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 stances 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 through a passivity-based multi-contact controller. The approach is evaluated in simulations and experiments in different scenarios using the humanoid robot TORO.**

I. Introduction

Research in humanoid robotics is veering nowadays towards autonomous operation in challenging environments, e.g. disaster scenarios. Such conditions might require from the robot the ability to use multi-contact interactions with the environment in order to cope with situations such as traversing a field of debris, egressing from a car or climbing stairs using a handrail, as demonstrated in the recent DARPA Robotics Challenge [1]–[3]. Several challenges arise when solving these situations, including perception of the terrain, determination of possible contact areas, and 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 a humanoid robot to interact 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. Solutions to this problem have been proposed based on convex optimization [4]. To find 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 perception pipeline, a similar approach has been previously considered [5], but tested only in simulation. The real implementation must deal with the imprecision coming from the robot sensors; to keep a consistent alignment of successive frames, we use an alignment based on strong landmarks on the scene.

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 [6], or a predefined set of possible footstep locations [7]; 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 [4], [8], but also using 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 [9] to find the real locations of the foot that can be reached from a given configuration; the combination of this map with the real vision data provides the 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 the whole motion 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 [10]. 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 [11], 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) [12] that explicitly handles closed kinematic chains. 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 traversing a low constrained path using different contact states.

The planned trajectory is finally executed via a suitable controller. A model-based QP multi-contact con-

This work was partially supported by the Helmholtz Association (grant number VH-NG-808) and by the European Commission (grant number H2020-ICT-645097, COMANOID).

All authors are with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Wessling, Germany. Email: {firstname.lastname}@dlr.de.

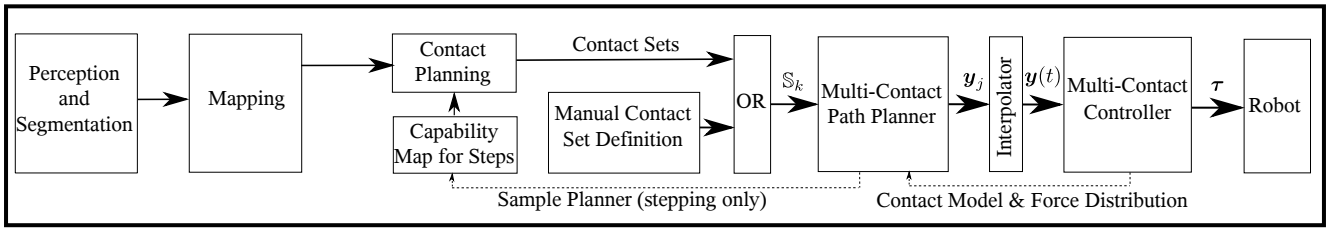


Fig. 1: Pipeline used to generate quasi-static motions either from perception input (for stepping only) or manual annotations (using additional hand contacts).

troller was for instance used in [13] for climbing a ladder with a humanoid robot. Our balance controller is based on passivity control, implementing a compliance that stabilizes the robot by applying suitable contact wrenches to guarantee the balance at each time step while providing robustness to unknown external perturbations [14]. The complete pipeline described here (Fig. 1), from perception to robust execution, was implemented and tested on TORO, the humanoid robot developed at DLR [15], [16]. The robot is capable of autonomously moving towards a goal configuration by 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 labeled; the robot successfully 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. Section II focuses on the perception and abstraction of the environment. Section III presents the planning of footsteps required to traverse the terrain. The multi-contact planner is presented in Section V, and Section 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 Section VII.

II. Perception

Our robot is equipped with an Asus Xtion RGB-D camera. The task of the perception subsystem is to localize and perceive two modalities around the robot: planar surfaces suitable for foot contact attachment, and obstacles to be avoided. The surface areas of relevance for stepping range from the footprint of the robot's feet to all ground around the robot. The used sensors typically suffer two shortcomings, noise and unmodeled distortion. Our first step in the pipeline, segmentation, is sensitive to both. While noise is a local problem affecting fine details in our mapping process, the distortion modifies the depth geometry of the whole image. To minimize the global distortion we use CLAMS - a calibration and undistortion method proposed in [17]. CLAMS takes in a recorded SLAM sequence from the sensor. In an offline phase it finds depth multiplication parameters to make real straight lines appear straight in the depth data as well. This improves the perception of large flat surfaces that would otherwise appear curved in the data. Afterward, we smooth the depth map with a 3×3 Gaussian kernel to lower the effects of noise.

