Walking of the iCub humanoid robot in different scenarios: implementation and performance analysis

Yue Hu\(^1\), Jorhabib Eljaik \(^2\), Kevin Stein\(^1\), Francesco Nori\(^2\), Katja Mombaur\(^1\)

Abstract—The humanoid robot iCub is a research platform of the Fondazione Istituto Italiano di Tecnologia (IIT), spread among different institutes around the world. In the most recent version of iCub, the robot is equipped with stronger legs and bigger feet, allowing it to perform balancing and walking motions that were not possible with the first generations. Despite the new legs hardware, walking has been rarely performed on the iCub robot. In this work the objective is to implement walking motions on the robot, from which we want to analyze its walking capabilities. We developed software modules based on extensions of classic techniques such as the ZMP based pattern generator and position control to identify which are the characteristics as well as limitations of the robot against different walking tasks in order to give the users a reference of the performance of the robot. Most of the experiments have been performed with HeiCub, a reduced version of iCub without arms and head.

I. INTRODUCTION

Being humanoid robots still at a research level, there are only few platforms that are widely spread and studied across research institutes. The Fondazione Istituto Italiano di Tecnologia (IIT) built the robot iCub \(^1\) and it is now available for research purposes in several institutes all over the world in different versions.

The iCub is a medium size child-like humanoid robot with 53 degrees of freedom (DOF), of which 38 DOF in the upper body including 18 DOF of the hands, and 15 DOF in the torso and legs. It has a height of circa 1.1m and weight of 33kg. It is a very complex system designed with the aim of performing cognitive science studies \(^1\). Among the current abilities of iCub there are objects recognition, voice commands recognition, grasping, tactile sensing, and many others that are growing thanks to the several projects in which it is involved and the open source quality of the project.

In the first versions, iCub had very small feet as it was not designed for walking. In recent versions of iCub, bigger feet were introduced, allowing the robot to perform whole body balancing tasks in double and single support \(^2\). In recent

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007 - 2013) under grant agreement n. 611909 (KoroiBot), www.korobot.eu

\(^1\)Interdisciplinary Center for Scientific Computing, Heidelberg University, 69115 Heidelberg, Germany
\(^2\)Department of Robotics, Brain and Cognitive Sciences, Fondazione Istituto Italiano di Tecnologia (IIT), 16163 Genoa, Italy

yue.hu@iwr.uni-heidelberg.de
jorhabib.eljaik@iit.it
kevin.stein@iwr.uni-heidelberg.de
francesco.nori@iit.it
katja.mombaur@iwr.uni-heidelberg.de

Fig. 1: iCub and HeiCub robots, designed and built by iCub Facility department, IIT

hardware upgrades the mechanical design of the legs were derived from the CoMAN humanoid robot of IIT \(^3\). The new legs have Series Elastic Actuators (SEA) with removable springs in the knee and ankle pitch joints \(^4\).

In the framework of the European Project KoroiBot \(^5\), a customized version of the iCub was delivered to Heidelberg University (hereafter HeiCub) in order to perform walking experiments. The HeiCub robot is an iCub without head and arms and with only 15 DOF, namely 3 in the torso and 6 in each leg. It has the standard iCub hardware \(^1\), including two fixed frontal cameras mounted in the torso and the SEAs with removable springs. In the scope of this work the springs are removed to use rigid actuators only.

The objective of the KoroiBot Project is the improvement of existing humanoid robots walking abilities. In the project we first analyze the current walking skills of the available platforms to have a comparison between the actual performances and those that will come with the novel methods provided by the project.

On some of the robots of the project, however, walking has never or rarely been implemented, either in simulation or on the actual platform. This is the case of iCub, on which there have been only few experiments on walking on level ground, despite being such a diffused robot in research facilities as well as projects.

With this work we want to bring walking motions on the iCub robot in different scenarios for the first time. To this end we developed software modules to implement walking motions on the robot in order to identify suitable environmental constraints within its mechanical limits and perform quantitative measurements, e.g. tracking precision and current consumption. The objective is to give the iCub community as well as the humanoid research community a point of reference for the possibilities and limitations of the
iCub in walking using a simple software module to generate walking motions.

