Multi-Contact Planning and Control for a Torque-Controlled Humanoid Robot

Alexander Werner, Bernd Henze, Diego A. Rodriguez, Jonathan Gabaret, Oliver Porges, Máximo A. Roa

***Abstract*— Humanoid robots that need to traverse constrained and uncertain environments require a suitable combination of perception, planning and control. This paper presents an integrated pipeline that allows the robot to autonomously acquire visual information, define step locations, compute feasible multi-contact situations using hands and feet, and generate a motion plan to reach the desired goal even going through different contact states. The execution of the desired path is guaranteed though an optimization-based multi-contact controller. The approach is evaluated in simulations and experiments in two different scenarios using the humanoid robot TORO.**

I. Introduction

Research in humanoid robotics is veering nowadays towards using the robots to autonomously interact and traverse unknown environments, such as disaster scenarios. Such conditions might require from the robot the ability to use multi-contact interactions with the environment, in situations such as egressing from a car or climbing stairs using a handrail, as demonstrated in the recent Darpa Robotics Challenge. Several challenges arise when solving these situations, including the perception of the terrain, the determination of possible contact areas, and the generation of a suitable trajectory to reach the goal position, even by interacting with the environment to enhance the stability of the robot.

This paper presents an integrated framework to enable in a humanoid robot the interaction with an unknown environment. The approach contains four major components: perception of the terrain, step planning, multi-contact motion planning, and multi-contact control. First, as the environment is unknown, the robot must acquire images, recognize suitable geometries for potential contacts and obstacles to be avoided, and create a map that can be dynamically expanded as more information is flowing in. This requires the fusion of inertial and visual information into a world model that is later segmented into primitive geometries. For contacts with the feet, flat regions that can accommodate the foot size are searched for in the image. An approach to solve this problem based on convex optimization was already proposed [1]; to speed up the process of finding such regions, we use a region growing algorithm that quickly identifies flat surfaces, then we apply restrictions on the maximum slope allowed for the stepping surfaces, and identify and store the maximum hull contained within each feasible plane. For the full vision pipeline, a similar approach has been previously considered [2], but tested only in simulation. The real implementation must deal with the imprecision coming from the robot state estimation; to keep a consistent alignment of successive frames, we use a continuous ICP alignment process.

All authors are with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR)

The problem of step planning, or finding a list of step locations to reach a desired goal, has been solved using basically two families of techniques: discrete searches and continuous optimizations. Discrete searches in general require some way to estimate possible displacements from one step to the other, using for instance approximations to the reachable space for the feet [3], or a predefined set of possible footstep locations [4]; the sequence of steps is usually obtained with an A* or RRT algorithm. The problem can also be formulated as an optimization problem on the poses of the footsteps [1], [5], but they also use some sort of geometric approximation to the reachable regions for the footstep locations. In our approach, we use the reachability of the leg given by a capability map [6] to find the real locations of the foot that can be reached from a given configuration; using this map, combined with the real vision data, gives an online estimation of feasible footholds.

Once possible locations of the contacts have been determined, either automatically or manually, a suitable motion plan must be found to take the robot from the initial to the final configuration while traversing stages with different contact configurations. During all these motions, different constraints must be simultaneously respected: overall stability, joint and torque limits, and collision and self-collision avoidance. A multi-level hierarchical control structure that allows prioritization of the constraints was formulated in [7]. Another approach that combines a best-first search on the space of contacts with a posture generator solved as a non-linear optimization problem was proposed in [8], although computational times make this approach unfeasible for real-time generation of trajectories. Our approach follows a similar structure as the latter work. Once a set of desirable contact locations is obtained, the feasibility of the set is evaluated with a hierarchical inverse kinematics solver. Then, the path to move from one configuration to the next one is obtained using a modified Constrained Bidirectional Rapidly-exploring Random Trees (CBiRRT) [9] which handles closed kinematic chains explicitly. A higher level planner also uses an RRT variant to plan the subsequent contact changes until reaching the desired goal configuration. This implementation allows a computation within seconds for low constrained path traversing different contact states.

The planned trajectory is finally executed via a suitable controller. A model based QP multi-contact controller was for instance used in [10] for climbing a ladder with a humanoid robot. Our multi-contact controller

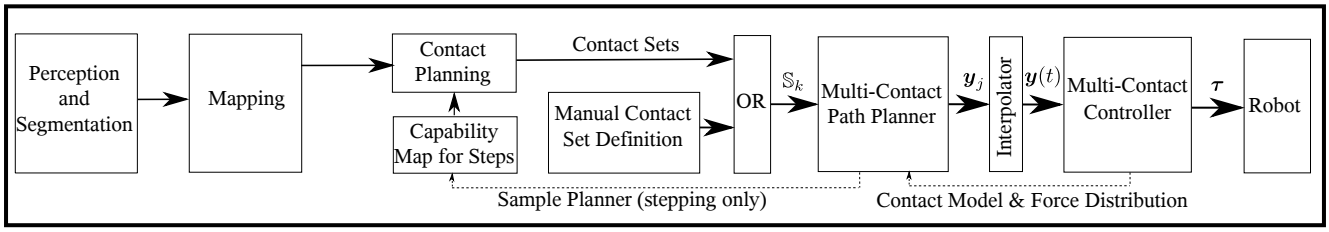


Fig. 1: Pipeline used to generate quasi-static motions either from perception input (for stepping only) or manually (additionally using hand contacts).

solves an optimization problem that stabilizes the robot posture by applying suitable wrenches to guarantee the balance at each time step [11], while providing robustness to unknown external perturbations. The complete pipeline described here, from perception to robust execution, was implemented and tested on TORO, the humanoid robot developed at DLR [12]. The robot is capable of autonomously move to a goal configuration exploiting multi-contact interactions when required. For our current implementation, the footstep locations are automatically computed, but the potential locations for hand contacts are manually labelled; the robot succesfully employs quasi-static trajectories to reach the desired final configuration in a constrained environment, while being robust to perturbations during the execution.

The structure of the paper follows the processing pipeline depicted in Fig. 1. Sec. II focuses on the perception and abstraction of the environment. Sec. III generates step plans to traverse the terrain. The multi-contact planner is presented in V, and Sec. VI describes the control approach used to execute the planned paths. Finally, the simulations and experiments that verify the applicability of the approach are presented in VII.