The image processing pipeline consists of two stages: single frame processing and continuous mapping. To achieve better performance, the developed algorithms operate on depth images instead of unstructured 3D point sets.

A. Segmentation, labeling and surface extraction

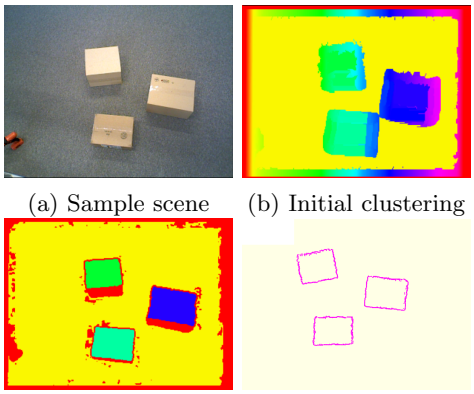
After the initial undistortion and Gaussian blur filter, the 3D point cloud is calculated using a standard pinhole camera model. For fast estimation of surface normals we took inspiration from [18]: Using integral images and adjacency information from the depth image, the normals are estimated as the cross product of vectors from adjacent points in the 3×3 pixel neighborhood.

The next step is the segmentation process, to separate and label planar surfaces in the image. We use a modified non-recursive Flood fill algorithm to find connected components in depth and normal images. Let a depth pixel in image with coordinates i, j be denoted as $D(i, j)$, and $N(i, j)$ the normal associated with this pixel. Mask and label values are similarly $M(i, j)$ and $L(i, j)$. The Flood fill algorithm operates on open-closed list to avoid recursion:

Algorithm 1 Flood fill adjacency condition between pixel coordinates i, j and k, l

```
if D(i,j) − D(k,l) ≤ depth threshold then
    if ∠(N(i,j), N(k,l) ≤ angular threshold then
        Open list ← (i,j)
    end if
end if
```

Algorithm 1 labels each pixel from the open list to a given value. A pixel gets evaluated if there is no label assigned in the label image or if it is not marked as invalid in the mask image. The label is determined in an outer routine that calls the Flood fill on every non-visited pixel. The next pass through the image collects additional information for each label - average normal direction and a member count. Member count determines how large the flat surface is, while the average normal gives the direction. The label image typically contains a lot of noise and several large clusters. All the labels containing under 1% member pixels of the whole image are not considered for further processing. The remaining clusters are checked for orientation, leaving only those whose normals do not deviate more than a given threshold from the gravity vector provided by state estimation. These are the clusters of interest. Due to noise, they might contain small holes, which are corrected by calling



(a) Sample scene (b) Initial clustering

(c) Labeling (d) Extracted hulls

Fig. 2: Segmentation process on depth and normals: a) Color image of a test scene; b) Output of flood fill clustering; c) Clusters of interest with different labels after they were grown; d) Hulls delimiting the flat surfaces.

a region growing process on each cluster of interest. Connected pixels are re-evaluated, and their label is changed if they fall within a given tolerance of a plane defined by the cluster parameters.

At this point we have obtained segmented planar surfaces with a normal direction within the given tolerance for deviation from the gravity vector. One more pass over the label map is required to select the boundaries of the clusters of interest. A boundary pixel is a labeled pixel having a neighbor of different label. At this stage, the boundary pixels are extracted as 3D points into unorganized 3D point sets. Fig. 2 shows an example of the full segmentation process.

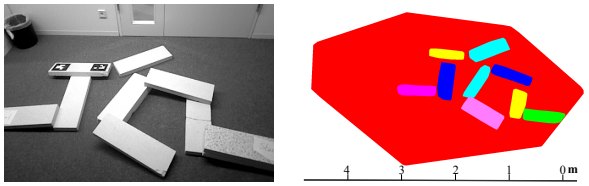
B. Continuous mapping

The field of view of a camera 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 that creates a consistent map of the perceived feasible stepping areas. As the camera moves, every new frame is processed and the map is updated with the new set of extracted hulls.