In order to achieve our goal, we implemented an offline ZMP-based pattern generator, which is an extension of the work in [6], that generates feet and center of mass (CoM) trajectories, which are then given as inputs to a module that performs inverse kinematics to retrieve joint trajectories. The obtained trajectories are executed on the robot using position control.

The approach has been used to carry out walking in different scenarios, including level ground, slopes and stairs. In every scenario different patterns have been tested to reach the limits of the robots within the proposed methods, where we also analyze the reason of the unsuccessful trials. The experiments were performed mainly on the HeiCub robot, but in the flat ground case also on complete iCub(s) at IIT.

II. SOFTWARE MODULES

The software module consists of three parts: the pattern generator, the inverse kinematics and the interface to the robot. The pattern generator takes as input user defined foothold positions for right and left feet and a predefined center of mass height variation pattern that is adapted to different environments, from which it generates the feet trajectories and the 3D center of mass trajectory. These trajectories are then used by inverse kinematics as desired end effector positions to compute the joint trajectories. The center of mass is assumed as a point attached to the chest and joint positions are updated with the real center of mass at every time instant, instead of assuming it being a fixed point attached to a link (usually the pelvis) as many applications do.

The reason for which we introduced the CoM height variation is to obtain motions that are closer to human-like walking, where it is known that the CoM height has a wave shape variation. This variation allows the robot to walk with stretched knee and perform bigger steps despite restricted joint limits as we will discuss later in section III.

In the following sections we review the theory behind the first two parts, explaining also the extensions introduced to fulfill our goals. Then the full workflow is described.

A. Pattern generator

One of the most common methods used in humanoid robotics to generate walking motions is the inverted pendulum or table cart combined with the ZMP principle [7]. Here we make an extension on the one proposed by Kajita et al. in [6], which allows arbitrary foot placement and assumes fixed center of mass height over the whole walking. In our extension we allow the center of mass height to vary according to a predefined pattern.

The pattern generator is based on 3D linear inverted pendulum and ZMP as in [6] and [8], where the ZMP equations including CoM height variation are described as:

\[ p = x - \frac{z}{2} \cdot \frac{\ddot{z}}{z + g} \]  \hspace{1cm} (1)

Where \( p = [p_x, p_y] \) represents the ZMP coordinates, \( x = [x, y]^T \) the CoM coordinates, \( z \) the height variation of the CoM and \( g \) the gravity.

In [6] a constrained plane \( z = k_x x + k_y y + z_i \) is assumed. By imposing \( k_x = k_y = 0 \), the equation is simplified into:

\[ p = x - \frac{z}{g} \]  \hspace{1cm} (2)

However in this work we use the formulation as in equation (1), as we want to set a predefined variable CoM trajectory in the \( z \) direction instead of keeping the CoM height constant.

The CoM on the walking plane (\( x \) and \( y \) directions) is computed from equation (1) as described in [9], with modifications introduced to take into account the \( z \) variations. The accelerations \( \dddot{x} \) is approximated as:

\[ \dddot{x}_i = \frac{x_{i-1} + 2x_i + x_{i+1}}{\Delta t^2} \]  \hspace{1cm} (3)

where \( i = 1...N \) is the number of the sample with \( N \) being the total number of samples. The same approximation is done for \( \dddot{y} \).

The ZMP equation as in Eq. (1) can be discretized as follows to have the formulation \( p = Ax \) (the same is applied for the \( y \) coordinate):

\[
\begin{bmatrix}
   p_1 \\
   p_2 \\
   \vdots \\
   p_{N-1} \\
   p_N
\end{bmatrix}
= \begin{bmatrix}
   a_1 + b_1 & c_1 & 0 & \cdots \\
   a_2 & b_2 & c_2 & \cdots \\
   \vdots & \vdots & \vdots & \cdots \\
   a_{N-1} & b_{N-1} & c_{N-1} & \cdots \\
   0 & a_N & b_N + c_N & \cdots
\end{bmatrix}
\begin{bmatrix}
   x_1 \\
   x_2 \\
   \vdots \\
   x_{N-1} \\
   x_N
\end{bmatrix}
\]  \hspace{1cm} (4)

