Singularity-tolerant inverse kinematics for bipedal robots: An efficient use of computational power to reduce energy consumption

Salman Faraji* and Auke Jan Ijspeert*

***Abstract*—We propose a nonlinear inverse kinematics formulation which solves for positions directly. Compared to various other popular methods that integrate velocities, this formulation can better handle fast, asymmetric and singular-postured balancing tasks for humanoid robots. We also introduce joint position and velocity boundaries as inequality constraints in the optimization to ensure feasibility. Such boundaries provide safety when approaching or getting away from joint limits or singularities. Besides, mixing positions and velocities in our proposed algorithm facilitates recovery from singularities, which is very difficult for conventional inverse kinematics methods. Extensive demonstrations on the real robot prove the applicability of the proposed algorithm while improving power consumption. Our formulation automatically handles different numerical and behavioral difficulties rising from singularities, which makes it a reliable low-level conversion block for different Cartesian planners.**

I. INTRODUCTION

Bipedal robots are often mechanically very complex, designed to perform various types of tasks apart from walking. The location and the number of degrees of freedom (DoF) are mainly inspired by the human skeletal system, comprising limbs with at least six joints. From a control perspective, it becomes difficult however to plan a trajectory for each joint individually to perform a desired motion at the end-effector level. Therefore, one prefers to transform complex joint-space formulations into Cartesian space [1], making trajectory planning easier. Using such transformations, one can easily convert Cartesian trajectories to joint motions, required by individual joints in the robot. Cartesian control is popular in manipulation [1], humanoid balance [2] and locomotion [3]. The output of this transformation can generate joint positions, velocities, or accelerations. The first two quantities are often used in position-controlled robots (Inverse Kinematics, *IK*), while accelerations are used more often with the full dynamics model of the robot [4], resulting in desired joint torques (Inverse Dynamics, *ID*).

Although *IK* conversion is nonlinear, there are various methods to solve it either in a closed form [5] or iteratively [6], both suitable for online control. The *ID* problem is more complex but linear with respect to accelerations, forces, and torques. One can solve this linear system in a closed form [4] or set up a quadratic optimization problem (QP) to consider boundary constraints [7]. These constraints can ensure satisfaction of physical limitations on joint torques, contact frictions, and joint positions. The latter constraint is typically realized by putting boundaries on accelerations based on Taylor series expansion of joint positions. In a previous work [8], we successfully combined our QP-based *ID* algorithm with advanced state-estimation and torque tracking methods and demonstrated various compliant balancing behaviors on the Coman robot. However, on one hand, the position limit constraint was not effective as it produced largely infeasible accelerations very close to boundaries. On the other hand, the algorithm was not numerically stable enough in singular positions. In practice, operating in crouched postures with bent knees caused higher torques and therefore, less precise tracking performance. Besides, the consequence over the long term was permanent spring deflections in series elastic elements, increased backlash, over-heating and large power consumption in addition to less human-like postures.

In the literature, there are plenty of algorithms proposed for stretched-knee walking. Apart from model-free walking approaches [9], which deal with singularity problem differently, a large number of model-based algorithms are based on preplanned knee trajectories, where other joint angles are adjusted to realize desired swing or Center of Mass (CoM) motions [10], [11]. Likewise, the walking planner proposed in [12] modifies a parametrized 2D CoM trajectory to limit large knee velocities close to singular positions. For arbitrary balancing tasks, however, such periodic trajectories can not be found. The idea of limiting velocities is inspiring, though, the time linkage between positions and velocities is encoded in the CoM trajectory, not in the *IK* method. In other words, preplanned Cartesian CoM velocities are found in a way to satisfy limitations on the velocity of the knee joint. The same paradigm is proposed in [13] for balancing tasks, but again, the trajectory planner handles stretched knee postures.

Given desired Cartesian trajectories, the *IK* or *ID* method is supposed to find joint trajectories that follow the task as precise as possible while satisfying physical constraints automatically. In this article, instead of focusing on planning trajectories, we aim at studying physical limitations and propose algorithms that handle them in a unified online control setup. We limit our study to *IK* methods to explore important aspects of joint position boundaries and singularities exclusively. In future, however, force/torque constraints of our previously developed *ID* method [8] will be combined with geometrical constraints studied in this article to address a larger number of hardware limitations.

*Biorobotics Laboratory, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

Digital Object Identifier 10.1109/LRA.2017.2661810

Stretched-leg postures are more convenient for the mechanical hardware of humanoid robots, but they introduce two major difficulties to the control problem:

1) Singularities: which mainly refer to the alignment of the hip, knee and ankle joints and lead to ill-conditioned Jacobians. Such postures limit controllability and might produce large velocities in the knee joint.
2) Joint position limits: which should be respected together with velocity limitations to avoid impacts, especially in the knee.

Handling these problems requires a robust method that safely approaches the singularity, does not vibrate, and safely leaves the singularity again. Besides, in all these phases, the method should find the closest solution to the desired task. Singularities and joint limits can be handled in planning level or inside the low-level IK algorithm. For any specific task, one can adjust the planner to match better with the geometry of the robot. However, we focus on the baseline *IK* method to make it robust and general, without modifying Cartesian planners.

For manipulators, robust *IK* methods [14], [15] have been developed to deal with singularity conditions robustly. These methods either bring the target position closer to the manipulator [15] or find the closest solution [14]. The former falls into planner-level category while the latter is more interesting as it provides a generic IK method, normally in the form of least-squares error minimization. To handle stability of trajectories in singular postures, one can add a damping term which is widely used in literature [14], [16], [17]. Such damping can improve numerical stability in redundant robots as well [16]. Although one can modify weightings in these unconstrained least-squares optimizations to avoid joint limits [18], expressing them explicitly in constrained optimizations is also popular [19]. A more general form of such optimization is formulated in [20] or [21] where velocities are strictly bounded or adaptively damped respectively.

All previously mentioned methods are based on time-integration, where the outcome of optimization is being integrated over time to obtain desired joint positions. Another class of *IK* methods solves the exact nonlinear constraint by performing many iterations in the same time-step [22]. Each iteration here is similar to solving a quadratic program like before, aiming at getting closer to the exact solution. Adding position and velocity limitations as well as singularities are less studied in this class of optimizations, mainly because of computational cost.