II. Perception

Our robot is equipped with a Asus Xtion depth camera. The task of the perception subsystem is to localize and percieve two modalities around the robot - obstacles to avoid and planar surfaces suitable for attaching a contact. The surface areas of relevance for stepping range from the footprint of the robot's feet to all ground around the robot. Perceiving large planes in the depth data is challenging due to the distortion of the sensor which causes flat surfaces in reality to appear slightly curved in the data. This issue can be mitigated using the CLAMS [13] undistortion which uses a set of multiplication factors on the depth image to correct the perceived geometry. This step enables us to robustly perceive larger planes, however, there is still a mismatch of perceived distances and distances in the real world. We found out that there is a global scaling factor that brings the depth image closer to the real world distances. This factor is obtained by measuring several distances from the sensor data and matching them with distances measured on the scene.

A. Single frame processing

From each frame information is extracted that is later fused with a global map model. The point cloud is first filtered, a lower distance limit removes any parts of the robot and any other points too high for stepping. The upper filter limit is chosen arbitrarily. The further the point is from the camera the more imprecise is the measurement and it doesn't carry meaningful information. We use the sensor at VGA resolution and apply a voxel filter to reduce the amount of data and improve processing speed. At this point we estimate surface normals to augment next processing steps. RANSAC identifies the major plane in the image. If the current frame is the first shot we consider this plane to be the ground and the normal should represent the gravity vector. The major plane gets removed from the image and we apply region growing to label the cloud clusters. Clusters that show a normal direction not parallel to the gravity vector are removed and considered as obstacles. Remaining clusters are flat surfaces with a normal vector within an allowed threshold, which is based on the limit of inclination we allow the robot to step on. Each plane parameters are estimated again using RANSAC. To reduce the amount of data for global mapping we construct a concave hull of each plane. Remaining points no longer carry meaningful information for us and are removed. Additional information is stored with each hull, namely position in the world frame, surface area and its centroid. Sample of a processed scene and extracted concave hulls is shown in Fig. 3.

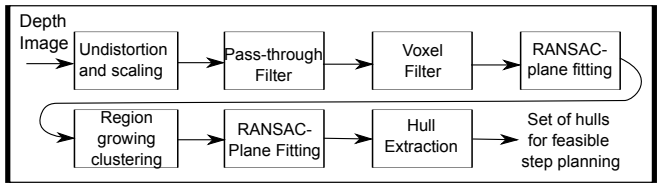
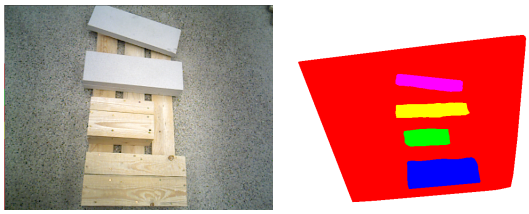


Fig. 2: Single frame processing workflow



(a) Scene view (b) Extracted hulls

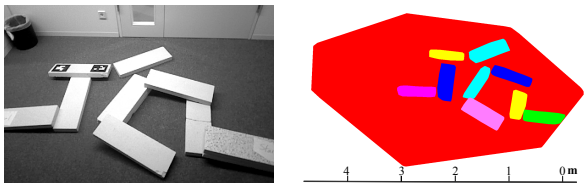
Fig. 3: Single frame processing

B. Continuous mapping

A camera angle covers only a small part of the environment and is not useful for planning more than a few steps. Therefore, we have built a mapping solution which creates a consistent map of the perceived feasible stepping

areas. As the camera moves we transform each new point cloud through the camera pose into the world frame. The image is then processed and the map is updated with the new set of extracted hulls. The update process consists of two modes - hull update and hull addition. If the hull to be added overlaps with a previously seen hull we perform a union of the two. This typically happens when a planar surface is perceived only partially and it gets into the field of view during motion. If there is no overlap a new concave hull is created.

The critical step in every mapping solution is the camera pose estimation. Herein, the a state estimation based on fusion of kinematics and IMU data is used. A small error is caused by slippage of the contacts and by the output characteristics of the IMU. The camera pose obtained from state estimation is a good initial guess, however, the precision is not sufficient for aligning the camera data into the world frame. In parallel camera data is processed to refine the alignment. On initialization we transform the first point cloud using the estimated camera pose into the world frame to form a base for alignment. Every next frame gets transformed using the current camera pose and the alignment gets refined using an Iterative Closest Point algorithm augmented with normals. The normals play an important role to achieve the desired precision. The base point cloud gets replaced by the newly aligned one and the process repeats. An example of resulting map with relaxed plane inclination tolerance is presented in Fig. 4.



(a) Scene view (b) Mapping of the planes

Fig. 4: Map generated by continuous mapping

III. Step Planning

A. Mapping of multi-contact planner capabilities

The presented step planner relies on a quick validation of a large number of feasible foot step locations. Creating a step plan in an obstacle constrained environment the required number of validations is in order of hundreds. For this reason it is not feasible to run the full featured multi-contact planner. Instead, we create an analogy of Capability map, described in [6], of feasible foot step locations. These maps are commonly used to validate reachable frames in grasping applications, support path planing algorithms or mobile robot base placement.

The generation process considers the stance foot to be in the origin of the map while the swing foot is assigned to coordinate frame within the reachable workspace. The workspace is discretized with a resolution of $0.025m$ with 18 different rotations of the frame. We do not need to consider all of the rotational space because we are only stepping on flat surfaces parallel to the ground using the front of the foot in an opening angle of 150 degrees. The generated foot position pairs are validated by the multi-contact planner. If a valid solution within all considered constraints is found, the frame of the swing foot is marked as reachable. The map is queried with a desired coordinate frame for swing foot and it returns the status of reachability. This way we are able to verify over 1500 swing foot steps per millisecond.

B. Step planner

The step planner generates a series of feasible footsteps to reach a given target position. The planner has to respect the perceived environment as well as the planned series of contacts have to be valid with respect to the robot's capabilities. The core of the planner implements a classical A* algorithm with an optional variant of Weighted A*. We chose A* becasue it guarantees to find a path if one exists and it is optimal with respect to a given cost function. In the case of generating longer paths, classical A* becomes slow and we can trade the path optimality for shorter computation time with the Weighted A* variant.