Where the terms of the matrix are as follows:

\[
\begin{align*}
   a_i &= -\frac{z}{(\dddot{z} + g)\Delta t^2} \\
   b_i &= \frac{z}{(\dddot{z} + g)\Delta t^2 + 1} \\
   c_i &= a_i
\end{align*}
\]  \hspace{1cm} (5)

The system is then solved for the CoM coordinates \( x \). The matrix \( A \) has dimension \( N \times N \). This means that the higher is the number of samples, the bigger is the matrix \( A \). However this computation is performed offline and does not represent a concern at the moment.

B. Inverse kinematics (IK)

In order to perform walking in different environments, we need inverse kinematics that gives feasible joint angles for given multiple end effectors positions and orientations. Therefore we extended and modified the existing algorithm implemented in the Rigid Body Dynamics Library (RBDL) [10], which did not allow orientation constraints. We adopt the residuals definition as of Sugihara in [11]:

\[
e_i(q) = \frac{d^4 \mathbf{p}_i - \mathbf{p}_i(q)}{R_i^T(q) \cdot a(\cdot)}
\]  \hspace{1cm} (6)

Where \( d^4 \mathbf{p}_i \) and \( \mathbf{p}_i \) are the desired and current positions of the end effectors \( i \) and \( a(\cdot) \) is a function that computes
angular velocities from the current rotation matrix \( \mathbf{R}_i \) and the desired orientation expressed as rotation matrix \( \mathbf{R}_{d} \).

The openly available software for solving non-linear optimization problems IPOPT [12] has been used to solve the inverse kinematics problem, which is formulated as follows:

\[
\min _{\mathbf{q}} \| \mathbf{e}(\mathbf{q}) \|_2^2
\]

subject to:

\[
\mathbf{q} \geq \mathbf{q}_{\text{min}}
\]

\[
\mathbf{q} \leq \mathbf{q}_{\text{max}}
\]

(7)

Where \( \mathbf{e} \) is the residuals of stacked position and orientation errors of all end effectors and \( \mathbf{q}_{\text{min}} \) and \( \mathbf{q}_{\text{max}} \) are the vectors representing the minimum and maximum limits of every joint.

The following quantities are defined in order to solve the optimization problem using the Ipopt solver:

- Gradient of the objective: \( \nabla \mathbf{e} = \mathbf{J}^T \mathbf{e} \).
  
  Where \( \mathbf{J} \) is the the Jacobian of all the end effectors stacked \( \mathbf{J} = \{ \mathbf{J}_0, \ldots, \mathbf{J}_N \} \). The single \( \mathbf{J}_i \) describes \( \dot{\mathbf{X}}_i = \mathbf{J}_i \mathbf{q} \), being \( \hat{\mathbf{X}}_i \in \mathbb{R}^6 \) the linear and angular velocities of the end effector \( i \) in the world reference frame.

- Hessian of the Lagrangian function, for which we use a Gauss-Newton approximation: \( \mathbf{H} \approx \mathbf{J}^T \mathbf{J} \), where \( \mathbf{J} \) is the same Jacobian as in the gradient.

In common optimization problems the Jacobian of the constraints would also be required, but since in our case the constraints consist only in the box constraints represented by the joint limits given that the task of matching the end effector positions are formulated in the objective function, we do not need to formulate the constraints Jacobian.

In the implementation we allow also to use individual position or orientation constraints, as this is useful in some cases, e.g. in circle walking, where by constraining the feet and chest only the pelvis would not rotate with the body, in this case an additional orientation constraint on the pelvis can be imposed.

C. Workflow

The pattern generator is implemented in MATLAB (The MathWorks, Inc.), it takes as input a file containing several parameters for the different walking environments. The script automatically generates the feet pattern (foot hold positions for right and left feet) for a given set of parameters as in Table I.

