Model-Based Real-Time Motion Tracking using Dynamical Inverse Kinematics

Lorenzo Rapetti$^{1,2}$, Yeshasvi Tirupachuri$^{1,3}$, Kourosh Darvish$^1$, Claudia Latella$^1$, Daniele Pucci$^1$

Abstract—This paper contributes towards the development of motion tracking algorithms for time-critical applications, proposing an infrastructure for solving dynamically the inverse kinematics of human models. We present a method based on the integration of the differential kinematics, and for which the convergence is proved using Lyapunov analysis. The method is tested in an experimental scenario where the motion of a subject is tracked in static and dynamic configurations, and the inverse kinematics is solved both for human and humanoid models. The architecture is evaluated both terms of accuracy and computational load, and compared to iterative optimization algorithms.

I. INTRODUCTION

Nowadays, real-time motion tracking has many established applications in different fields such as medicine, virtual reality, and computer gaming. Moreover, in the field of robotics there is a growing interest in human motion retargeting and imitation $^{[1][2]}$. Different tracking technologies and algorithms are currently applied. Among these, optical tracking techniques are more spread and have been available since the eighties $^3$. Inertial/magnetic tracking technologies have been available only with the advent of micromachined sensors, and ensure higher frequency of data and lower latency, that makes them suited for demanding real-time applications $^4$. The objective of motion tracking algorithm is to find the human configuration from a set of measurement. Tracking algorithms can use human body representations with different level of complexity spacing from contours $^5[6]$, stick figure $^7[8]$, and volumes $^9[10]$. For some techniques it is not required to know a priori the shape of the model, and its identification is part of the algorithm $^5[7]$. When the human is modelled as a kinematic chain, the solution of the model inverse kinematics has a major role in the algorithm $^1[1][2][3][4]$. Hence, strategies to solve it efficiently are required.

Traditionally, in the field of robotics, a common inverse kinematics problem consists in finding the mapping between the end-effector of a manipulator and the corresponding joint configuration. Compared to an industrial manipulator, solving the inverse kinematics for a human kinematic model can be demanding. The human kinematic chain is redundant, it generally has an high number of degrees of freedom (DoF), and may also take into account musculoskeletal constraints in order to ensure realistic motion. Moreover, it has to be taken into account the fact that a human is a floating base system, and its configuration space lies on a differentiable manifold $^{15}$. Since finding an analytical closed-form solution for the inverse kinematics of a human model is not always either possible or efficient, a numerical solution is often preferred. One of the most common approach for solving inverse kinematics is to formulate the problem as a non-linear optimization that is solved via iterative algorithms $^{16}[17]$. This class of algorithms, referred to as instantaneous optimization, aims to converge to a stable solution for each time step. Although instantaneous optimization algorithms converge fast to a solution for common robotics applications, finding the solution for a human model at a sufficient rate for time critical application becomes demanding. In some case, better performance are achieved using heuristic iterative algorithms $^{[18]}$, learning algorithms $^{[19]}$, or combining analytical and numerical methods $^{20}$. An alternative approach consists in rephrasing the inverse kinematics problem as a control problem $^{[21]}$. This class of algorithms will be referred to as dynamical optimization since the model configuration is controlled in order to dynamically converge over time to the desired task space targets. From a computational point of view, the main advantage of this approach consists into the fact that at each time-step the solution is directly computed in a single iteration. The absence of iterations makes the dynamical optimization approach faster, and suited for solving whole-body inverse kinematics in motion tracking applications.

This article introduces a scheme for real-time motion tracking using a human, or humanoid, model defined as floating-

Fig. 1: Model of the human (left) and the iCub humanoid robot (right) in T-pose, with frame definitions of the corresponding physical links.

$^1$ Dynamic Interaction Control at Istituto Italiano di Tecnologia, Center for Robotics Technologies, Genova, Italy. (email: name.surname@iit.it)

$^2$ Machine Learning and Optimisation, The University of Manchester, Manchester, United Kingdom.