The mapping procedure is initialized with the first frame. While the output map is always transformed into the world frame, the internal state of the map and camera tracking is always related to the first received frame. The 3D point sets received from segmentation are hulls, i.e. boundary points of planar surfaces in the scene. Each newly received frame is processed in the same way. The points and hull parameters are transformed through the last known camera pose to be aligned with the initial frame. The first guess of the assignment of the hulls is made based on the centroid proximity. The centroid is calculated in the segmentation step using all the member points of the hull, which makes the measurement more stable. If there are at least three successful matches in the scene, we are able to compute a covariance matrix and obtain the rigid transformation between the new frame and the map. The advantage of aligning only the centroid compared to a full ICP solution is the computational speed. At this point it is not guaranteed that the alignment was successful; therefore, the process of correspondence assignment is repeated again. If the correspondences have not changed compared to the situation before the alignment, we proceed to fuse the newly observed hulls with the map. Unmatched hulls are considered as new hulls and are added without fusion.

If the hull to be added overlaps with a previously seen hull, the two hulls are fused. This typically happens when a planar surface is only partially perceived and it gets into the field of view during motion. Two hulls are fused as follows. The centroids and normals of the hulls are updated using a low pass filter with $\frac{1}{5}$ gain on the new observation. The points are fused and simplified through an angular descriptor that discretizes the 0 to 2π range into a user defined number of bins. Upon adding a hull to the map, a zero orientation vector is arbitrarily chosen, going from the centroid to one arbitrary member point of the boundary. Each point in the hull is assigned to a bin b_i according to the angle from the zero vector in the plane of the hull. The new hull boundary is described by only one point per bin; it is the farthest point from the centroid, regardless of whether it comes from the new observation or from the previously simplified map. This process downsamples the boundary point set, which improves the performance of the path planner. It also helps us to handle the corner cases when planes appear and disappear while going in and out of the field of view.

An example of resulting map with the continuous mapping process is presented in Fig. 3. The image processing pipeline employed here runs at $25Hz$ on a single core PC.



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

Fig. 3: Map generated by continuous mapping.

III. Step Planning

To provide a sequence of realizable footsteps in the perceived terrain we use a sampling-based step planning method. The approach takes into account the environment, represented as support planes, and efficiently exploits precomputed knowledge about the capabilities of the robot.

A. Robot capabilities

The step planner relies on a quick validation of a large number of feasible footstep locations. A feasible footstep requires a suitable support surface in the environment, which should be reachable given the kinematic constraints of the robot. To be able to efficiently evaluate hundreds of possible footsteps, we rely on a pre-computed exhaustive map of feasible step locations relative to the current stance foot, similar to the capability maps commonly used in manipulation [9].

To build the map, the swing leg workspace is discretized with a resolution of $0.025m$ in all three translation directions and with 18 orientations of the frame, covering an angle of 150° in front of the foot. This yields 42716 footstep candidates. The generated foot positions

are validated by the multi-contact planner, which can verify up to 1500 swing foot positions per millisecond on the perceived terrain. If a valid solution within all considered constraints is found, the goal frame for the swing foot is marked as reachable.

B. Step planner

The step planner generates a series of feasible footsteps from the start stance N_{start} to a given target stance N_{goal}. A graph-based approach is used here, where the nodes of the graph are contact frame locations of the stance foot and the edges are steps from the map of feasible steps. Alg. 2 summarizes the planning approach. The first node is the initial stance. The planner expands the graph by projecting the capability map for the swing foot onto the current environment, thus finding feasible foot frames relative to the current stance. The validation function, *has_support*() in Alg. 2, tests if all four corner points of the foot are inside one of the support planes. A priority queue is used to order the new nodes generated through this expansion process, with a cost function given by