From the feet pattern the desired ZMP trajectory \( \mathbf{ZMP}_d \) is computed as well as the desired trajectory of the CoM. The ZMP trajectory is generated such that a smooth transition of the ZMP from one foot to the other is performed. This corresponds to have a double support phase of the duration \( T_{\text{switch}} \), which is included in the stride time. This means that the time to perform a step, in the sense of leg swing from one foot hold to the next one, corresponds to \((T_{\text{stride}} - T_{\text{switch}})/2\).

The coM height variation \( \mathbf{CoM}_z \) is computed from a set of heights given at desired times, then a spline interpolation is performed to obtain the \( \mathbf{CoM}_z \) trajectory. As explained previously, this is inspired from human walking and allows the robot to perform larger motions. In the case of level ground walking we compose the pattern by introducing a height offset corresponding to half time of the single support time, such that the robot stretches the stance leg during this phase and then bends it to go towards the double support phase. In the case of stairs the height offset is introduced in correspondence to when the foot has to step over the stair, such that the robot can clear the edge of the stair by stretching the stance leg and impose less force on the knee joint of the stance leg.

The ZMP and CoM trajectories are given as inputs to a function that implements the theory described in section II-A obtaining the full CoM trajectory \( \mathbf{CoM}_d \). Feet trajectories are generated using spline interpolation, by introducing a certain height \( \text{step \_ height} \) to lift the feet off the floor during the swing phase.

The computed CoM and feet trajectories are given as inputs to the inverse kinematics module implemented in C++, where the kinematic model of the robot is loaded from a robot specific URDF (Unified Robot Description File). We use as end effectors the center of the sole reference frames of iCub and the CoM attached to the chest of the robot, at the end of the torso chain, in order to use also the torso joints to compensate for CoM variations. The local CoM is updated to the real CoM at every sample in order to match the correct CoM and have higher stability.

Once the joint trajectories are computed from inverse kinematics, we use an iCub-dedicated C++ software module to execute the motion on the robot through Yarp [13] interfaces.

The module first brings the robot from its initial commanded one using a minimum jerk position controller, then sends joint position commands at a desired thread rate, which should correspond to the sampling time given in the parameters table. The same workflow can be applied both for the iCub in the simulation environment and the real robot. Furthermore the pattern generator and inverse kinematics module can be used with any two legged robot with an available URDF file containing kinematic and dynamic properties.

| Name          | Description                        |
|---------------|------------------------------------|
| \( ts \)     | sampling time                      |
| \( \mathbf{CoM}_z \)  | CoM height                         |
| \( \mathbf{CoM}_z \_\text{offset} \) | height variation of CoM           |
| \( n_{\text{strides}} \) | number of strides                  |
| \( T_{\text{stride}} \)  | time to perform a stride           |
| \( T_{\text{switch}} \)  | double support time                |
| \( \text{step \_ width} \) | distance between two feet          |
| \( \text{step \_ length} \) | distance travelled during a step   |
| \( \theta \)  | inclination during slope walking   |
| \( \text{stair \_ length} \) | length of the stair                |
| \( \text{stair \_ height} \) | height of the stair                |
| \( \text{right \_ step \_ first} \) | by default the first step is the left foot |
| \( \text{type} \) | specifies the walking environment   |

| TABLE I: Pattern generator parameters |
We have applied the illustrated method to generate walking in the cases as per Table II. All the cases illustrated hereafter are referred to experiments on the HeiCub robot. In all the cases we try to push the limits of the robot allowed by our method. Every case has been tested for 5 times to ensure repeatability of the motion. Specific cases will be illustrated hereafter.

It should be noted that walking down stairs has not been performed due to restricted joint limits of the ankle pitch joints. When walking down the stairs, the ankle needs a very wide angle even in humans, where it reaches up to -40°~45°. It needs to be higher in humanoid robots with flat feet such as the iCub, as there is no possibility of performing tip-toe rolling motions.

### A. Key Performance Indicators definition

In the KoroiBot project, a series of performance indicators have been defined as a means to measure the capabilities of a robot. They are called the Key Performance Indicators (KPIs)[14] and are divided in different categories such as human likeness, computational indicators and technical indicators. We use here part of the technical KPIs as a means to summarize the performance of the iCub, which are defined as follows.