Focusing on previously-mentioned geometrical constraints, we formulate a general *IK* module that handles inequality constraints in the form of nonlinear optimizations. We use generalized-coordinate models instead of per-limb models and go beyond walking, to target arbitrary whole body balancing tasks for our floating-based humanoid robot in 3D. By reimplementing popular *IK* methods in the literature, we show that getting to singular positions and coming out in a safe manner can most of the time be problematic for *IK* methods based on time-integration. Therefore, the novelty of this work lies in proper analysis and handling of singularities and joint limitations via the proposed nonlinear method which combines positions, velocities and inequality constraints in the same optimization. Our analysis covers multiple behavioral and computational aspects, proving applicability of the proposed method for the real robot. This is demonstrated for a couple of different symmetric and asymmetric balancing tasks. The structure of this paper is as follows: in the next section, we formulate different *IK* optimization problems and present our proposed formulation. Next, we will demonstrate simulations and experiment results, characterizing the performance and handling physical limitations. Finally, we conclude the paper by discussing possible future improvements in the last section.

II. METHODOLOGY

The problem of inverse kinematics refers to the conversion of a set of Cartesian trajectories to joint-space. These trajectories which are called tasks hereafter, describe the translational or rotational motion of interesting points of the robot, for example CoM, hands, or feet. Given that sometimes tasks cannot be realized exactly due to the singularities or joint position limits, one might compromise a few, depending on the application and precision requirements. Imagine we have $x \in \mathbb{R}^M$ tasks, where a subset of size $N \leq M$ can be compromised. If the robot has $q \in \mathbb{R}^K$ Degrees of Freedom (DoF), the goal is to find q that satisfies $f(q) = x + \begin{bmatrix}\delta & 0\end{bmatrix}^T$ where the slack variable $\delta \in \mathbb{R}^N$ is to be minimized. Alternatively, one can find $\dot{q}$ that satisfies $[\partial f(q)/\partial q]\dot{q} = \dot{x} + \begin{bmatrix}\delta & 0\end{bmatrix}^T$ and integrate to get q over time. In case of adding limitations, an inequality of the form $l_b \leq g(q,\dot{q}) \leq u_b$ should be satisfied as well, where $l_b, u_b \in \mathbb{R}^K$ represent limits and $g(q,\dot{q})$ could be a nonlinear function. Is it better to optimize positions and then differentiate to find velocities or alternatively, find velocities first and then integrate them to get positions? This question has many aspects, including computational cost, tracking precision, robustness in singularities, ability to approach joint limits and ability to escape from singularities safely. Here, we consider five different *IK* algorithms and compare them regarding the previously mentioned criteria. The first three (IK_1, IK_2 and IK_3) are common in robotics while the other two (IK_4 and IK_5) are new ones proposed in this paper.

A. IK_1: Error integration

Similar to [14], one can find joint delta angles based on the task error and then integrate over time. To handle singularities, we also introduce damping factors (diagonal positive definite matrix R), resulting in the following unconstrained quadratic optimization problem:

$$\min_{\Delta q, \delta} \delta^T Q \delta + \Delta q^T R \Delta q \qquad (1)$$
$$f(q^-) + \frac{\partial f(q^-)}{\partial q^-}\Delta q = x + \begin{bmatrix}\delta \\ 0\end{bmatrix}$$

where q^- is the previous desired trajectory and $\Delta q \in \mathbb{R}^K$ is motion adjustment to be found. The matrix $R \in \mathbb{R}^{KxK}$ is the well-known damping in least-square methods [14]. Imagine S_N is a selection matrix which takes the N compromised tasks

out of the vector x if multiplied from left, i.e. $S_N x$. Similarly, S_{M-N} selects the rest of the tasks. Defining the Jacobian $J = \partial f(q^-)/\partial q^-$ and the error $E = x - f(q^-)$, the optimization of (1) has a closed form solution, calculated by setting the derivative of the Lagrange equation to zero:

$$\begin{bmatrix} \Delta q \\ \lambda \end{bmatrix} = \begin{bmatrix} J^T S_N^T Q S_N J + R & J^T S_{M-N}^T \\ S_{M-N} J & 0 \end{bmatrix}^\dagger \begin{bmatrix} J^T S_N^T Q s_N \\ S_{M-N} \end{bmatrix} E \tag{2}$$

where † is Moore-Penrose pseudo-inverse and λ is Lagrange coefficient for $M-N$ exact constraints. The desired trajectory and its derivative are then found by:

$$q = q^- + \Delta q, \quad \dot{q} = \frac{\Delta q}{\Delta t} \tag{3}$$

where Δt is the time-step. This fast formulation is equivalent to integrating velocities, only requiring to solve a linear system of certain dimensions. Note that the error, however, converges to zero over time-steps with certain dynamics.

B. IK_2: Conjugate gradient method

In this method, we perform all iterations in a single optimization at each time-step:

$$\min_{q,\delta} \delta^T Q \delta \tag{4}$$
$$f(q) = x + \begin{bmatrix} \delta \\ 0 \end{bmatrix}$$

This optimization is solved via conjugate gradient method where each iteration is solved similar to (2) with same damping mechanism. The velocities are then found by solving the following optimization:

$$\min_{\dot{q},\delta} \delta^T Q \delta + \Delta t^2 \dot{q}^T R \dot{q} \tag{5}$$
$$\frac{\partial f(q)}{\partial q} \dot{q} \Delta t = \dot{x} \Delta t + \begin{bmatrix} \delta \\ 0 \end{bmatrix}$$

where the time-step Δt is used to preserve consistency with (1). This optimization can be solved in closed form, similar to (2). Here, we find the exact solution for both positions and velocities at each time-step, yet without inequality constraints.