$$\Gamma(N) = f_{state}(N) + f_{path}(N) \quad (1)$$

$$f_{state}(N) = w_0|x_{goal} - x_N| + w_1|\phi_{goal} - \phi_N| \quad (2)$$

$$f_{path}(N) = w_2 \sum_{N_{start}}^{N} |z_{i+1} - z_i| \quad (3)$$

where x and z are the horizontal and vertical component of the goal position, ϕ the yaw rotation, and w_i suitable weights. The weights were selected to maximize step lengths, prevent unnecessary changes of orientation and side walking, and favor stepping on supports of similar height. The process is repeated until a goal is successfully reached or the maximum number of allowed iterations is exceeded. The final step sequence is determined through backward traversal of the graph, and it is passed on to the multi-contact planner to generate complete robot poses. An illustration of this planning process is shown in Fig. 4, with a typical processing time under 1 second.

Algorithm 2 Generate a feasible footstep sequence

```
queue = [N_start]
while size(queue)! = 0 do
    N_best = sort(queue, Γ)[0]
    for s in step_list do
        N_new = N_best + s
        if has_support(N_new) then
            if norm(N_new − N_goal) ≤ thr then
                return Path(N_start, N_new)
            else
                Insert N_new → queue
            end if
        end if
    end for
end while
```

IV. Modelling

The dynamics of a humanoid robot can be described using a model with free-floating base (e.g. the hip). But as we presented in [14], it is also possible to choose the

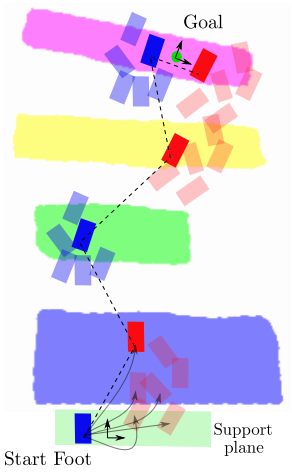


Fig. 4: Generation of a feasible path on the set of hulls.

location of the CoM $\boldsymbol{x}_c \in \mathbb{R}^3$ and the orientation of the hip $\boldsymbol{R}_b \in \mathbb{R}^{3\times3}$ as base coordinates. The corresponding translational and rotational velocities are given by $\dot{\boldsymbol{x}}_c$ and $\boldsymbol{\omega}_b \in \mathbb{R}^3$, which can be combined into $\boldsymbol{v}_c = (\,\dot{\boldsymbol{x}}_c^T \;\boldsymbol{\omega}_b^T\,)^T$. 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

$$\boldsymbol{M}\begin{pmatrix}\dot{\boldsymbol{v}}_c\\ \ddot{\boldsymbol{q}}\end{pmatrix} + \boldsymbol{C}\begin{pmatrix}\boldsymbol{v}_c\\ \dot{\boldsymbol{q}}\end{pmatrix} + \begin{pmatrix}m\boldsymbol{g}_0\\ \boldsymbol{0}\end{pmatrix} = \begin{pmatrix}\boldsymbol{0}\\ \boldsymbol{\tau}\end{pmatrix} + \boldsymbol{\tau}_{\text{ext}}. \quad (4)$$

Here, $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{(6+n)\times(6+n)}$ denotes the inertia matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{v}}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{(6+n)\times(6+n)}$ 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 that all the external forces are exclusively acting on the Ψ end effectors, $\boldsymbol{\tau}_{\text{ext}}$ is given by

$$\boldsymbol{\tau}_{\text{ext}} = \sum_{i=1}^{\Psi} \boldsymbol{J}_i^T \boldsymbol{F}_i \quad (5)$$

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

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

$$f_{i,z} \geq f_{i,z}^{\min} \qquad \forall i = 1\ldots\psi, \quad (6)$$

$$\left|f_{i,x/y}\right| \leq \mu_i f_{i,z} \qquad \forall i = 1\ldots\psi, \quad (7)$$

$$|\tau_{i,z}| \leq \tilde{\mu}_i f_{i,z} \qquad \forall i = 1\ldots\psi, \quad (8)$$

$$p_{i,x/y} \in \left[p_{i,x/y}^{\min}, p_{i,x/y}^{\max}\right] \qquad \forall i = 1\ldots\psi \quad (9)$$