- **Cost of transport**

\[
E_{CT} = \frac{\sum_{m=1}^{M} \int_{t_0}^{t_f} I_m(t) V_m(t) dt}{m_{robot} \cdot g \cdot d} \tag{9}
\]

Where \(M\) is the total number of motors, in this case \(M = 15\), \(I_m\) and \(V_m\) are the current and voltage measurements of the motor \(m\), \(m_{robot}\) is the mass of the robot, \(g\) is the gravity and \(d\) is the travelled distance.

- **Froude number**, used for level ground only

\[
Fr = \frac{v_{\text{max}}}{\sqrt{g \cdot h}}, \text{ for } h = l_{\text{leg}} \tag{10}
\]

- **Precision of task execution**: we define them here as the set of tracking errors of the CoM, ZMP and joint angles, computed as RMSE (root mean squared error).

### III. RESULTS

The following walking cases are supported:

- Level ground straight walking
- Slope walking up/down
- Stairs climbing

It is also possible to generate ZMP and CoM trajectories for a given set of feet patterns that is not regular, e.g. step stones, as the algorithm is suitable for arbitrary foot placement. However this has not been included in this paper.

### TABLE II: Walking environments and success rates over 5 trials for the HeiCub robot

| Level ground | \(T_{\text{stride}}\) [s] | Step length [m] | Success rate |
|--------------|----------------|----------------|-------------|
| 8            | 0.10           | 0.01           | 100 %       |
| 6            | 0.10           | 0.01           | 100 %       |
| 5            | 0.10           | 0.01           | 100 %       |
| 4            | 0.10           | 0.01           | 100 %       |
| 3            | 0.10           | 0.01           | 100 %       |
| Slope up     | \(T_{\text{stride}}\) [s] | Slope inclination [deg] | Success rate |
| 8            | 4.5            | 90 %           | 100 %       |
| 6            | 7              | 100 %          | 100 %       |
| 5            | 7              | 100 %          | 100 %       |
| Slope down   | \(T_{\text{stride}}\) [s] | Slope inclination [deg] | Success rate |
| 8            | 4.5            | 90 %           | 100 %       |
| 6            | 7              | 100 %          | 100 %       |
| 5            | 7              | 100 %          | 100 %       |
| Stair up     | \(T_{\text{stride}}\) [s] | Stair height [m] | Success rate |
| 10           | 0.01           | 0.01           | 100 %       |
| 10           | 0.02           | 0.02           | 100 %       |
| 8            | 0.02           | 0.02           | 100 %       |

Please note that quantities such as CoM and ZMP depend on the floating base estimation, which is now implemented in a module that performs the estimation using an odometry based method and the kinematic model of the robot, which is not highly accurate. For this reason the CoM and ZMP errors are not very precise and might be smaller than they result in this paper.
TABLE III: HeiCub joint limits

| Joint                  | Limits            |  
|------------------------|-------------------|
| [hip_pitch, hip_pitch] | [-33, 100]       |
| [hip_roll, hip_roll]   | [-19, 90]        |
| [hip_yaw, hip_yaw]   | [-15, 73]        |
| [knee, knee]          | [-100, 100]      |
| [ankle_pitch, ankle_pitch] | [-56, 27]     |
| [ankle_roll, ankle_roll] | [-24, 24]      |
| torso_pitch           | [-20, 60]        |
| torso_roll            | [-20, 60]        |
| torso_yaw             | [-50, 50]        |

File). From the kinematic point of view the HeiCub and all the other iCub(s) of the same generation are the same. In particular, we have successfully tested flat ground walk also with two full iCub(s) at the IIT facilities, one full size iCub with all the DOF and a full iCub with battery pack, which adds an additional weight of circa 3kg.

The HeiCub robot, being without arms and head, weights 26.4kg. The total height is 0.97m, the leg length (from the hip axis) is 0.51m, and the feet are 0.2m long and 0.1m wide, which are quite small compared to many other state of the art walking humanoid robots. The robot is equipped with a large variety of sensors that include joint and motor encoders, 6 axis force torque sensors, an inertial measurement unit (IMU) and many others, from which we gathered data for the performance analysis.