C. IK_3: Integrating errors with inequality constraints

To handle position and velocity limits, one can introduce boundary conditions to (1), i.e. constraining the first stage of IK_1. An arbitrary safety criterion can also be defined as a function of q and $\dot{q}$. For example:

$$g(q, \dot{q}) = \frac{(2q - (q^l + q^u))^2}{(q^u - q^l - 2q_s)^2} + \frac{\dot{q}^2}{\dot{q}_{max}^2} - 1 \leq 0 \tag{6}$$

which represents an ellipse spanning between minimum and maximum joint limits $q^l, q^u \in \mathbb{R}^K$, allowing for maximum velocity $\dot{q}_{max}$ in the middle and zero velocity in boundaries (refer to Fig.5A). Reducing velocity boundaries when approaching joint limits helps avoiding impacts and sudden stopping which is harmful for the mechanical hardware. The variable q_s is a safety margin for the joint limit. If position controllers of the real robot overshoot in certain trajectories, this variable helps avoiding reaching the limit and producing impacts. Equation (6) is element-wise, though we avoid indices for simplicity. One can define polygon-based safe regions as well, similar to [20]. The quadratic optimization problem for this stage will be:

$$\min_{\Delta q,\delta} \delta^T Q \delta + \Delta q^T R \Delta q \tag{7}$$
$$f(q^-) + \frac{\partial f(q^-)}{\partial q^-} \Delta q = x + \begin{bmatrix} \delta \\ 0 \end{bmatrix}$$
$$g(q^- + \Delta q, \Delta q/\Delta t) \leq 0$$

Next, the unknown positions q and velocities $\dot{q}$ are calculated in a similar way to (3).

D. IK_4: Direct position optimization

The formulation of our proposed IK method is similar to (4), though with inequality constraints:

$$\min_{q,\delta} \delta^T Q \delta \tag{8}$$
$$f(q) + \gamma[\frac{\partial f(q)}{\partial q}(q - q^-) - \dot{x}\Delta t] = x + \begin{bmatrix} \delta \\ 0 \end{bmatrix}$$
$$g(q, (q - q^-)/\Delta t) \leq 0$$

The regulator γ is introduced to help the joint getting out of singular positions faster, as explained in the next section. The novel formulation of IK_4 is similar to IK_2, although the optimization method is not conjugate gradient anymore and velocities result from differentiation, instead of being linked to the derivatives of the task $\dot{x}$ directly.

E. IK_5: Two-slack optimization

The flexible formulation of IK_4 allows for escaping the singularity by incorporating the knowledge of $\dot{x}$ into the optimization. A similar way is to define a new slack variable on velocities:

$$\min_{q,\delta,\varepsilon} \delta^T Q \delta + \gamma \varepsilon^T Q \varepsilon \tag{9}$$
$$S_N[\frac{\partial f(q)}{\partial q}(q - q^-) - \dot{x}\Delta t] = \varepsilon$$
$$f(q) = x + \begin{bmatrix} \delta \\ 0 \end{bmatrix}$$
$$g(q, (q - q^-)/\Delta t) \leq 0$$

which decouples velocity and position equations, resulting in slightly faster convergence shown later. The behavioral performance however remains the same as IK_4.

Note that IK_1 and IK_3 are similar in the sense that they both integrate velocities to find positions. The integration in these methods is over time, where trajectories reach the target with particular dynamics. On the other side, IK_2, IK_4 and IK_5 are similar because they perform all iterations at once to reach the desired reference trajectory faster. In IK_3, IK_4 and IK_5, we add inequality constraints to make sure the motion is feasible whereas in IK_1 and IK_2, there is not such guaranty.

III. Results

The five IK methods are evaluated in this section over different symmetric and asymmetric balancing tasks for our robot Coman [8]. This kid-size robot weighs about $30kg$ and is about $1m$ tall. Disabling the upper body of the robot, we have $K = 18$ DoF out of which six degrees are floating base variables for the pelvis of the robot. We consider six Cartesian tasks of three dimensions: orientations of the two feet and torso (three tasks), positions of the feet and position of CoM (three tasks). In this paper, we do not explore redundant systems and limit our application to a fully actuated case ($M = 18$). We also compromise the CoM position task ($N = 3$) to handle singularities better. The position and orientation of both feet are fixed in our experiments. All closed form solutions like (2) are calculated with LU factorization of Eigen library while inequality-constrained optimizations are solved with SNOPT. This package uses sequential quadratic programming and a similar factorization method. Besides, the damping factors R are already implemented in SNOPT which provide numerical stability of trajectories in singular conditions. The model of the robot is also calculated by SD-Fast and forward simulations are done in Webots. For this paper, we consider a unit-diagonal Q matrix that gives equal weight to different CoM tasks. In optimization-based methods (IK_2, IK_3, IK_4, IK_5), we iterate n times and always use the solution of the previous time-step to make the optimization faster, i.e. warm starting. In general, the optimization is continuous but non-convex, because of nonlinear constraints. Previous solutions however help to find locally optimal continuous solutions. Simulations and control of the real robot are done on a Core-i5 1.7GHz CPU in C++ language, using no balancing feedback and purely sending position commands. All parameters are listed in Table.I.

Parameter	simulations	real robot
Q	diag([1 .. 1])	diag([1 .. 1])
R	diag([1e-2 .. 1e-2])	diag([1e-2 .. 1e-2])
γ	10	10
n	15 iterations	15 iterations
$\dot{q}_{max}$	4 rad/s	2 rad/s
q_s	0 degrees	5 degrees
q^l and q^u	joint specific	joint specific

TABLE I: All parameters used in our IK formulations. Except γ and R to be tuned, the rest of these parameters are robot specific or for safety reasons. The matrix R is manually tuned to ensure precise and robust tracking of our fast motions. The parameter γ depends on the speed of motion. Larger values make our fastest task numerically unstable. For slow tasks however, this value can be increased.

A. Symmetric squatting without singularity

The first trivial task is an up-down motion of CoM (at 1Hz) without reaching limits or singularities. This scenario merely compares baseline performance of the algorithms. Although iterating $n = 15$ times is enough, we explore fewer iterations as well to investigate the effectiveness of this extra computational cost. Note that our optimization package uses SQP method, and we only limit major iterations, not those performed at each QP stage. Fig.1 demonstrates the tracking error of the five IK methods in logarithmic scale as well as the computation time. On average, IK_4 and IK_5 are better than others, even after reducing the number of iterations. IK_2 and IK_3 are more sensitive to the number of iterations, however. We can also notice that extra iterations of IK_4 and IK_5 are not improving the error considerably. For this simple yet fast scenario, IK_1 seems to have enough precision and light computation time which makes it attractive for many robotic applications. However, we will show that it is not well-suited for singular motions.