$^3$ DIBRIS, University of Genova, Genova, Italy.
base kinematic chain. The tracking is achieved at high frequency by solving the inverse kinematics using dynamical optimization. The theoretical background for applying dynamical inverse kinematics optimization to floating-base systems is presented using rotation matrix parametrization, and the convergence of the method is proved using Lyapunov theory. The implementation of the proposed scheme is tested with static and dynamical motions, and involving both human and humanoid models. The performance are compared to the results obtained using instantaneous optimization. The paper is organized as following: Section II introduces the notation, the human modeling, and the formulation of the motion tracking as inverse kinematics. Section III presents the theory for applying dynamical optimization inverse kinematics to a floating base models. Section IV lays the experimental details, and in V the results are discussed and compared to instantaneous optimization. Conclusions follow in Section VI.

II. BACKGROUND

A. Notation

- \( \mathcal{I} \) denotes an inertial frame of reference.
- \( I_{n \times n} \in \mathbb{R}^{n \times n} \) denotes the identity matrix of size \( n \).
- \( A_{pb} \in \mathbb{R}^{3} \) is the the position of the origin of the frame \( B \) with respect to the frame \( A \).
- \( A_{R} \in SO(3) \) represents the rotation matrix of the frames \( B \) with respect to \( A \).
- \( A_{w} \in \mathbb{R}^{3} \) is the angular velocity of the frame \( B \) with respect to \( A \), expressed in \( B \).
- The operator \( \text{tr}(\cdot) : \mathbb{R}^{3 \times 3} \to \mathbb{R} \) denotes the trace of a matrix, such that given \( A \in \mathbb{R}^{3 \times 3} \), it is defined as \( \text{tr}(A) := A_{1,1} + A_{2,2} + A_{3,3} \).
- The operator \( \text{sk}(\cdot) : \mathbb{R}^{3 \times 3} \to so(3) \) denotes skew-symmetric operation of a matrix, such that given \( A \in \mathbb{R}^{3 \times 3} \), it is defined as \( \text{sk}(A) := (A - A^\top)/2 \).
- The operator \( S(\cdot) : \mathbb{R}^{3} \to SO(3) \) denotes skew-symmetric vector operation, such that given two vectors \( v, u \in \mathbb{R}^{3} \), it is defined as \( v \times u = S(v)u \).
- The vee operator \( V : so(3) \to \mathbb{R}^{3} \) denotes the inverse of the skew-symmetric vector operator, such that given a matrix \( A \in so(3) \) and a vector \( u \in \mathbb{R}^{3} \), it is defined as \( Au = A' \times u \).
- The operator \( \| \cdot \|_2 \) indicates the squared norm of a vector, such that given a vector \( v \in \mathbb{R}^{3} \), it is defined as \( \|v\|_2 = \sqrt{v_1^2 + v_2^2 + v_3^2} \).

B. Modelling

The human can be modelled as a multi-body mechanical system composed of \( n \) rigid bodies, called links that are connected by \( n \) joints with one degree of freedom (DoF) each. Additionally, the system is assumed to be floating base, i.e. none of the links has an a priori constant pose with respect to the inertial frame \( \mathcal{I} \). Hence, a specific frame, attached to a link of the system, is referred to as the base frame, and denoted by \( B \).

The model configuration is characterized by the position and the orientation of the base frame along with the joint positions. Accordingly, the configuration space lies on the Lie group \( Q = \mathbb{R}^3 \times SO(3) \times \mathbb{R}^n \). An element of the configuration space \( q \in Q \) is defined as the triple \( q = (\gamma_{pb}, \gamma_{R}, s) \) where \( \gamma_{pb} \in \mathbb{R}^3 \) and \( \gamma_{R} \in SO(3) \) denote the position and the orientation of the base frame respectively, and \( s \in \mathbb{R}^n \) is the joints configuration representing the topology of the mechanical system. The position and orientation of a frame \( A \) attached to the model can be obtained via geometrical forward kinematic map \( h_{A}(\cdot) : Q \to (SO(3), \mathbb{R}^3) \) from the model configuration. The forward kinematics can be decomposed into position, i.e. \( \gamma_{pb} = h_{A}^p(q) \), and orientation, i.e. \( \gamma_{R} = h_{A}^R(q) \), maps.

The model velocity is characterized by the linear and angular velocity of the base frame along with the joint velocities. Accordingly, the configuration velocity space lies on the group \( V = \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^n \). An element of the configuration velocity space \( \nu \in V \) is defined as \( \nu = (\nu_{pb}, \nu_{R}, \nu_{\omega}) \) where \( \nu_{pb} = (\dot{\gamma}_{pb}, \nu_{\omega}) \in \mathbb{R}^3 \) denotes the linear and angular velocity of the base frame, and \( \nu_{R} \) denotes the joint velocities.