The standard iCub has restricted joint limits in the leg joints imposed via low level control, which do not correspond to the mechanical limits of the robot. We have extended these limits by leaving only a small margin for protection reasons. The limits are described as in Table III.

Furthermore on the iCub a current limit of 5A is imposed as protection for the motors. This limit is however too conservative for some walking tasks, such as large steps and stair walking. It has been increased on the HeiCub, however, this limit cannot be increased over a certain maximum, which is still not sufficient for more challenging tasks (e.g. higher stairs).

C. Level ground

Level ground walk has been performed with several different walking velocities. The step length was kept constant at 0.1m while the time has been reduced, as shown in Table III.

In Fig. 4 we can see the plot of the feet pattern with the ZMP and CoM trajectories in the $[x, y]$ plane in the fastest case we could achieve, i.e. 4s per stride. In this case we used 1s as double support time, which means that the swing time of a leg in single support phase is of 1.5s. Also, the step length is referred as the distance between the two feet when they are both on the ground, this means that the actual distance travelled by the foot in swing phase is twice the step length, as we can see also in Fig. 5.

The center of mass was allowed to vary in height, and this has demonstrated to be very useful to achieve bigger steps without reaching joint limits as the robot stretches the stance leg while the other one is moving forward.

With 4s the walking motion is stable, as the CoM can be quite closely tracked and the ZMP always lies inside the support polygon as shown in Fig. 5a. The current consumption in this case is higher on the hip roll and knee motors, as they are the most demanding ones during swing phases where the whole weight of the robot is on a single leg.

By further decreasing the time the walking could not be achieved anymore due to high destabilization of the motion. We have computed the CoM, ZMP and joint angles tracking errors, and have observed that there is an increase in the errors with the increase of walking velocity. In the case of stable walking the average CoM tracking error is in the range of 0.03 ~ 0.04m, and average ZMP tracking error of 0.08 ~ 0.09m. The joints tracking error shows an increase with the reduction of stride time, ranging from 0.57° in the 4s case up to 2° in the 4s case. In the case of walking with 3s the errors have been much higher, mainly in the joint tracking error, which is of 6° in average. This led to very unstable motions that brought the CoM far from the desired one and consequently the ZMP out of the support area, from which the robot could not recover.
TABLE IV: KPIs in level ground walk

| Cost of transport $E_{C,F}$ | 4.27 |
|-------------------------------|------|
| Maximum velocity $v_{max}$   | 0.037 [m/s] |
| Froude number $Fr$           | 0.0165 |
| Execution precision          | $Co_M = 0.008$ [m] $ZMP_x = 0.09$ [m] $q_e = 2.11$ [deg] |

Measurements are referred to the case as in Fig. 4.

D. Slope

We have performed experiments with both up and down slopes with two different inclinations and a initial constant step length of 0.05m, increased then to 0.1m. The step length has been increased seeing that the motion with smaller step length has proven to be very stable.

The upslope motion has been demonstrated to be stable with inclination of 7° and a step time of 5s, however we can see in Fig. 5a that the CoM and the ZMP are not tracked as closely as in the level ground case. In particular we can observe that the ZMP is shifted backwards due to the robot slipping slightly downslope. In fact the success rate is of 40% only, against the much more stable case of 6s. From the CoM, ZMP and joint angles tracking errors we could observe that the errors are in average higher than the level ground case. In particular the CoM tracking error is in the range of 0.04 ~ 0.05m and ZMP tracking error of 0.11 ~ 0.13m, while joint angles tracking error of 0.8°~ 1°.

The knee motor requires less current here due to the shorter stride time and the inclination of the slope that allows the knee to have smaller motion range.