The planner operates on grids with a given resolution (in our case $0.01m$). These grids represent a height and a cost map. Coordinate frames of the grids are aligned with the world frame and are consistent with the perceived map. We initialize the grid as follows. The current state of the map (set of concave hulls) is projected into the cost grid. The cost of cells without associated perception information or covered by obstacles (infeasible for stepping) is ∞. Cells that are within the concave hulls are set to 0 cost. The height map cells contain the projected heigh of the perceived planes. The planner is then parametrized with the current feet positions (x y z ϕ), the desired first swing leg and a goal position. At this point we prepare the A* node expansion. The capability map of the swing foot is projected over the cost and height grids which filters out the infeasible foot configurations with respect to the real world. The remaining subset of poses is further filtered down by checking if the projected footprint of the possible step is fully within the concave hull. There is a small safety margin ($0.01m$) by which we inflated the footprint in order to avoid stepping too close to edges of supporting planes. Remaining foot steps that passed all tests expand the current node. The next node to expand is the one with the lowest cost value. The cost function for A* is defined as $f(n) = g(n) + h(n) + n \times c$ where $g(n)$ is the cost of action, $h(n)$ is the remaining cost estimate and $n \times c$ is a coefficient which scales with the depth of the node tree. This factor helps us to minimize the number of generated steps. The Weighted variant of A* works with $f(n) = g(n) + \varepsilon h(n) + n \times c$ where ε is a relaxing factor. With $\varepsilon > 1$ we speed up the exploration with the risk that the generated path is sub-optimal. Estimated cost to goal is an euclidean distance measure, therefore $h(n) = d(n, target)$. The cost computation $g(h)$ is identical in both cases.

$$g(h) = g(h-1) - d(n, n-1) + \alpha * (\phi_n - \phi_{n-1})^2 + C(x, y, \phi) + (H(n) - H(n-1))^2 \quad (1)$$

Where the meaning of individual contributions are the cost of the previous node, $d(n, n-1)$ denotes the Euclidean distance between the nodes, α is a scaling factor (typically 0.5) applied to the angular difference of the

step orientation, associated values from the cost map and height difference of the footsteps. These contributions to the action cost are motivated by the desired foot placement behaviour. An illustration of the node expansion criteria can be found in Fig. 5. The cost function maximizes step lengths, prevents unnecessary change of orientation and side walking and favours stepping on similar supports of similar heights. Node expansion process repeats until a goal is reached with success or maximum allowed iterations are exceeded. The resulting step plan is created in a standard manner by following the graph from goal to the root node. The generated contact sequence is passed to the multi-contact planner to generate complete poses.

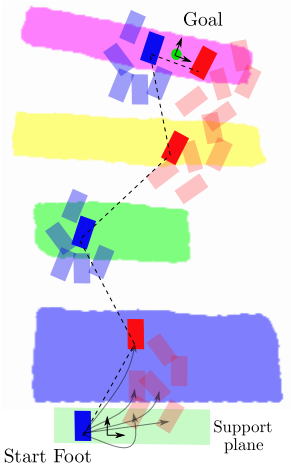


Fig. 5: Generation of the path on the set of concave hulls. Typical computation time for such case is under 1 second

IV. Modelling

The dynamics of a humanoid robot can be described by using a model with a free-floating base e.g. the hip. Let $\boldsymbol{x}_b \in \mathbb{R}^3$ and $\boldsymbol{R}_b \in \mathbb{R}^{3\times3}$ denote the position and orientation of the base relative to the world frame $\mathcal{W}$. The corresponding translational and rotational velocities are given by $\dot{\boldsymbol{x}}_b$ and $\boldsymbol{\omega}_b \in \mathbb{R}^3$, which can be combined into $\boldsymbol{v}_b = (\,\dot{\boldsymbol{x}}_b^T\ \boldsymbol{\omega}_b^T\,)^T$. Considering the angles $\boldsymbol{q} \in \mathbb{R}^n$ and velocities $\dot{\boldsymbol{q}}$ of the n joints, the floating-base dynamics is given by

$$\boldsymbol{M}(\boldsymbol{q})\begin{pmatrix}\dot{\boldsymbol{v}}_b\\ \ddot{\boldsymbol{q}}\end{pmatrix} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{v}}_b,\dot{\boldsymbol{q}})\begin{pmatrix}\boldsymbol{v}_b\\ \dot{\boldsymbol{q}}\end{pmatrix} + \boldsymbol{g}(\boldsymbol{q}) = \begin{pmatrix}\boldsymbol{0}\\ \boldsymbol{\tau}\end{pmatrix} + \boldsymbol{\tau}_{\text{ext}}. \tag{2}$$

In here, $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{(6+n)\times(6+n)}$ denotes the inertia matrix, $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{v}}_b,\dot{\boldsymbol{q}}) \in \mathbb{R}^{(6+n)\times(6+n)}$ the Coriolis and centrifugal matrix and $\boldsymbol{g}(\boldsymbol{q})$ the gravity vector[1]. The control torques are given by $\boldsymbol{\tau} \in \mathbb{R}^n$ and the influence of external forces by $\boldsymbol{\tau}_{\text{ext}} \in \mathbb{R}^{6+n}$. In case all external forces are exclusively acting on the Ψ end effectors $\boldsymbol{\tau}_{\text{ext}}$ becomes

$$\boldsymbol{\tau}_{\text{ext}} = \sum_{i=1}^{\Psi} \boldsymbol{J}_i^T \boldsymbol{F}_i \tag{3}$$

[1]For the sake of simplicity, all dependencies will be dropped in the notations for the remainder of this paper.

with $\boldsymbol{F}_i \in \mathbb{R}^6$ representing the wrench at the i-th end effector and $\boldsymbol{J}_i \in \mathbb{R}^{(6+n)\times6}$ the corresponding Jacobian matrix.

As we presented in [11], it is also possible to choose a frame $\mathcal{C}$ as base frame which shows the same location $\boldsymbol{x}_c \in \mathbb{R}^3$ as the CoM and the same orientation $\boldsymbol{R}_c = \boldsymbol{R}_b$ as the hip. Combining the corresponding translational and rotational velocities $\dot{\boldsymbol{x}}_c$ and $\boldsymbol{\omega}_c \in \mathbb{R}^3$ into $\boldsymbol{v}_c = (\,\dot{\boldsymbol{x}}_c^T\ \boldsymbol{\omega}_c^T\,)^T$ leads to the transformation

$$\begin{pmatrix}\boldsymbol{v}_c\\ \dot{\boldsymbol{q}}\end{pmatrix} = \underbrace{\begin{bmatrix}\boldsymbol{Ad}_{bc} & \boldsymbol{J}_{bc}\\ \boldsymbol{0} & \boldsymbol{I}\end{bmatrix}}_{\bar{\boldsymbol{T}}}\begin{pmatrix}\boldsymbol{v}_b\\ \dot{\boldsymbol{q}}\end{pmatrix} \tag{4}$$