B. Symmetric squatting with singularity

One is now interested to know how these IK methods behave when going to singular postures. The results of this test are shown in a similar squatting task of a larger CoM excursion in Fig.2. In this case, the reference trajectory goes beyond limitations, though all IK algorithms can comply with it, thanks to flexible formulations. Although they all approach the singularity without vibration, they leave this posture with different dynamics. Integration based methods (IK_1 and IK_3) are slower while direct position-based methods (IK_2, IK_4 and IK_5) escape faster. The motion of IK_1 is infeasible as it goes beyond $\dot{q}_{max}$. It is expected, though, because there is no boundary imposed. Here, IK_4 and IK_5 are less sensitive to iteration numbers, and the extra iterations do not improve

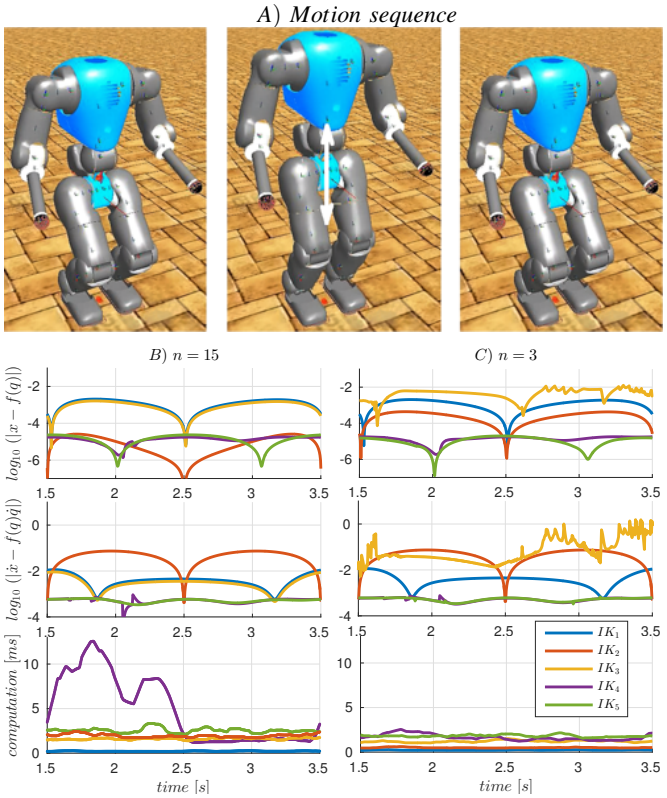

Fig. 1: Simple symmetric squatting simulations are shown in **A)** without reaching singularities. We repeat the test for $n = 15$ in **B)** and $n = 3$ iterations in **C)**. In each group, top: task error, middle: task derivative error and bottom: computation time (filtered). All IK methods provide convincing tracking down to few millimeters. IK_2, IK_4 and IK_5 perform roughly 2 orders of magnitude better, thanks to exact nonlinear equations. With fewer iterations, IK_2 and IK_3 perform worse while IK_4 slightly loses precision. This means that most of the work is done in minor iterations of the first QP step in the nonlinear optimization. The fastest algorithm is IK_1 in this simple test, but less precise.

the precision considerably. Other algorithms (IK_2 and IK_3) are very sensitive, though, often introducing large delays.

C. Asymmetric squatting with singularity

So far we only explored symmetric tasks with few joints being active. One is also interested to know how these algorithms perform in asymmetric cases where only one leg goes to singularity at a time. Here, we simulate similar squatting with singularities and add a tilt and roll motion to the torso. Fig.3 shows the resulting trajectories, where we observe that IK_1, IK_2 and IK_3 fail. IK_4 and IK_5 are both stable, though IK_5 freezes with $n = 3$ iterations. Here, IK_4 and IK_5 cost more computation times (with $n = 15$) but guarantee a feasible motion. Unlike symmetric cases, IK_1 and IK_2 produce infeasible motions here which are not desired. Even though IK_3 was able to handle symmetric singularities and satisfy safety criteria, it is not able to handle asymmetric motions due to its limited integrating nature. This test proves the capability of our proposed formulations (IK_4 and IK_5) to handle different arbitrary tasks, where IK_4 only needs few iterations which cost around $2ms$ of computation time.

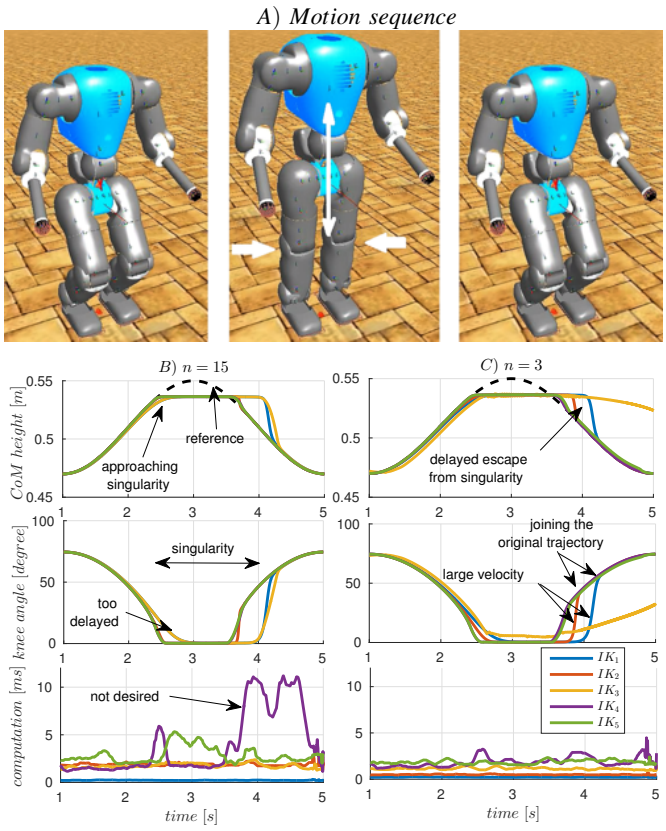

Fig. 2: Large symmetric squatting simulations are shown in **A)**, spending few moments in singularity. We repeat the test for $n = 15$ in **B)** and $n = 3$ iterations in **C)**. In each group, top: CoM trajectory, middle: knee angle and bottom: computation time (filtered). Although the reference trajectory is infeasible, except IK_3, other algorithms can comply, i.e going to singular posture without vibration and coming out. IK_1 and IK_3 have a large delay when coming out of the singular posture. They result in large velocities (often infeasible) afterwards to catch up with the actual trajectory. IK_2, IK_4 and IK_5 come out faster however. When performing fewer iterations, IK_4 and IK_5 can still do the job while IK_2 and IK_3 perform worse. IK_4 requires more iterations than IK_5 on average. Considering the version with fewer iterations however, it turns out that the extra computation is not practically useful in this case.