In the downslope case we could achieve stable motions with the same inclination of 7°, but with a minimum stride time of 6s, where we have achieved only 20% success rate. As shown in Fig. 6a the ZMP is slightly shifted forward, for the same reason as in the upslope case. In this case the tracking errors are smaller than the upslope case, however the motion is also slower. The CoM tracking error is in the range of 0.4 ~ 0.5m, the ZMP tracking error of 0.08 ~ 0.1m and the joint tracking error of 0.8°~ 1°. Stable motions with smaller stride time have not been achieved successfully because downslope situations are more difficult to handle given that the robot shifts the weight forward and the gravity
and slippage effects destabilize the motion. Knee motor current is also more demanding as the knee joint has to bend more to keep the weight on the stance foot while swinging the other one towards a lower position. Downslope walking is, in fact, also demanding for the knee joint in humans.

It should be noted that walking on slope is in general less stable than the level ground case due to gravity and slippage effects, and therefore tracking errors, mainly of CoM and ZMP, are bigger. With a slope inclination of 7° the ankle pitch joint is very close to its limits. With proper control of the robot a slightly higher inclination might be achieved, however the mechanical limitation does not allow stable motions in higher inclinations.

### E. Stairs

It is a known fact, also from human walking, that stair climbing is demanding for the knee joint. For this reason, in the case of stair walking we tested first for a height of 0.01 m, where we had experienced current limits on the knee motor. To successfully perform the motion the limit has been increased up to 5.5 A. We achieved walking on stairs of 0.02 m high, by increasing the knee current limit up to 6.5 A, as we can see from Fig. 7a. We decided to not increase further the stair height and the walking velocity for safety reasons.

Stair walking has been the one with highest CoM and ZMP tracking errors. While the average joint angles tracking error is of 0.9°, the CoM tracking error reached up to 0.09 m and the ZMP tracking error up to 0.14 m. Proving that in all the tested environments, stair climbing is the most challenging.

As most of the walking humanoid robots, iCub also has legs that are short with respect to its feet size. This means that in stair walking, it is kinematically possible to perform a stair per step as humans do, but it would be dynamically demanding resulting in a very high current required for the knee motor. In order to avoid this, the robot should perform one stair per time, i.e. put both feet on the stair at every second step. In our case, in order to achieve a more “human-like” stair walk, we created a scenario in which the stairs are shifted, as we can see from Fig. 3 and 7a. This means that

### F. KPIs in Stair Climbing

| Cost of transport $E_{CT}$ | 5.08 |
|-----------------------------|------|
| Maximum step height $h_{sp}$ | 0.02 [m] |
| Execution precision | CoM$_e$ = 0.09 [m] ZMP$_e$ = 0.14 [m] $q_e = 0.9$ [deg] |

Measurements are referred to the case as in Fig. 7.
F. Summary

All the performed motions are stable within the limitations of the robot and the applied methodology. We could gather interesting information about the walking capabilities of iCub and measured the KPIs that can serve as reference for future improvements.

From all the analyzed walking scenarios, it is possible to summarize that the joint limits of the iCub are conservative for walking in environments different from flat ground. Mostly the limit lies in the ankle pitch joint motion range. Given that the feet of the robot are one single block with rigid flat surface, without the possibility of tip toe flexibility, there is a necessity of increasing this limit.

A possible way to cope with the current situation without mechanically modifying the robot is to build soles of different shapes for the robot. For instance, in the case of walking down stairs or up slope, the issue lies in the limited angle of the foot while bending towards the lower leg, while the joint does not need to bend in the other direction. In this case it is possible to build a slope shaped sole (a wedge) for the robot, such that the zero position of the ankle pitch is shifted, increasing the motion range. This might introduce further problems, as the height of the robot increases and the range of the ankle pitch is reduced in the opposite direction. Furthermore this method cannot be used in walking down a slope, where the opposite problem exists, and further investigations are necessary to verify if with such external sole the robot is still capable of level ground walking. Nevertheless it is worth testing in further experiments.

IV. CONCLUSIONS

We illustrated and analyzed the performances of the iCub robot with the reduced version HeiCub by using an offline ZMP based pattern generator and an IPOPT based inverse kinematics module.

The software module as illustrated in section II-C can be used to test also other walking environments that do not expect external perturbations, such as beam, step stones, curved walk etc. It allows a fast check of the wanted/expected motion on the simulator or the real robot, to gain useful information to be used in future implementations of control methods such as the ones we discussed in the paper. It also can serve as a trajectory generator, where the trajectory can be then fed to a torque control strategy.