The velocity of a frame \( A \) attached to the model is denoted by \( \nu_{A} = (\dot{\gamma}_{A}, \dot{\nu}_{A}) \) with the linear and the angular velocity components respectively. The mapping between frame velocity \( \nu_{A} \) and configuration velocity \( \nu \) is achieved through the Jacobian \( J_{A} = J_{A}(q) \) and the angular part \( J_{A}^{\omega}(q) \) that maps respectively the linear and the angular velocities, i.e. \( \nu_{A} = J_{A}(q)\nu \) and \( \nu_{\omega} = J_{A}^{\omega}(q)\nu \).

C. Problem Statement

Motion tracking algorithms aims to find the human configuration given a set of targets describing the kinematics of its links. Those targets are the measurements of the link pose and velocity expressed in a world reference frame, and are retrieved from sensors or image processing. The process of estimating the configuration of a mechanical system from task space measures is generally referred to as inverse kinematics, and can be formulated as follows.

**Problem 1.** Given a set of \( n_{p} \) frames \( \mathcal{P} = \{P_1, P_2, \ldots, P_{n_p}\} \) with the associated target position \( \gamma_{pb}(t) \in \mathbb{R}^3 \) and target linear velocity measurements \( \nu_{pb}(t) \in \mathbb{R}^3 \), and given a set of \( n_{o} \) frames \( \mathcal{O} = \{O_1, O_2, \ldots, O_{n_o}\} \) with the associated target orientation \( \nu_{R}(t) \in SO(3) \) and target angular velocity measurements \( \nu_{\omega}(t) \in \mathbb{R}^{3} \), find the state configuration \( (q(t), \nu(t)) \) of a model such that:

\[
\begin{align*}
\gamma_{pb}(t) &= h_{P_i}^p(q(t)), & \forall i = 1, \ldots, n_p \\
\gamma_{R}(t) &= h_{O_i}^o(q(t)), & \forall j = 1, \ldots, n_o \\
\nu_{pb}(t) &= J_{P_i}^p(q(t))\nu(t), & \forall i = 1, \ldots, n_p \\
\nu_{\omega}(t) &= J_{O_i}^{\omega}(q(t))\nu(t), & \forall j = 1, \ldots, n_o \\
A_{s}(t) &\leq b, \\
C_{s}(t) &\leq d,
\end{align*}
\]

where \( A \) and \( b \) are two parameters that represent the limits for the joint configuration of the model, and \( C \) and \( d \) represents the limits for the joint velocity.
The following quantities are defined in order to have a compact representation of Problem 1. The targets are collected in a pose target vector $x(t)$ and velocity target vector $v(t)$:

$$
x(t) := \begin{bmatrix}
I_p q(t) \\
I_p n_p q(t) \\
I R_O (t) \\
I R_{O_n} (t)
\end{bmatrix}, \quad v(t) := \begin{bmatrix}
I p q(t) \\
I p n_p q(t) \\
I R_O (t) \\
I R_{O_n} (t)
\end{bmatrix},
$$

and, forward geometrical kinematic can be expressed as a single vector $h(q(t))$, and Jacobians as single matrix $J(q(t))$:

$$
h(q(t)) := \begin{bmatrix}
h_p(q(t)) \\
h_{p_n}(q(t)) \\
h_{O}(q(t)) \\
h_{O_n}(q(t))
\end{bmatrix}, \quad J(q(t)) := \begin{bmatrix}
J_p(q(t)) \\
J_{p_n}(q(t)) \\
J_{O}(q(t)) \\
J_{O_n}(q(t))
\end{bmatrix},
$$

the set of equations in (1) can then be reduced, using the definitions of (2) and (3), to the following two equations describing respectively the forward kinematics and differential kinematics for all the target frames:

$$
x(t) = h(q(t)), \quad v(t) = J(q(t)) \nu(t).
$$

As mentioned in Section 1 in the case of highly articulated systems, like humans, finding an analytical solution to Equation (4) is often vain, and a numerical optimization solution is preferred. Hence, distance measurements used in the optimization problem have to be defined. The distances between a given state $q(t)$ and the pose targets $x(t)$ are collected in a residual vector $r(q(t), x(t))$ defined as follow:

$$
r(q(t), x(t)) = \begin{bmatrix}
I p q(t) - h_p(q(t)) \\
I p n_p q(t) - h_{p_n}(q(t)) \\
\text{sk}(h_{O}(q(t))^{\top} I R_O(t))^{\top} \\
\text{sk}(h_{O_n}(q(t))^{\top} I R_{O_n}(t))^{\top}
\end{bmatrix}.
$$