D. Simulating other joint limits

Along with singularities, we mentioned that joint limits are also important for IK algorithms. Here, we simulate a large tilting motion of the torso where the CoM is also required to be high. Leaning forward is fine, but when leaning backward, the hip-pitch joints reach the limit, where the IK algorithm must leave the singular position in the knee, compromise the CoM task more, and eventually respect the limit in the hip. This task is not doable with IK_1 and IK_2 due to violating boundaries and with IK_3 due to the slow integrating nature. IK_4 is, however, able to handle the task demonstrated in Fig.4. If we use IK_2, the robot ignores infeasible hip trajectories which might affect the actual CoM position and endanger the overall balance shown in the accompanying video.

E. Effect of boundary design

One might be interested in using polygon boundary models [20] instead of circular ones (6). The ultimate effect depends on the task velocity and the design of such polygon. As observed in Fig.5, polygons can generally add a delay when

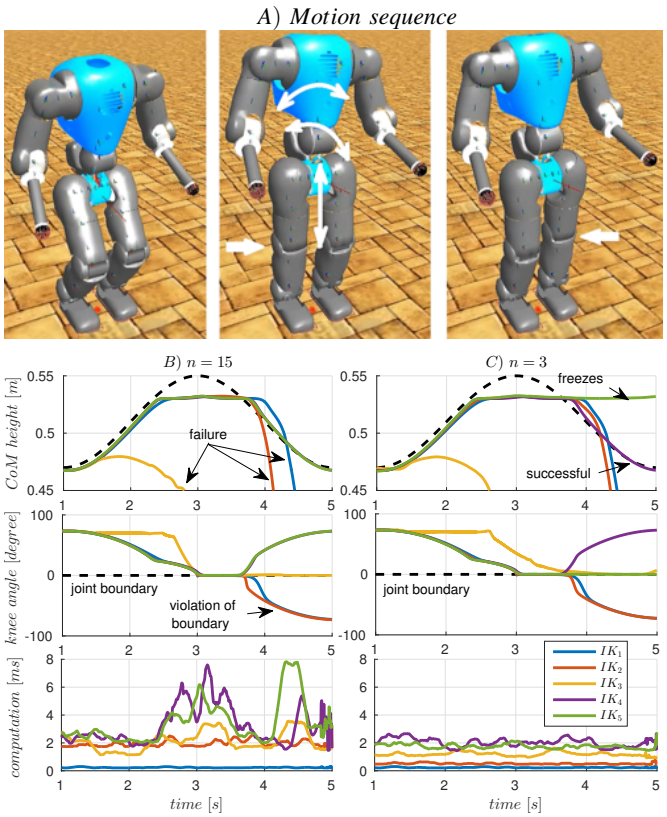

Fig. 3: Large asymmetric squatting simulations are shown in **A)**, where each leg spends few moments in singularity, but not at the same time. We repeat the test for $n = 15$ in **B)** and $n = 3$ iterations in **C)**. In each group, top: CoM trajectory, middle: knee angle and bottom: computation time (filtered). Here, only IK_5 ($n = 15$) and IK_4 can survive the motion, while other versions either get frozen or lead to falling eventually. It is also observed that IK_1 and IK_2 can cause infeasible knee angles which is not desired. IK_5 requires more iterations to perform the task while IK_4 can still survive with few iterations.

coming out of the singularity, since the derivative of velocity with respect to position is bounded close to joint limits. With a circular shape, however, the joint can quickly accelerate and escape the singularity.

F. Effect of γ

As said before, the variable $\gamma \geq 0$ mainly influences dynamics of recovering from singularities in IK_4 and IK_5. Consider Fig.6, where the real robot is performing symmetric squatting with singularity. In the case of $\gamma = 0$, although the joint can still leave the singularity (with more delay), when reaching the actual knee trajectory, it suddenly changes velocity which is not desired (demonstrated in Fig.5C too). Large accelerations cause high currents which are harmful for electronics of the robot. Here, γ helps to get out of singularity faster, by using the knowledge of $\dot{x}$ which is negative (Fig.5B). Besides, γ can smoothen the transition when joining the actual trajectory too. Although slower tasks might still be stable with larger γ, we found that $\gamma = 10$ is enough to cover all simulations mentioned in this paper. Larger values make IK_4 and IK_5 unstable in very fast motions. One can also think of limiting accelerations by adding more inequalities, but this might add unwanted oscillations [20]. Besides, such constraints do not help to get out of singularity faster, because the knowledge of $\dot{x}$ is missing.