with $\boldsymbol{Ad}_{bc}$ and $\boldsymbol{J}_{bc}$ denoting the Adjoint and the Jacobian of the frame $\mathcal{C}$ relative to the hip. Thus the floating-base dynamics can be rewritten as

$$\bar{\boldsymbol{M}}\begin{pmatrix}\dot{\boldsymbol{v}}_c\\ \ddot{\boldsymbol{q}}\end{pmatrix} + \bar{\boldsymbol{C}}\begin{pmatrix}\boldsymbol{v}_c\\ \dot{\boldsymbol{q}}\end{pmatrix} + \begin{pmatrix}m\boldsymbol{g}_0\\ \boldsymbol{0}\end{pmatrix} = \begin{pmatrix}\boldsymbol{0}\\ \boldsymbol{\tau}\end{pmatrix} + \boldsymbol{\tau}_{\text{ext}}. \tag{5}$$

with the transformed inertia matrix $\bar{\boldsymbol{M}} = \bar{\boldsymbol{T}}^{-T}\boldsymbol{M}\bar{\boldsymbol{T}}^{-1}$ and Coriolis matrix $\bar{\boldsymbol{C}} = \bar{\boldsymbol{T}}^{-T}\boldsymbol{C}\bar{\boldsymbol{T}}^{-1} + \bar{\boldsymbol{T}}^{-T}\boldsymbol{C}\frac{d}{dt}\bar{\boldsymbol{T}}^{-1}$. The influence of gravity is taken into account by $m\boldsymbol{g}_0$ with $m \in \mathbb{R}$ denoting the overall mass of the robot and $\boldsymbol{g}_0 \in \mathbb{R}^6$ the gravitational acceleration vector. Again, if all external forces are exclusively acting on the end effectors $\boldsymbol{\tau}_{\text{ext}}$ becomes

$$\boldsymbol{\tau}_{\text{ext}} = \sum_{i=1}^{\Psi} \bar{\boldsymbol{J}}_i^T \boldsymbol{F}_i \tag{6}$$

with the transformed Jacobian matrix $\bar{\boldsymbol{J}}_i = \boldsymbol{J}_i\bar{\boldsymbol{T}}^{-1}$.

Furthermore, each end effector in contact with the environment is subjected to a contact model $\boldsymbol{F}_i \in \mathcal{F}_i$ limiting the transmittable wrench $\boldsymbol{F}_i$ by

$$f_{i,z} \geq f_i^{\min} \qquad \forall i = 1\dots\psi, \tag{7}$$
$$\left|f_{i,x/y}\right| \leq \mu_i f_{i,z} \qquad \forall i = 1\dots\psi, \tag{8}$$
$$|\tau_{i,z}| \leq \mu_i f_{i,z} \qquad \forall i = 1\dots\psi, \tag{9}$$
$$p_{i,x/y} \in \left[p_{i,x/y}^{\min}, p_{i,x/y}^{\max}\right] \qquad \forall i = 1\dots\psi \tag{10}$$

In order to reduce the complexity for the planner and the controller we assume that each contact shows a rectangular contact area with its longitudinal and transversal axis given by the x- and y-axis of the end effector frame $\mathcal{T}_i$. The z-axis is perpendicular to the contact surface. The unilaterality of the contact is taken into account by (7) limiting the perpendicular contact force to a minimum of $f_i^{\min}$. In order to prevent the end effector from tilting, the Center of Pressure (CoP) $\boldsymbol{p}_i \in \mathbb{R}^3$ is restricted to the contact surface by (10). The friction within the contact surface is approximated with three linear and independent constraints given by (8) and (9) limiting $f_{i,x}$, $f_{i,y}$ and $\tau_{i,z}$. Note that the constraints (7) to (10) can be dropped individually depending on the features of the contacts, for instance (7) in case of an bilateral contact.

V. Multi-Contact Planning

A robot moving through a constraint space must support its own weight through a number of contacts with the environment. Given a model of the environment from

synthetic data or perception, the first step is to generate possible contact points for hands and the feet. These contact points are combined into adjacent contact sets $\mathbb{S}_k$ such that only one link can be attached or detached and arranged such that the transition from $\mathbb{S}_k$ to $\mathbb{S}_{k+1}$ consists of either attaching or detaching one end effector. Furthermore, they are chosen such that they let the robot reach a higher level goal e.g. locomoting to a desired location in the environment.

A contact set associates some of the robots end effectors with a number of desired frame $\boldsymbol{\mathcal{T}}_i$, which are fixed to the environment. The pose of the remaining end effectors can either be left unspecified or follow a desired trajectory in Cartesian space. For each given contact set a feasible, quasi-statically stable configuration $\boldsymbol{y}_j$ is determined by using a constrained inverse kinematics described in Sec. V-B. The configuration $\boldsymbol{y}_j$ can weakly depend on the previous one $\boldsymbol{y}_{j-1}$ in order to minimize the motion between them. To generate a path between $\boldsymbol{y}_{k-1}$ and $\boldsymbol{y}_k$ we use the Constrained Rapidly exploring Random Tree algorithm (CBiRRT) [9] for multi-contact interaction as described in Sec. V-C.

A. Constraints

The proposed planning framework considers the following constraints:

- *Joint Position Limits* imposed by the geometry of the robot.
- *Self-Collisions Avoidance* between robot links using swept sphere volumes.
- *Environment-Collisions Avoidance* between the robot links and the objects of the environment.
- *Singularity Avoidance* ensuring that the controller is able to apply suitable contact forces.
- *Joint Torque Limits* given by the hardware of the robot.
- *Quasi-Static Stability* in order to maintain the balance of the robot with respect to the contact model in (7) to (10).

B. Hierarchical Inverse Kinematic

The inverse kinematics is formulated as a gradient-based optimization problem that minimizes the deviation of all desired tasks $\boldsymbol{\mathcal{T}}_i$ in $\mathbb{S}_k$, and meets all constraints described in V-A. The cost function of the inverse kinematics problem is defined as:

$$\Gamma(\boldsymbol{y}) = \boldsymbol{x}_d - \boldsymbol{x}(\boldsymbol{y}) \tag{11}$$

where $\boldsymbol{x}_d$ is the desired location of the end effectors in $\mathbb{S}_k$ and $\boldsymbol{x}(\boldsymbol{y})$ is their location in the current configuration $\boldsymbol{y}$.