Traditionally, Euler angles or unit quaternion parametrization are used in literature while defining the distance for link orientations. However, using rotation matrix parametrization, we use the $\text{sk}(\cdot)^{\top}$ operator for defining the distance. The motivation behind the choice of this operator is in the prove of convergence presented in Appendix. The distances from the target velocities $v(t)$ are collected in the velocity residual vector $u(q(t), v(t))$ defined as:

$$
u(q(t), v(t)) = v(t) - J(q(t)) \nu(t).
$$

III. DYNAMICAL OPTIMIZATION METHOD

The dynamical optimization method aims to minimize the residual vectors, whose dynamics is described by the differential equations (5) and (6), by controlling the state configuration $(q(t), v(t))$. The block diagram of the method is presented in Figure 2. The scheme is composed by three main parts: a) correction of velocity targets to ensure convergence of pose targets, b) inversion of the model differential kinematics to obtain the state velocity $v(t)$, and c) integration of state velocity to obtain the configuration $q(t)$.

A. Velocity Correction

At this stage, we assume the state velocity $v(t)$ is the input to a dynamical system, and we want to control the system in order to drive the residual vectors towards zero.

**Lemma 1.** Assume $r(q(t), x(t))$ defined as in (5), $u(q(t), v(t), v(t))$ defined as in (6), and the system

$$
u(q(t), v(t), v(t)) + Kr(q(t), x(t)) = 0, \quad (7)
$$

where $K \in \mathbb{R}^{(3n_p+3n_o) \times (3n_p+3n_o)}$ is a positive definite diagonal matrix. Then, $(r, u) = (0, 0)$ denotes an (almost) globally asymptotically stable equilibrium point for the system.

The proof is provided in appendix. Lemma 1 shows that we can control the system input $v(t)$ so that $r(t)$ and $u(t)$ converge to zero for (almost) any initialization $q(t_0)$. Replacing the expression of $u(q(t), v(t))$, presented in (6), in the system (7), it can be observed that the expression is linear in state velocity $v(t)$.

$$
u(q(t)) \nu(t) = v(t) + Kr(q(t), x(t)).
$$

The rate of convergence depends on the magnitude of the elements of matrix $K$. Higher values of $K$ imply faster convergence of the system (7) towards zero. However, the implementation of discrete time solution bounds the values of $K$ depending on the sampling time $\Delta t$. In fact, Equation (8) is solved for a discrete control input $\nu(t_k)$ from the following equation:

$$
u(q(t_k-1)) \nu(t_k) = v(t_k) + Kr(q(t_k), x(t_k)).
$$

The way the discrete control input $\nu(t_k)$ is obtained will be discussed in III-B. It is interesting to notice that the Equation (8) establishing the control input law, is equivalent to a differential kinematic equation (45) where the target velocity vector $v(t)$ is replaced by a corrected target velocity vector $v^*(t) := v(t) + Kr(q(t), x(t))$.

B. Inverse Differential Kinematics

The inverse differential kinematics is the problem of inverting the differential kinematic Equation (45) in order to find the configuration state velocity $v^*(t)$ for a given set of task space velocities. Computing control input $\nu(t_k)$ from (9) is equivalent to solving the inverse differential kinematics for the corrected target velocity $v^*(t_k)$. The solution depends on the rank of the Jacobian matrix $J(q(t_k))$, and in most of the
cases it is found numerically. Being it a common problem
to most of the inverse kinematics methods [16][17][21][24],
different strategies can be found in literature. Among the
possible solutions, a common approach is to use Jacobian
generalized inverse [25]. In order to take into account also
the model constraints, an alternative approach is to solve it
via QP optimization:

\[
\begin{align}
\text{minimize} & \quad \|v^*(t_k) - J(q(t_k-1))v(t_k)\|_2 \quad \text{(10a)} \\
\text{subject to} & \quad Gv(t_k) \leq g \quad \text{(10b)}
\end{align}
\]

where \( G \) and \( g \) are two parameters used for limits avoidance
that depend on \( q(t_k-1) \) and the model constraints parameters
\( A, b, C \) and \( d \).