G. How singularities improve power consumption

We tested a rather simple motion on the real robot both in a singular and multiple crouched postures to compare the power consumption. Demonstrated in Fig.7, the robot is performing a lateral motion with torso twisting. This scenario is computationally hard due to asymmetry, but IK_4 can still handle it even with $n = 3$ iterations. We have plotted the average power consumption of the robot in all scenarios as well. It

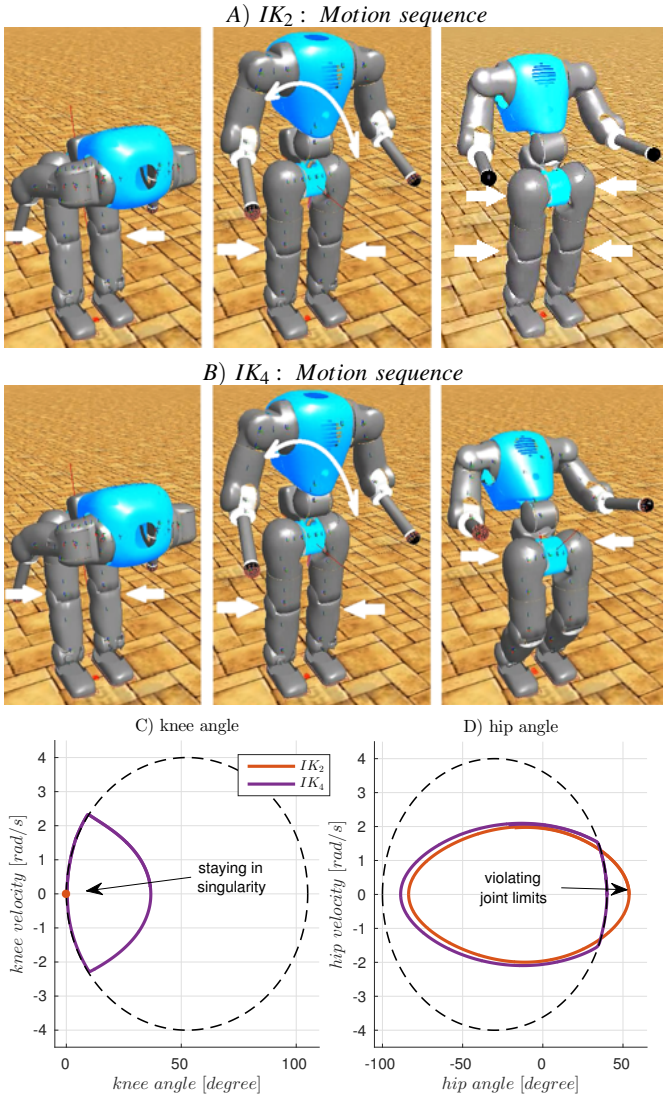

Fig. 4: When leaning backward, the robot must leave the singular posture in the knee to respect the joint limit in the hip-pitch joint. IK_2 gives infeasible hip angles **A)** and keeps the knee in a straight posture while IK_4 can easily handle the task **B)**. Trajectories of the knee and hip joints are shown in **C)** and **D)** respectively. Refer to the accompanying video for full demonstrations.

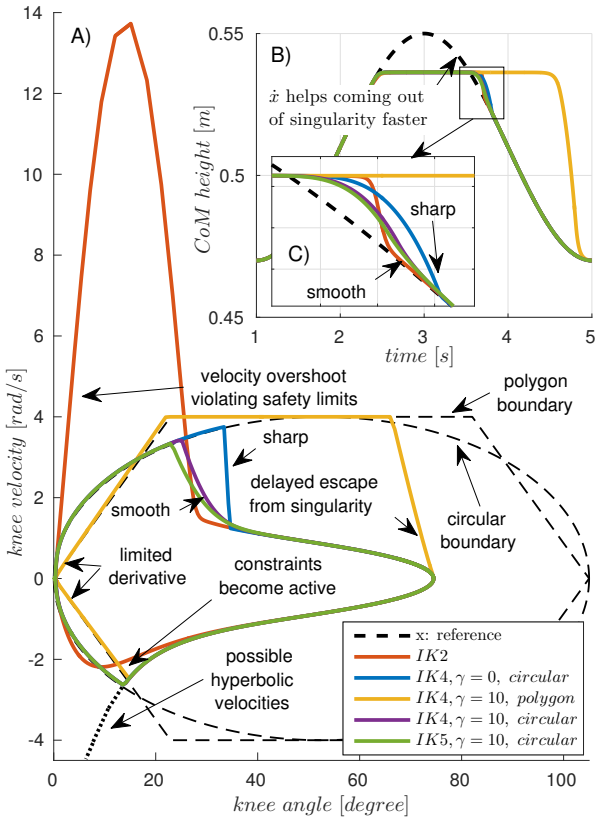

Fig. 5: The role of joint boundaries and the parameter γ in our IK algorithms. **A)** and **B)** reference and actual CoM height trajectories, **C)** knee trajectories in position-velocity space. Starting from an arbitrary point within the feasible boundaries, the knee joint approaches the singularity in 0 degrees over the defined boundary. It spends few moments in the singular position without vibrations and then comes out with certain dynamics. The boundaries limit hyperbolic velocities of the joint when approaching singularity. When coming out, however, it takes time for the joint to catch the actual trajectory again. Different boundaries might introduce certain delays, depending on the task. Here, a too safe design of polygon boundary proposed by [20] leads to a high delay. Note also that removing γ results in a delayed escape from singularity and instantaneous jump in the velocity which is not desired (refer to Figure.6 too).

is known that humanoids consume more power in crouched postures. Here we fit a line to experimental measurements to quantify the steep rise of power consumption. In the most crouched scenario which is quite similar to many humanoid robot demonstrations, Coman consumes over three times the power of stretched leg scenario (because of nearly zero knee torques in stretched-leg postures, similar to Fig.6D). This test motivates the benefit of our proposed IK method in many balancing tasks to reduce power consumption and provide more human-like postures.