Taking advantage of the high redundancy of humanoid robots, the associated errors of each constraint are minimized using a null-space projector of the body Jacobians $\boldsymbol{J}_i$ of the end-effectors defined in $\mathbb{S}_k$. In this manner, any contribution to meet the constraints does not affect the primary task, i.e., the location of the end-effectors in $\mathbb{S}_k$. Thus, a step in the gradient-based inverse kinematics is expressed as

$$\Delta\boldsymbol{y} = \boldsymbol{K}\boldsymbol{J}^{\dagger}\Delta\boldsymbol{x} + \boldsymbol{N}_0\Delta\boldsymbol{y}_1 \tag{12}$$

where $\Delta\boldsymbol{x} = \boldsymbol{x}_d - \boldsymbol{x}$, $\boldsymbol{J}^{\dagger}$ is the Moore-Penrose pseudoinverse of the $\boldsymbol{J}$, $\boldsymbol{N}_0 = \boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J}$ is the null-space projector, $\Delta\boldsymbol{y}_1$ is a vector containing the contribution of the constraints in joint space, and $\boldsymbol{K}$ is a diagonal matrix that defines the step size and consequently the convergence time.

Generally, there are constraints that are more important than others. For instance, we would rather prefer to go into a singular configuration than violate fixed contacts. Thus, a hierarchical structure is used such that constraints with lower hierarchical level are projected into the null-space of constraints of higher levels, and therefore the former ones do not affect the latter ones. So, Equation 12 can be extended as,

$$\begin{aligned}
\Delta\boldsymbol{y} &= \boldsymbol{K}\boldsymbol{J}^{\dagger}\Delta\boldsymbol{x} + \boldsymbol{N}_0\Delta\boldsymbol{y}_1 && (13)\\
\Delta\boldsymbol{y}_1 &= \boldsymbol{K}_1\boldsymbol{J}_1^{\dagger}\boldsymbol{e_1} + \boldsymbol{N}_1\Delta\boldsymbol{y}_2 && (14)\\
&\vdots\\
\Delta\boldsymbol{y}_{L-1} &= \boldsymbol{K}_{L-1}\boldsymbol{J}_{L-1}^{\dagger}\boldsymbol{e}_{L-1} + \boldsymbol{N}_{L-1}\Delta\boldsymbol{y}_L && (15)\\
\Delta\boldsymbol{y}_L &= \boldsymbol{K}_L\boldsymbol{J}_L^{\dagger}\boldsymbol{e}_L && (16)
\end{aligned}$$

with L number of levels in the hierarchy, $\boldsymbol{e}_l$ the error vector of the constraints in level l, $\boldsymbol{J}_l^{\dagger}$ the pseudoinverse of the constraint Jacobian $\boldsymbol{J}_l$ that maps from the constraint space to joint space, $\boldsymbol{N}_l$ the null-space projector of $\boldsymbol{J}_l$, and $\boldsymbol{K}_l$ a diagonal matrix that serves to weight different constraints in the same level. In case that a hierarchical level contains multiple constraints, the constraint Jacobian matrices and errors are vertically stacked.

For a two level inverse kinematics, for instance, the step $\Delta\boldsymbol{y}$ is defined as:

$$\Delta\boldsymbol{y} = \boldsymbol{K}\boldsymbol{J}^{\dagger}\Delta\boldsymbol{x} + (\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J})[\boldsymbol{J}_1^{\dagger}\boldsymbol{e_1} + (\boldsymbol{I} - \boldsymbol{J}_1^{\dagger}\boldsymbol{J}_1)\boldsymbol{J}_2^{\dagger}\boldsymbol{e}_2]. \tag{17}$$

The required Jacobian for the constraint joint position and singularity avoidance are defined analytically. For collision constraints, numerical Jacobians are calculated based on the penetration. Also for the quasi-static stability constraint presented in detail in the following, the Jacobian is computed using finite differences.

In multi-contact interaction, a force distribution problem arises, i.e., a configuration $\boldsymbol{y}$ can be associated with a subspace of feasible contact wrenches $\boldsymbol{F}_i$. To resolve this force-level redundancy, a quadratic programming (QP) problem which minimizes $\boldsymbol{F}_i$ is formulated. which follows the same contact model presented in Sec. IV.

Additionally, the equations (2) and (3) give a linear mapping from external wrenches to joint torques which allows to include the following constraint is included into the QP problem

$$\boldsymbol{\tau}_{max} \geq \left| \sum_{i=1}^{\Psi} \bar{\boldsymbol{J}}_{i,l}^{T} \boldsymbol{F}_i \right|. \tag{18}$$

where $\bar{\boldsymbol{J}}_{i,l}$ is the joint space part of the end effector Jacobian.

In multi-contact scenarios the contact wrenches $\boldsymbol{F}_i$ are under determined due to the kinematic chain, which is also known as the wrench distribution problem [14]. As this formulation combines the quasi-static stability and joint torque limits in a QP problem, potentially no

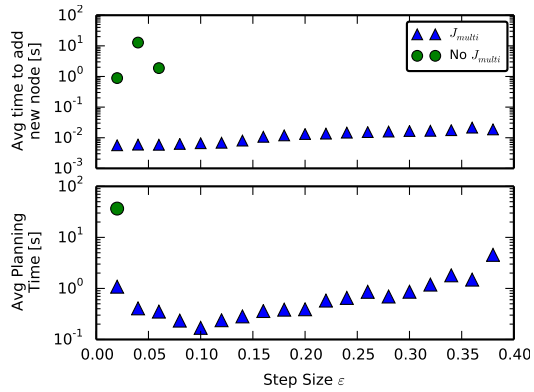


Fig. 6: Average time to add a node and to find a path out of 50 runs. In the upper plot missing points denote that no node was added to the tree and for the lower plot that no feasible path was found. The success rate of the CBiRRT with the proposed J_{multi} was 100% for all ε while pure CBiRRt with no J_{multi} has an success rate of 68% only for the lowest ε, otherwise 0%.

solution exists. To detect this the QP minimizes:

$$e_{qp} = \left\| m\boldsymbol{g}_0 - \sum_{i=1}^{\Psi} \bar{\boldsymbol{J}}_{i,u}^T \boldsymbol{F}_i \right\|_2 \tag{19}$$

which is the difference between the required force on the COM $m\boldsymbol{g}_0$ to maintain the configuration and the force realized on the COM by the contact forces $\boldsymbol{F}_i$. $\bar{\boldsymbol{J}}_{i,u}$ represent the base part of the end effector Jacobian. This quantifies whether the configuration is feasible, in which case this part of the cost function is zero. (19) is used as an inequality constraint in (17).