C. Numerical Integration

Given the configuration velocity solution \( v(t_k) \), it is possible
to compute the state configuration \( q(t_k) \) by defining an initial
configuration \( q(t_0) \) and integrating over time. Base position
\( I_{PB} \) and joints configuration \( s \) lie in vector space over \( \mathbb{R} \)
for which most of the numerical integrations methods proposed in
literature can be used [26]. The integration of the base angular
velocity \( \dot{\omega}_B(t_k) \) is not trivial, numerical integration errors can
lead to the violation of the orthonormality condition [27] for
the base orientation \( I_{RB}(t_k) \) over SO(3). A possible approach presented
in literature is the Baumgarte stabilization [27], the convergence
of \( I_{RB}(t_k) \) over SO(3) is ensured computing the base orientation
matrix dynamics \( I_{RB}(t_k) \in \mathbb{R}^{3\times3} \) as follow:

\[
\begin{align}
A(t_k-1) = \rho(\dot{I}_{RB}(t_k-1) + I_{3\times3}), \quad \text{(11a)} \\
I_{RB}(t_k) = (S(\dot{\omega}_B(t_k)) + A(t_k-1)) I_{RB}(t_k-1), \quad \text{(11b)}
\end{align}
\]

where \( \rho \in \mathbb{R}^+ \) is the gain regulating the convergence towards
the orthonormality condition, and \( \Delta t_k = t_k - t_{k-1} \) is the
time step. The advantage of obtaining the configuration \( q(t_k) \)
through integration of velocity \( v(t_k) \) is that the two estimated
quantities are directly related, and continuity of the state
configuration is ensured.

![Fig. 2: Dynamical optimization algorithm for real-time inverse kinematics of floating base model.](image)

IV. EXPERIMENTS

A. Motion Data Acquisition

The proposed method has been implemented and tested
using motion data acquired with the Xsens Awinda wearable
suit [28] providing pose and velocity of a 23 links human
model, computed from a set of distributed Inertial Measurement
Units (IMUs). The motion data is streamed through
YARP middleware [29] that facilitates recording and real-
time playback of data. The motion data is acquired for three
scenarios with different levels of dynamicity: \( t \)-pose where the
subject stands on two feet with the arms parallel to the ground,
walking where the subject walks on a treadmill at a constant
speed of 4 \( \text{km/h} \), and running where the subject runs on a
treadmill at a constant speed of 10 \( \text{km/h} \).

B. Models

The motion tracking is performed by using two different
human models defined as in [11][B] Both the models are com-
pose by 23 physical links representing segments of the human
body. Each physical link is attached to the next one through a
certain number of rotational joint connected trough dummy
links, i.e. links with dimension zero, in order to model human
joints with multiple DoFs. In one human model (Human66)
all the physical links are connected through spherical joints
(3 rotational joints), i.e. a total of 66 DoFs and 67 links.
The second model (Human48) is based on the modelling of the
human musculoskeletal system as described in clinical
studies [30][31][32], it has a reduced number of joint, i.e. 48DoFs,
and takes into account human joint limits.

Additionally, we consider experiments with a model of the
iCub humanoid robot [33]. The motivation behind this is to
highlight the performance in achieving motion tracking, and
motion retargeting from the human to a humanoid. The iCub
model is composed of 15 physical links connected through 34
rotational joints, with joint limits defined accordingly to the
real robot mechanic constraints.

The definition of link frames for the human model and the
robot model is highlighted in Figure [1] Considering that all
the models have only rotational joints, the inverse kinematics
problem is defined with a rotational and angular velocity target
for each physical link, i.e. \( n_o = 23 \) for the human model and
\( n_o = 15 \) for the humanoid model. Additionally, as both the
models are floating base, a position and linear velocity target
is used for the base frame, i.e. \( n_p = 1 \).