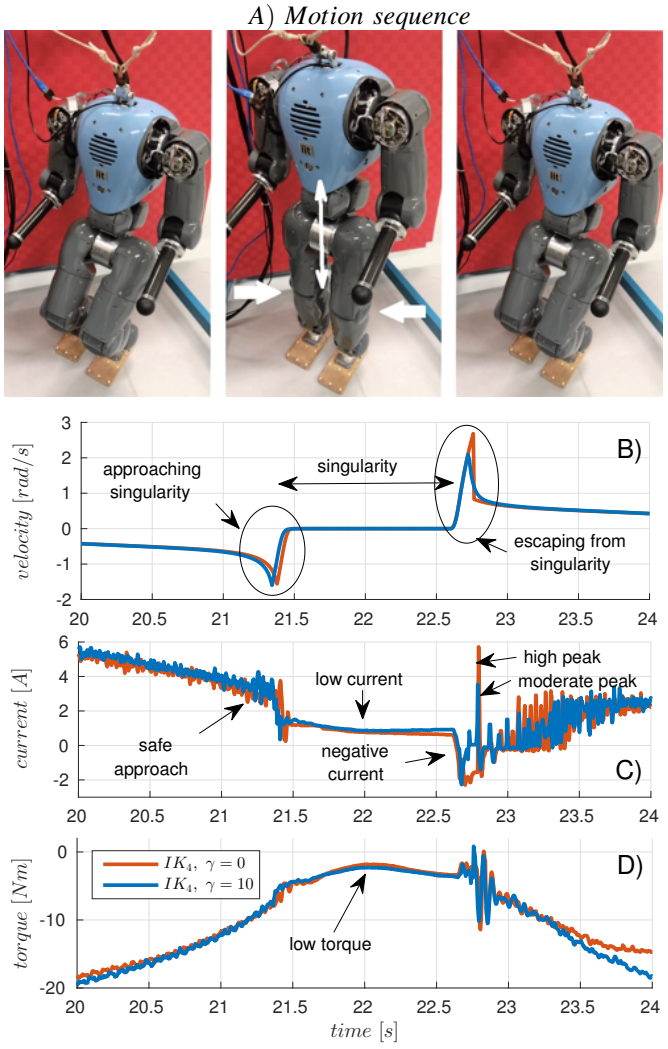

Fig. 6: The real robot in **A)** is performing large and symmetric squatting motions (like Fig.2) with IK_4 while reaching singularities. This task requires large velocities demonstrated in **B)** when approaching or coming out of singularity. One can clearly observe that the current **C)** and joint torque **D)** decrease drastically in singular position. Thanks to circular boundaries, the knee joint does not hit the limit with high current, i.e. safely approaches the joint limit and singularity. Coming out of singularity, however, the algorithm without γ results in a momentary peak in current while introducing γ can decrease this dangerous peak. Another remarkable point is that the joint produces negative current directly after leaving singularity. In these short moments, the robot transfers weight to the knee and loses potential energy.

IV. CONCLUSION

Balancing with crouched knees is very popular for humanoids as it provides full controllability properties. However, it brings many mechanical problems in the long term and requires more electrical power. Motivated by solving these practical issues, we formulated and tested traditional *IK* al-

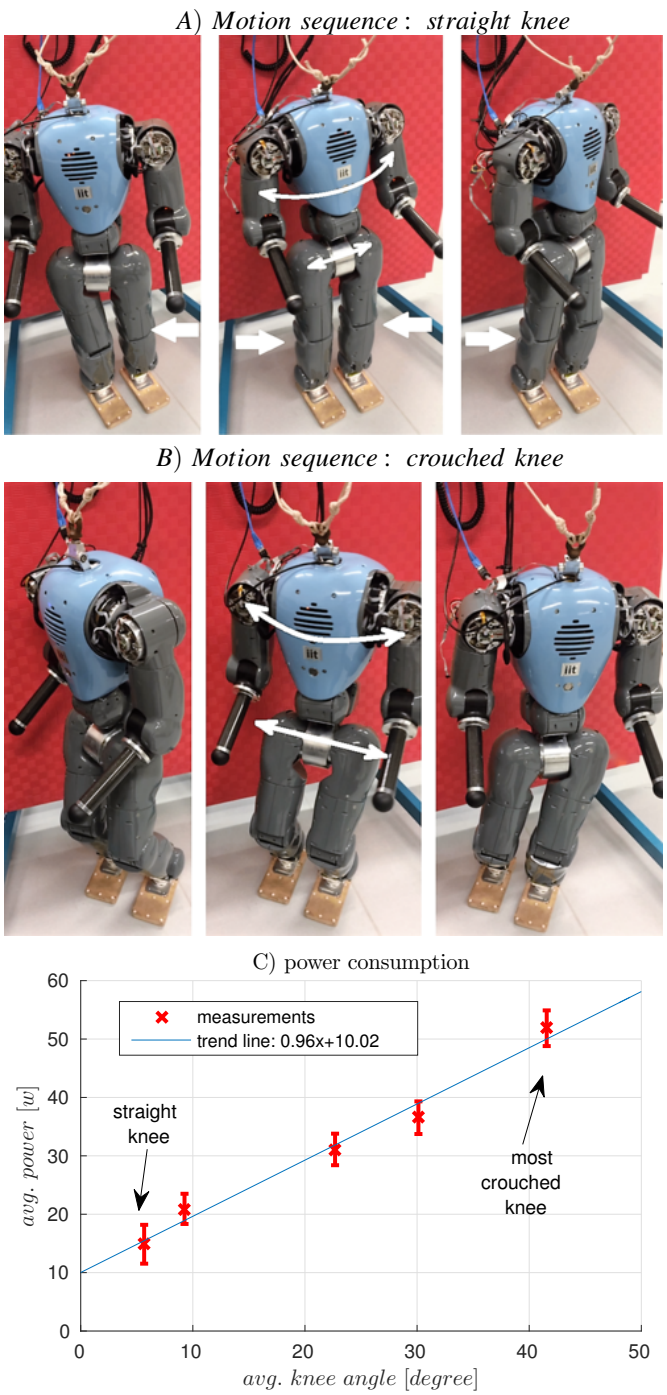

Fig. 7: Demonstration of lateral and twisting motions on the real robot with **A)** stretched and **B)** crouched knees. **C)** One can observe that crouched postures consume much more power on average. Here, the consumptions of electronics (about 108 watts) and quiet stretched-leg stance (about 10 watts) are subtracted.

gorithms, aiming at performing balance with stretched legs. Thanks to flexible damped least-squares formulations, these algorithms were numerically robust in singular positions and could handle to some extent, the safety criteria when reaching joint limits. However, due to the integrative nature, they largely fail in arbitrary symmetric and asymmetric tasks where rapid escape from singularity plays an important role in tracking. These algorithms often freeze or even lead to falling in such scenarios. Besides, in the literature, most of the straight-knee walking algorithms use preplanned trajectories which limit the generality of an *IK* block for arbitrary balancing tasks.