Finally, a backtracking line search algorithm finds the local minimizer in the computed search direction $\Delta\boldsymbol{y}$.

C. Constraint Rapidly-exploring Random Trees

The planning algorithm presented here is an extension of the Constrained Bidirectional Rapidly-exploring Random Trees (CBiRRT) algorithm [9]. For multi-contact interaction the ability to handle closed kinematic chains e.g. in [9], where CBiRRT is combined with Task Space Regions in order to handle closed chains. Our algorithm, on the other hand, is able to handle closed chains explicitly, as described hereafter.

In order to maintain closed chains, the relative location between the ψ end effectors which are in contact must not change. This can which can be formulated as

$$0 = \boldsymbol{J}_{multi} \quad \dot{\boldsymbol{y}} \tag{20}$$

where $\boldsymbol{J}_{multi}$ contains all relative body Jacobians between all possible pairs of contact end effectors,

$$\boldsymbol{J}_{multi} = \begin{bmatrix} {}^{rel}\boldsymbol{J}_{1,2}^T & {}^{rel}\boldsymbol{J}_{1,3}^T & \dots & {}^{rel}\boldsymbol{J}_{\psi-1,\psi}^T \end{bmatrix}^T \tag{21}$$

Equation (20) suggests that any change $\Delta\boldsymbol{y}$ in the configuration must belong to the nullspace of $\boldsymbol{J}_{multi}$ such that the closed chains are not violated. This observation is exploited and integrated into the CBiRRT algorithm in the manner how the nodes are added to the trees.

The modified CBiRRT works as follows: first, a random configuration $\boldsymbol{q}_{rand}$ is sampled in joint space, and its nearest neighbor $\boldsymbol{q}_{near}$ is found. Then the step $\Delta\boldsymbol{q}$ is defined going from $\boldsymbol{q}_{near}$ towards $\boldsymbol{q}_{rand}$ with a step size of ε:

$$\Delta\boldsymbol{q} = \varepsilon \frac{(\boldsymbol{q}_{rand} - \boldsymbol{q}_{near})}{\|\boldsymbol{q}_{rand} - \boldsymbol{q}_{near}\|} \tag{22}$$

where $\boldsymbol{\varepsilon}$ is a defined step size. $\Delta\boldsymbol{q}$ is then projected into the null-space of $\boldsymbol{J}_{multi}(\boldsymbol{y_{near}})$ and a candidate configuration $\boldsymbol{q}_{cand}$ to add to the graph is defined as

$$\boldsymbol{q}_{cand} = \boldsymbol{q}_{near} + (\boldsymbol{I} - \boldsymbol{J}_{multi}^{\dagger}\boldsymbol{J}_{multi})\Delta\boldsymbol{q}. \tag{23}$$

Due to the drift phenomena caused by the local linearisation of the kinematics in $\boldsymbol{J}_{multi}$ an error accumulates. Therefore additional constraints have to be added to the algorithm to ensure that $\boldsymbol{q}_{cand}$ stays on or close to the manifold. To obtain $\boldsymbol{y}$ for a given $\boldsymbol{q}$ the kinematic relation of one end effector in full contact is used. The performance benefits of this approach are defined in Fig. 6.

D. High level Graph based planning

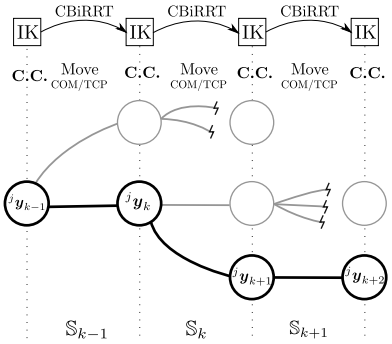


Fig. 7: The inverse kinematics generates configurations $\boldsymbol{y}_k$ (nodes), while the CBiRRT planner generates the path (edges) in between. If there is no feasible path between nodes, a defined number of new configurations are generated. If none of these new configurations can be reached, the node is discarded and the planning continues with the previous node. *C.C.* denotes a contact change.

So far, the presented planning framework is able to find a configuration $\boldsymbol{y}_k$ for a desired contact set $\mathbb{S}_k$ meeting a list of constraints using the inverse kinematics (Sec. V-B), and a path between two configurations through the planning algorithm based on CBiRRTs (Sec. V-C).

However, the planner does not know in beforehand if following the first found path greedily, the goal contact set will be reached. So, the planner can get stalled in a configuration where it can not reach the goal contact set even if a feasible path exists, i.e., the planner is trapped.

Thus, a graph based high level planning is proposed such that new configurations $\boldsymbol{y}_k$ are found using the inverse kinematics if a path between contact sets is not feasible. If after a maximum number of trials, the planner is still not able to find a feasible path, the current node is discarded and the high level planning continues with the previous node. Fig. 7 depicts the high level planning, in which a node represents a configuration $\boldsymbol{y}_k$ and an edge represents the path going from a configuration $\boldsymbol{y}_k$ to the next configuration $\boldsymbol{y}_{k+1}$. When a node is added to the graph, a new contact set is active, therefore a node serves also as representation for a contact change, and

an edge to represent either a COM or TCP movement, during which no contact changes are allowed.

VI. Control

To compute a trajectory from the path generated by the planning algorithms, a minimum jerk interpolator is used in the combined space of kinematics and contact forces $\begin{bmatrix}\boldsymbol{y}^T, & \boldsymbol{W}_i, & \dots\end{bmatrix}^T$. The interpolator can thus limit velocities and accelerations on all these components, making it possible to scale the motion in time as required. In order to follow the quasi-static trajectory generated by the offline planner, we reuse a simplified version of our controller presented in [11]. Thus, here it will be recapitulated only briefly. One feature of the controller is to divide the end effectors into two groups. One is actively used for balancing by generating sufficient contact wrenches. The other can be used for performing an interaction task like manipulating an object, which will be used here for moving an end effector from one contact location to another. Without the loss of generality, let us assume that the end effectors 1 to ψ are used for balancing while the remaining ones ($\psi+1$ to Ψ) are operated in the interaction mode. In order to maintain the balance, the controller stabilizes the CoM frame $\mathcal{C}$ by generating a Cartesian compliance $\boldsymbol{F}_c \in \mathbb{R}^6$ consisting of a translational and rotational stiffness- and damping matrix. The interaction end effectors are also stabilized by a translational and rotational Cartesian compliance, whose wrenches can be combined into $\boldsymbol{F}_{\text{int}} = \begin{pmatrix}\boldsymbol{F}_{\psi+1}^T & \dots & \boldsymbol{F}_{\Psi}^T\end{pmatrix}^T$. The contact wrenches generated with the balancing end effectors $\boldsymbol{F}_{\text{bal}} = \begin{pmatrix}\boldsymbol{F}_1^T & \dots & \boldsymbol{F}_{\psi}^T\end{pmatrix}^T$ will be determined in the remainder of this section.