V. RESULTS

The performance of dynamical optimization inverse kinematics
solver is compared to instantaneous optimization imple-
mentations. The evaluation is done in terms of computational
load and accuracy on a 2.3GHz Intel Core i7 processor
with 16GB of RAM. The accuracy is measured with two
metrics, one for the orientation targets and one for the velocity
targets. The mean normalized trace error (MNT E) is a
dimensionless metric measuring the overall accuracy of the
orientation targets:

\[
MNT E = \frac{1}{n_o} \sum_{j=1}^{n_o} \frac{\text{tr}(I_{3\times3} - \dot{I}_{RO_j}(q))}{2}, \quad \text{(12)}
\]

where \( \dot{I}_{RO_j}(q) \) is the frame orientation given the state \( q \), and
the \( \frac{1}{2} \) factor normalize the value of the trace between 0 and
1. For the angular velocities, the overall error is evaluated as root mean squared error (RMSE):

$$RMSE = \sqrt{\frac{1}{n_o} \sum_{j=1}^{n_o} \left\| \hat{\omega}_{\mathcal{O}_j} - \mathcal{C}^*_{\omega_{\mathcal{O}_j}}(q, \nu) \right\|^2_2},$$

(13)

where $\hat{\omega}_{\mathcal{O}_j}(q, \nu)$ is the frame velocity given the configuration $(q, \nu)$. The computational load is evaluated as the time for computing the state $(q, \nu)$. The statistics have been collected discarding the transient of 2 second from the initial time $t_0$, and the results are shown in Figure 4.

A. Instantaneous Optimization

As mentioned in Section I, the instantaneous optimization methods solve the inverse kinematics at each time-step $t_k$ through non-linear optimization. A general formulation of the optimization problem is defined as follows:

$$\text{minimize} \quad \| K_r r(q(t_k), x(t_k)) \|_2$$

subject to \quad $As(t_k) \leq b$

(14a)

(14b)

where $K_r$ is a weight matrix that matches the unit measurements of targets distances, and eventually assigns a weight to each of the target. A common approach for solving the non-linear optimization problem is to consider the linear approximation of the system by recalling the Jacobian matrix definition, and solve the problem iteratively [34] [35]. However, in order to enforce state configuration constraints, recent approaches make also use of convex optimization [24] [36].

As benchmark, we have implemented instantaneous inverse kinematics optimization using iDynTree [37] multibody kinematics library, and the IPOPT software library for non-linear optimization [38]. The stopping criteria for the optimization is the pose error accuracy, and it has been tuned in order to find a solution in a time comparable to the dynamical optimization. Two different implementations have been tested. The first one, referred to as whole-body optimization, solves a single optimization problem instantiated for the whole-system. Instead, the second implementation instantiate the optimization process dividing the model into multiple subsystems, each consisting of exactly a pair of targets, and solves the subproblems in parallel. We refer to this implementation as pair-wise optimization.

The solution for the state velocity $\nu(t_k)$ comes from an optimization problem formulated as follow:

$$\text{minimize} \quad \| K_u u(q(t_k), \nu(t_k)) \|_2$$

subject to \quad $Cs(t_k) \leq d$

(15a)

(15b)

where $K_u$ is a weight matrix that matches the unit measurements of targets distances, and eventually assigns a weight to each of the target. In case of state velocity, it can be observed that there is a linear relationship with the velocity target vector as expressed by the differential kinematics Equation (4b). Hence, given $q(t_k)$ obtained as output of the optimization problem (14), the state velocity is the result of the inverse differential kinematics described in Section III-B and is obtained using OSQP library [39].

Looking at Figure 4 it can be observed that the performance of instantaneous optimization approaches decrease as the task gets more dynamic. This is particularly evident for the computational time. In the whole-body optimization, not only the average computational time increase, it is characterized by a large variance, reaching peaks above $25ms$ during the running task. Concerning the pair-wised optimization, the increase of time between walking and running is less evident. However, the pair-wised optimization takes longer for finding a solution for the iCub model because of the local difference between the human and the robot kinematics.

B. Dynamical Optimization

The dynamical optimization inverse kinematics has been implemented using iDynTree [37] multibody kinematics library, and the inverse differential kinematics is solved making use of OSQP [39]. Figure 3 highlights the rate of convergence of the error from a given initial zero configuration $(s(0) = 0)$ towards a target static pose. When the gain is zero, there is no velocity correction and the error remains constant. However, increasing the magnitude of $K$, the error converges to its steady state value in less then one second. A large value of the gain $K$ leads to system instability as mentioned in III-A.