We formulated nonlinear optimizations instead to remove the time integration and introduced constraints to handle safety criteria. The idea of using damped least-squares, boundaries, and nonlinear optimizations is adapted from literature. The novelty of this work, however, lies in combining them together and incorporating the knowledge of task derivatives to improve the behavior, especially when coming out of singularities. This is the key feature to handle arbitrary balancing tasks where each knee might go to singularity and come out at different times (refer to the accompanying video). To the extent of our knowledge, no other *IK* method combines position and velocity variables together and mix them with inequality constraints.

All *IK* methods were compared extensively over tasks of different nature. Control-related difficulties faced with singularities are not limited to numerical stability. We showed that the two proposed *IK* algorithms have superior performance in approaching singularities and coming out of them as well. IK_5 is generally faster than IK_4, but behaves similarly. Over asymmetric tasks however, it requires same computation times and shows more sensitivity to the number of iterations. Therefore, IK_4 outperforms all other algorithms, even with a fewer number of iterations. We also showed that for our nonlinear formulation, handling the typical 18 DoF of a balancing humanoid is computationally affordable (about 2 milliseconds). We consider implementation improvements and extension to more DoF for future works.

An interesting secondary result was also presented, comparing the power consumption of stretched and crouched balancing tasks. The steep rise of power motivates application of our proposed *IK* algorithm as a low-level block that robustly transforms arbitrary Cartesian tasks to joint-space motions. Such block can improve power consumption and mechanical durability of humanoid hardware. In future, we are going to use the ideas we developed here in our *ID* formulation [8] to provide a general framework that handles geometrical and dynamical limitations while tolerating difficulties of singular postures.

V. ACKNOWLEDGMENTS

This work was funded by the WALK-MAN project (European Community's 7th Framework Programme: FP7-ICT 611832).

REFERENCES

[1] O. Khatib, "A unified approach to motion and force control of robot manipulators: The operational space formulation," *The I. J. of Robotics Research*, vol. 3, no. 1, pp. 43–53, 1987.

[2] L. Sentis, J. Park, and O. Khatib, "Compliant control of multicontact and center-of-mass behaviors in humanoid robots," *IEEE Transactions On Robotics*, vol. 26, no. 3, pp. 483–501, June 2010.

[3] F. L. Moro, N. G. Tsagarakis, and D. G. Caldwell, "A human-like walking for the compliant humanoid coman based on com trajectory reconstruction from kinematic motion primitives," in *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*. IEEE, 2011, pp. 364–370.

[4] M. Mistry, J. Buchli, and S. Schaal, "Inverse dynamics control of floating base systems using orthogonal decomposition," *IEEE International Conference on Robotics and Automation*, pp. 3406–3412, May 2010.

[5] R. Diankov, "Automated construction of robotic manipulation programs," Ph.D. dissertation, Carnegie Mellon University, Robotics Institute, August 2010. [Online]. Available: http://www.programmingvision.com/rosen_diankov_thesis.pdf

[6] A. Goldenberg, B. Benhabib, and R. Fenton, "A complete generalized solution to the inverse kinematics of robots," *IEEE Journal on Robotics and Automation*, vol. 1, no. 1, pp. 14–20, 1985.

[7] L. Righetti, J. Buchli, M. Mistry, M. Kalakrishnan, and S. Schaal, "Optimal distribution of contact forces with inverse-dynamics control," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 280–298, 2013.

[8] S. Faraji, L. Colasanto, and A. J. Ijspeert, "Practical considerations in using inverse dynamics on a humanoid robot: torque tracking, sensor fusion and cartesian control laws," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 1619–1626.

[9] N. Van der Noot, A. J. Ijspeert, and R. Ronsse, "Biped gait controller for large speed variations, combining reflexes and a central pattern generator in a neuromuscular model," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 6267–6274.

[10] Y. Ogura, K. Shimomura, H. Kondo, A. Morishima, T. Okubo, S. Momoki, H.-o. Lim, and A. Takanishi, "Human-like walking with knee stretched, heel-contact and toe-off motion by a humanoid robot," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 3976–3981.

[11] N. Handharu, J. Yoon, and G. Kim, "Gait pattern generation with knee stretch motion for biped robot using toe and heel joints," in *Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2008, pp. 265–270.

[12] R. Kurazume, S. Tanaka, M. Yamashita, T. Hasegawa, and K. Yoneda, "Straight legged walking of a biped robot," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 337–343.

[13] N. Naksuk and C. G. Lee, "Utilization of movement prioritization for whole-body humanoid robot trajectory generation," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 1079–1084.

[14] S. Chiaverini, B. Siciliano, and O. Egeland, "Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator," *IEEE Transactions on control systems technology*, vol. 2, no. 2, pp. 123–134, 1994.

[15] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," *IEEE Journal of Robotics and Automation*, vol. 17, no. 1-19, p. 16, 2004.

[16] P. Kryczka, K. Hashimoto, H. Kondo, A. Omer, H.-o. Lim, and A. Takanishi, "Stretched knee walking with novel inverse kinematics for humanoid robots," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 3221–3226.

[17] T. Sugihara, "Solvability-unconcerned inverse kinematics by the levenberg–marquardt method," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 984–991, 2011.

[18] B. Dariush, M. Gienger, B. Jian, C. Goerick, and K. Fujimura, "Whole body humanoid control from human motion descriptors," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 2677–2684.

[19] A. Escande, N. Mansard, and P.-B. Wieber, "Fast resolution of hierarchized inverse kinematics with inequality constraints," in *ICRA 2010-*

IEEE International Conference on Robotics and Automation, 2010, pp. 3733–3738.

[20] W. Suleiman, "On inverse kinematics with inequality constraints: new insights into minimum jerk trajectory generation," *Advanced Robotics*, pp. 1–9, 2016.

[21] J. Vaillant, A. Kheddar, H. Audren, F. Keith, S. Brossette, A. Escande, K. Bouyarmane, K. Kaneko, M. Morisawa, P. Gergondet *et al.*, "Multi-contact vertical ladder climbing with an hrp-2 humanoid," *Autonomous Robots*, vol. 40, no. 3, pp. 561–580, 2016.

[22] L.-C. Wang and C.-C. Chen, "A combined optimization method for solving the inverse kinematics problems of mechanical manipulators," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 4, pp. 489–499, 1991.