The success of stable walking with the adopted method relies highly on the precision of the execution of the motion with position control, the fine calibration of the relative encoders and the precision of the kinematic model used in the IK. This means that the performances depend also on the fine tuning of the position PID (Proportional, Integral, Derivative) gains.

We have however achieved our goals of implementing walking on the iCub and exploiting its capabilities and limitations in the illustrated scenarios and proposed a possible solution for the restricted joint limits. However an issue that cannot be solved is the current of the motors. The motors can sustain only up to a certain current level, which means that it is not possible to perform challenging motions where high power is required, even though the motion would be kinematically possible. This of course is an exposed limitation when performing not only position-controlled tasks, but more advanced torque-controlled ones, that can now be anticipated through this study.

In future work we will perform walking with online pattern generator using as feedback data gathered through sensors of the robot, with a more accurate floating base estimation module, e.g., using motion capture systems. We will also perform walking with the SEA modules mounted in the knees and ankles, differently than the present work, by means of model-based optimal control taking into account the elasticity properties of the joints. The performance indices presented in this paper will thus serve as a basis of comparison for possible improvements.

REFERENCES

[1] G. Metta, L. Natale, F. Nori, G. Sandini, D. Vernon, L. Fadiga, C. Von Hofsten, K. Rosander, M. Lopes, J. Santos-Victor, et al., The icub humanoid robot: An open-systems platform for research in cognitive development, Neural Networks 23 (8) (2010) 1125–1134.
[2] F. Nori, S. Traversaro, J. Eljaik, F. Romano, A. Del Prete, D. Pucci, icub whole-body control through force regulation on rigid noncoplanar contacts, Frontiers in Robotics and AI 2 (6).
[3] L. Colasanto, N. G. Tsagarakis, D. G. Caldwell, A compact model for the compliant humanoid robot coman, in: Biomedical Robotics and Bioinformatics (BioRob), 2012 4th IEEE RAS & EMBS International Conference on, IEEE, 2012, pp. 688–694.
[4] A. Parmigiani, G. Metta, N. Tsagarakis, The mechatronic design of the new legs of the icub robot, in: Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on, IEEE, 2012, pp. 481–486.
[5] Korobot EU project, last visited - 27.02.2016. URL https://www.korobot.eu/
[6] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, H. Hirukawa, Biped walking pattern generation by using preview control of zero-moment point, in: Robotics and Automation, 2003. Proceedings. ICRA’03. IEEE International Conference on, Vol. 2, IEEE, 2003, pp. 1620–1626.
[7] M. Vukobratović, B. Borovac, Zero-moment point thirtyfive years of its life, International Journal of Humanoid Robotics 1 (01) (2004) 157–173.
[8] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Yokoi, H. Hirukawa, A Realtime Pattern Generator for Biped Walking, in: IEEE International Conference on Robotics and Automation (ICRA), 2002.
[9] S. Kajita, H. Hirukawa, K. Harada, K. Yokoi, Introduction to Humanoid Robotics, Vol. 101 of Springer Tracts in Advanced Robotics, Springer, 2014.
[10] M. L. Felis, Rbdlib: An efficient rigid-body dynamics library using recursive algorithms, Autonomous Robots (2016) 1–8. doi:10.1007/s10462-016-9574-0
[11] T. Sugihara, Solvability-Unconcerned Inverse Kinematics by the Levenberg-Marquardt Method, IEEE Transactions on Robotics 27 (2011) 984–991.
[12] A. Wächter, L. T. Biegler, On the Implementation of a Primal-Dual Interior Point Line Search Algorithm for Large-Scale Nonlinear Programming, Mathematical Programming 106 (1) (2006) 25–27.
[13] P. Fitzpatrick, G. Metta, L. Natale, Towards Long-Lived Robot Genes, Robotics and Autonomous Systems, Elsevier 56 (1) (2008) 29–45.
[14] A. Schubert, D. Clever, K. Mombaur, Key performance indicators for humanoid walking motions defined in the Korobot project, in: International Workshop on Wearable Robotics (WeRob 2014), 2014.