Based on this notation, the desired error dynamics of the closed loop system can be written as

$$\bar{\boldsymbol{M}}\begin{pmatrix}\dot{\boldsymbol{v}}_c\\ \ddot{\boldsymbol{q}}\end{pmatrix}+\bar{\boldsymbol{C}}\begin{pmatrix}\boldsymbol{v}_c\\ \dot{\boldsymbol{q}}\end{pmatrix}=\boldsymbol{\tau}_{\text{ext}}-\begin{pmatrix}\boldsymbol{F}_c\\ \boldsymbol{0}\end{pmatrix}-\begin{bmatrix}\bar{\boldsymbol{J}}_{\text{bal}}^T & \bar{\boldsymbol{J}}_{\text{int}}^T\end{bmatrix}\begin{pmatrix}\boldsymbol{F}_{\text{bal}}\\ \boldsymbol{F}_{\text{int}}\end{pmatrix}. \tag{24}$$

The left hand side represents the multi-body dynamics, which is excited by the difference between the external forces $\boldsymbol{\tau}_{\text{ext}}$ and the wrenches the controller is supposed to generate ($\boldsymbol{F}_c$, $\boldsymbol{F}_{\text{bal}}$ and $\boldsymbol{F}_{\text{int}}$). Comparing (24) with the dynamic model (5) leads to

$$\begin{pmatrix}\boldsymbol{0}\\ \boldsymbol{\tau}\end{pmatrix}=\begin{pmatrix}m\boldsymbol{g}_0-\boldsymbol{F}_c\\ \boldsymbol{0}\end{pmatrix}-\begin{bmatrix}\bar{\boldsymbol{J}}_{\text{bal,u}}^T & \bar{\boldsymbol{J}}_{\text{int,u}}^T\\ \bar{\boldsymbol{J}}_{\text{bal,l}}^T & \bar{\boldsymbol{J}}_{\text{int,l}}^T\end{bmatrix}\begin{pmatrix}\boldsymbol{F}_{\text{bal}}\\ \boldsymbol{F}_{\text{int}}\end{pmatrix} \tag{25}$$

with $\bar{\boldsymbol{J}}_{\text{bal}}$ and $\bar{\boldsymbol{J}}_{\text{int}}$ partitioned into $\bar{\boldsymbol{J}}_{\text{bal,u}} \in \mathbb{R}^{\psi\times 6}$, $\bar{\boldsymbol{J}}_{\text{bal,l}} \in \mathbb{R}^{\psi\times n}$ and $\bar{\boldsymbol{J}}_{\text{int,u}} \in \mathbb{R}^{(\Psi-\psi)\times 6}$, $\bar{\boldsymbol{J}}_{\text{int,l}} \in \mathbb{R}^{(\Psi-\psi)\times n}$.

The first line of (25) offers 6 equations for computing $\boldsymbol{F}_{\text{bal}}$, which has a size of ψ. Thus, $\boldsymbol{F}_{\text{bal}}$ is underdetermined if more than one end effector is used for balancing ($\psi \geq 6$), which is also known as the force distribution problem [14]. In order to obtain $\boldsymbol{F}_{\text{bal}}$ nevertheless, we solve the following optimization

$$\min_{\boldsymbol{F}_{\text{bal}}}\left(\boldsymbol{F}_{\text{bal}}-\boldsymbol{F}_{\text{bal}}^d\right)^T\boldsymbol{Q}\left(\boldsymbol{F}_{\text{bal}}-\boldsymbol{F}_{\text{bal}}^d\right) \tag{26}$$

with respect to the constraint

$$\boldsymbol{0}=m\boldsymbol{g}_0-\boldsymbol{F}_c-\bar{\boldsymbol{J}}_{\text{bal,u}}^T\boldsymbol{F}_{\text{bal}}-\bar{\boldsymbol{J}}_{\text{int,u}}^T\boldsymbol{F}_{\text{int}} \tag{27}$$

and to the contact model $\boldsymbol{F}_{\text{bal}} \in \mathcal{F}_{\text{bal}}$ specified by (7) to (10). In another step the control torque

$$\boldsymbol{\tau}=-\bar{\boldsymbol{J}}_{\text{bal,l}}^T\boldsymbol{F}_{\text{bal}}-\bar{\boldsymbol{J}}_{\text{int,l}}^T\boldsymbol{F}_{\text{int}} \tag{28}$$

can be computed from the second line of (25).

In order to allow the controller to deal with redundant robots and singular configurations, we added a basic null space controller in [11]

$$\boldsymbol{\tau}'=\boldsymbol{\tau}+\boldsymbol{N}_{\text{null}}\boldsymbol{\tau}_{\text{null}} \tag{29}$$

with $\boldsymbol{\tau}_{\text{null}}$ representing a joint compliance and $\boldsymbol{N}_{\text{null}}$ the corresponding null space projector. Note that the additional torque $\boldsymbol{N}_{\text{null}}\boldsymbol{\tau}_{\text{null}}$ can compromise the optimality of the force distribution as discussed in [11], [15].

VII. Experiments

Climbing stairs while using a handrail was chosen as the experiment to validate the approach. The constructed setup, shown in Fig. 8 includes 3 $0.05m$ high and $0.22m$ deep and $0.7m$ wide steps. The associated handrail is placed $0.75m$ left of the robot center and is $1.2m$ high. The step depth is only $0.02m$ deeper than the length of the foot of the robot. The planned COM Trajectory for the climbing the stairs is shown in Fig. 9.

In terms of the feet, the robot uses uni-lateral contacts capable of generating a full 6-DOF wrench. The hands are instead featuring a bilateral contact generating only forces and no torques.

The stepping stones (not shown) are spaced $0.08m$ apart and are $0.24m$ long and $0.34m$ wide. The stones are placed in an interleaving pattern. Both experiments can be seen on the video associated with the contributed paper.