From Figure 4, the dynamical optimization orientation error is always comparable with the results achieved with instantaneous optimization. The only case in which it shows worst orientation accuracy is with the Human66 model during running task. Concerning the angular velocity error, the performances are again comparable during dynamic motion, while it is higher for the t-pose with the constrained models. This angular velocity error may be due to the fact that a corrected angular velocity is used in place of the measured link angular velocities, and the joint constraints may introduce a constant constraint error because of an unfeasible configuration. Concerning the computational load, this method seems to outperform the others not only in terms of a mean computational time, having an average always below $3ms$, but also for its consistency in different scenarios.

Fig. 3: Convergence of the dynamical inverse kinematics optimization method for a static T-pose on a 66 DoF model, starting from a zero configuration. The convergence depends on the magnitude of the gain $K$. 

![Fig. 3: Convergence of the dynamical inverse kinematics optimization method for a static T-pose on a 66 DoF model, starting from a zero configuration. The convergence depends on the magnitude of the gain $K$.](image-url)
VI. Conclusions

This paper proposes an infrastructure for whole-body inverse kinematics of floating-base articulated models in real-time motion tracking applications. The theory is presented using rotation matrix parametrization of orientations, together with the proof of convergence through Lyapunov analysis. The proposed method has been implemented and the performances tested in an experimental scenario with different conditions. Differently from iterative algorithms, the dynamical optimization requires a single iteration at each time step keeping the computational time constant, and ensures fast convergence of the error over time. Furthermore, the integration of velocities ensures obtaining a continuous and smooth solution. Its characteristics make it suitable for time-critical motion tracking applications with highly dynamic motions, where iterative algorithms may not converge in a sufficient time. As a future work, the evaluation may be extended to a wider number of algorithms, models, and experimental scenarios. Another interesting future work may be the extension of the method to include the dynamic of the system.

Acknowledgments

This work is supported by Honda R&D Co., Ltd and by EU An.Dy Project that received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No. 731540. The content of this publication is the sole responsibility of the authors. The European Commission or its services cannot be held responsible for any use that may be made of the information it contains.

APPENDIX: Proof of Lemma 1.

The system in (7) can be written as follow:

\[
\begin{bmatrix}
    u_1^p(t) \\
    \vdots \\
    u_{P_p}(t) \\
    u_{O,1}(t) \\
    \vdots \\
    u_{O,n_o}(t)
\end{bmatrix} +
\begin{bmatrix}
    K_1^p r_1^p(t) \\
    \vdots \\
    K_n_p r_{n_p}(t) \\
    K_1^p r_1^o(t) \\
    \vdots \\
    K_n_o r_{n_o}(t)
\end{bmatrix} = 0,
\]

(16)

where \( r_p^p(t) = \mathcal{I}_{p_p}(t) - h_p^p(q(t)) \), \( r_1^o(t) = sk(h_1^o(q(t)))^T \mathcal{I}_{O}(t) \), \( u_{O,1}(t) = \mathcal{I}_{O,1}(t) - J_{O,1}(q(t)) \), \( u_{O,n_o}(t) = \mathcal{I}_{O,n_o}(t) - J_{O,n_o}(q(t)) \), \( K_1^p \) and \( K_o^o \) are \( R^3 \times R^3 \) blocks on the diagonal of \( K \). This system can be decomposed into a set of \( n_p + n_o \) independent systems, one for each target, depending on the type of target, each subsystem is described by one of the following two equations:

\[
\begin{align*}
    u_1^p(t) + K_1^p r_1^p(t) &= 0, \quad (17a) \\
    u_1^o(t) + K_1^o r_1^o(t) &= 0. \quad (17b)
\end{align*}
\]

The system (17a) is a linear first order autonomous system, and for \( K_i \) positive definite the equilibrium point \( (r_p^p,u_p^p) = (0,0) \) is globally asymptotically stable. For the system (17b) it can be proved that the equilibrium \( (r_0^o,u_0^o) = (0,0) \) is an almost globally asymptotically stable equilibrium point [49]. The almost global asymptotically stability of all the sub-systems is indeed proved for the point \( (r,u) = (0,0) \), thus the almost globally asymptotically stability of the equilibrium \( (r,u) = (0,0) \) for the system (7) is proved.
