE International Conference on Robotics and Automation (ICRA)*, 2008, pp. 2005–2012.

[7] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-priority based redundancy control of robot manipulators," *International Journal of Robotic Research*, vol. 6, no. 2, pp. 3–15, 1987.

[8] J. Zhao and N. I. Badler, "Inverse kinematics positioning using nonlinear programming for highly articulated figures," *ACM Transactions on Graphics*, vol. 13, no. 4, 1994.

[9] R. Boulic, R. Mas, and D. Thalmann, "Inverse kinetics for center of mass position control and posture optimization," in *Image Processing for Broadcast and Video Production*. Springer, 1995, pp. 234–249.

[10] J.-P. Merlet, *Parallel robots*. Springer, 2005, vol. 128.

[11] O. Bohigas, M. Manubens, and L. Ros, *Singularities of robot mechanisms: Numerical computation and avoidance path planning*. Springer, 2016, vol. 41.

[12] K. Wen and C. M. Gosselin, "Forward kinematic analysis of kinematically redundant hybrid parallel robots," *Journal of Mechanisms and Robotics*, vol. 12, no. 6, p. 061008, 2020.

[13] M. Schappler, S. Tappe, and T. Ortmaier, "Modeling parallel robot kinematics for 3t2r and 3t3r tasks using reciprocal sets of euler angles," *Robotics*, vol. 8, no. 3, p. 68, 2019.

[14] J. G. d. Jalon and E. Bayo, *Kinematic and Dynamic Simulation of Multibody Systems: The Real Time Challenge*. Springer-Verlag, 1994.

[15] M. Raghavan and B. Roth, "Inverse Kinematics of the General 6R Manipulator and Related Linkages," *Journal of Mechanical Design*, vol. 115, no. 3, pp. 502–508, 1993.

[16] D. Kuehn, F. Bernhard, A. Burchardt, M. Schilling, T. Stark, M. Zenzes, and F. Kirchner, "Distributed computation in a quadrupedal robotic system," *International Journal of Advanced Robotic Systems*, vol. 11, no. 7, p. 110, 2014.

[17] N. A. Radford, P. Strawser, K. Hambuchen, J. S. Mehling, W. K. Verdeyen, A. S. Donnan, J. Holley, J. Sanchez, V. Nguyen, L. Bridgwater, *et al.*, "Valkyrie: Nasa's first bipedal humanoid robot," *Journal of Field Robotics*, vol. 32, no. 3, pp. 397–419, 2015.

[18] W. Blajer, W. Schiehlen, and W. Schirm, "Dynamic analysis of constrained multibody systems using inverse kinematics," *Mechanism and Machine Theory*, vol. 28, no. 3, pp. 397–405, 1993.

[19] J. Nielsen and B. Roth, "On the kinematic analysis of robotic mechanisms," *International Journal of Robotics Research*, vol. 18, no. 12, pp. 1147–1160, 1999.

[20] A. Nayak, S. Caro, and P. Wenger, "Kinematic analysis of the 3-rps-3-spr series–parallel manipulator," *Robotica*, vol. 37, no. 7, pp. 1240–1266, 2019.

[21] K. G. Gim, J. Kim, and K. Yamane, "Design of a serial-parallel hybrid leg for a humanoid robot," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–5.

[22] K. Gim, J. Kim, and K. Yamane, "Design and fabrication of a bipedal robot using serial-parallel hybrid leg mechanism," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 5095–5100.

[23] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

[24] S. Hoshyari, H. Xu, E. Knoop, S. Coros, and M. Bächer, "Vibration-minimizing motion retargeting for robotic characters," *ACM Transactions on Graphics*, vol. 38, no. 4, pp. 1–14, 2019.