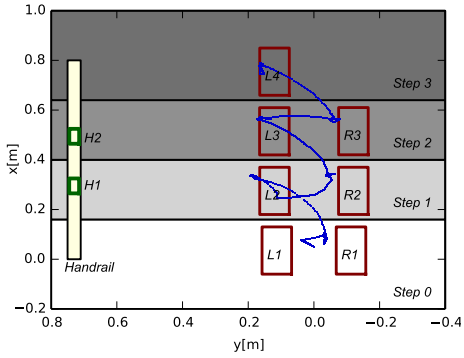


Fig. 9: Planned COM Trajectory for the stair setup. The contact sequence is $[L1, R1]$, $[L1, R1, H1]$, $[R1, H1]$, $[L2, R1, H1]$, $[L2, H1]$, $[L2, R2, H1]$, $[L2, R2]$, $[L2, R2, H2]$, ...

VIII. Conclusion

In execution of the trajectories the contact friction was found to be one of the limiting factors. Despite using a conservative value for the contact friction model in the controller, end effectors in contact often started to slide in situations with low normal forces. The main reason for this was found to be the joint friction and configurations close to singularity. Both can cause a deviation between the real contact wrenches and the one commanded by the controller.

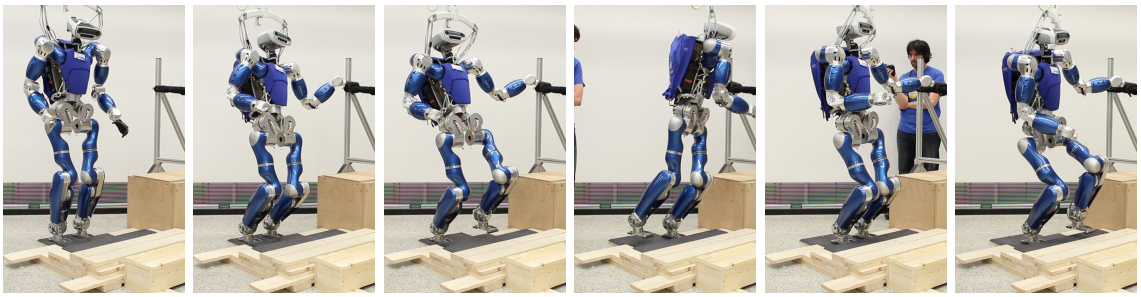


Fig. 8: Snapshots of the experiment climbing stairs. The main challenge of the experiment lies in the correct use of the handrail to support the robot weight, the height of the stairs is $0.05m$. The poses where optimized to have the COM as close as possible to the support foot, this results in quite large motions in the joint space.

Additionally, the base state estimation which relies on the assumption that the contacts do not slide and show infinitesimal stiffness. The first one can lead to a constant deviation in the robots position in the world while the second causes oscillations within the control loop.

When trying the trajectory for the stairs in one pass, the base state estimation accumulated some centimeters of error which can lead to the foot striking one step. The soft impedance control however enabled the robot let the front of the foot slide over the edge of the step and overcome this problem. It is important to have a rather soft stiffness so the reaction forces generated do not destabilize the balancing.

The RGBD Sensor used for perception proved to be prone to calibration errors which results in incorrect scaling and offsets of the detected planes. This was overcome by choosing a more conservative foot size in the planner.

Since much work has been done in the field of planning performance and on-line planning with reduced models, it makes sense to focus rather on completeness and build a planning concept that enables the use of the full capabilities of the robot. To achieve this, a more general contact planner should replace the step planning component making it possible to plan uni-lateral contacts based on perception data using all end effectors. Furthermore we want to explore planing methods more dynamic trajectories between statically stable configurations. Also relevant for a more robust execution is to reduce slippage of the contacts.

References

[1] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2014, pp. 279 – 286.

[2] P. Kaiser, D. Gonzalez-Aguirre, F. Schueltje, J. Borras, N. Vahrenkamp, and T. Asfour, "Extracting whole-body affordances from multimodal exploration," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2014, pp. 1036 – 1043.

[3] N. Perrin, O. Stasse, F. Lamiraux, and E. Yoshida, "Weakly collision-free paths for continuous humanoid footstep planning," in *IEEE Int. Conf. Intelligent Robots and Systems*, 2011.

[4] J. Kuffner, S. Kagami, K. Nishikawi, M. Inaba, and H. Inoue, "Online footstep planning for humanoid robots," in *IEEE Int. Conf. Robotics and Automation*, 2003.

[5] A. Herdt, H. Diedam, P. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, "Online walking motion generation with automatic footstep placement," *Advanced Robotics*, vol. 24, no. 5-6, pp. 719–737, 2010.

[6] O. Porges, T. Stouraitis, C. Borst, and M. A. Roa, "Reachability and capability analysis for manipulation tasks," in *ROBOT2013: First Iberian Robotics Conf.* Springer, 2014, pp. 703–718.

[7] L. Sentis and O. Khatib, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," *Int. J. Humanoid Robotics*, vol. 2, no. 4, pp. 505–518, 2005.

[8] A. Escande, A. Kheddar, and S. Miossec, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," *Robotics and Autonomous Systems*, vol. 61, no. 5, pp. 428–442, 2013.

[9] D. Berenson, S. S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *Int. J. Robotics Research*, p. 0278364910396389, 2011.

[10] J. Vaillant, A. Kheddar, H. Audren, F. Keith, S. Brossette, K. Kaneko, M. Morisawa, E. Yoshida, and F. Kanehiro, "Vertical ladder climbing by the hrp-2 humanoid robot," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2014, pp. 671 – 676.

[11] B. Henze, M. A. Roa, and C. Ott, "Passivity-based whole-body balancing for torque-controlled humanoid robots in multi-contact scenarios," *Int. J. Robotic Research*, vol. Accepted for publication, 2016.

[12] J. Englsberger, A. Werner, C. Ott, B. Henze, M. A. Roa, G. Garofalo, R. Burger, A. Beyer, O. Eiberger, K. Schmid, and A. Albu-Schäffer, "Overview of the torque-controlled humanoid robot TORO," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2014, pp. 916 – 923.

[13] A. Teichman, S. Miller, and S. Thrun, "Unsupervised intrinsic calibration of depth sensors via slam," in *Proceedings of Robotics: Science and Systems*, 2013.

[14] P. M. Wensing, G. B. Hammam, B. Dariush, and D. E. Orin, "Optimizing foot centers of pressure through force distribution in a humanoid robot," *Int. J. Humanoid Robotics*, vol. 10, no. 03, p. 1350027, 2013.

[15] B. Henze, A. Dietrich, and C. Ott, "An approach to combine balancing with hierarchical whole-body control for legged humanoid robots," *IEEE Robotics and Automation Letters*, vol. Accepted for publication, 2016.