In order to reduce the complexity for the planner and the controller, we assume that each contact has a rectangular contact area with longitudinal and transversal axis given by the x- and y-axis of the end effector frame $\boldsymbol{\mathcal{T}_i}$. The z-axis is perpendicular to the contact surface. The unilaterality of the contact is taken into account by (6) limiting the perpendicular contact force to a minimum of $f_{i,z}^{\min}$. In order to prevent the end effector from tilting, the Center of Pressure (CoP) $\boldsymbol{p}_i \in \mathbb{R}^3$ is restricted by (9) to lie within the contact surface . The friction at the contact surface is approximated with three linear and

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

independent constraints given by (7) and (8), limiting $f_{i,x}$, $f_{i,y}$ and $\tau_{i,z}$. Note that any of the constraints (6) to (9) can be individually dropped depending on the features of the contacts; for instance, (6) is removed in case of a bilateral contact.

V. Multi-Contact Planning

A robot moving through a constrained space must support its own weight using 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 the hands and feet. These contact points are combined into adjacent contact sets $\mathbb{S}_k$ such that only one link can be attached or detached at a time. Thus, the transition from $\mathbb{S}_k$ to $\mathbb{S}_{k+1}$ consists of either attaching or detaching one end effector. Furthermore, the sets $\mathbb{S}_k$ 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 end effectors with a number of desired frames $\boldsymbol{\mathcal{T}}_i$ 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 extend the CBiRRT algorithm [12] for the case of 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 robot hardware.
- *Quasi-Static Stability* in order to maintain the balance of the robot with respect to the contact model in (6) to (9).

B. Hierarchical Inverse Kinematics

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

$$\Gamma(\boldsymbol{y}) = \boldsymbol{x}_d - \boldsymbol{x}(\boldsymbol{y}) \tag{10}$$

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 errors associated to 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

$$\Delta\boldsymbol{y} = \boldsymbol{K}\boldsymbol{J}^\dagger\Delta\boldsymbol{x} + \boldsymbol{N}_0\Delta\boldsymbol{y}_1 \tag{11}$$

where $\Delta\boldsymbol{x} = \boldsymbol{x}_d - \boldsymbol{x}$, $\boldsymbol{J}^\dagger$ is the Moore-Penrose pseudoinverse of $\boldsymbol{J}$, $\boldsymbol{N}_0 = \boldsymbol{I} - \boldsymbol{J}^\dagger\boldsymbol{J}$ 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 prefer to go into a singular configuration rather than violate the 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. Therefore, (11) can be extended as

$$\begin{aligned}
\Delta\boldsymbol{y} &= \boldsymbol{K}\boldsymbol{J}^\dagger\Delta\boldsymbol{x} + \boldsymbol{N}_0\Delta\boldsymbol{y}_1 && (12)\\
\Delta\boldsymbol{y}_1 &= \boldsymbol{K}_1\boldsymbol{J}_1^\dagger\boldsymbol{e_1} + \boldsymbol{N}_1\Delta\boldsymbol{y}_2 && (13)\\
&\vdots\\
\Delta\boldsymbol{y}_{L-1} &= \boldsymbol{K}_{L-1}\boldsymbol{J}_{L-1}^\dagger\boldsymbol{e}_{L-1} + \boldsymbol{N}_{L-1}\Delta\boldsymbol{y}_L && (14)\\
\Delta\boldsymbol{y}_L &= \boldsymbol{K}_L\boldsymbol{J}_L^\dagger\boldsymbol{e}_L && (15)
\end{aligned}$$

with L the 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 $\boldsymbol{J}_l$ mapping from 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 weigh 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

$$\Delta\boldsymbol{y} = \boldsymbol{K}\boldsymbol{J}^\dagger\Delta\boldsymbol{x} + (\boldsymbol{I} - \boldsymbol{J}^\dagger\boldsymbol{J})[\boldsymbol{J}_1^\dagger\boldsymbol{e_1} + (\boldsymbol{I} - \boldsymbol{J}_1^\dagger\boldsymbol{J}_1)\boldsymbol{J}_2^\dagger\boldsymbol{e}_2] \tag{16}$$

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

A force distribution problem naturally arises in multi-contact interaction, 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 that minimizes $\boldsymbol{F}_i$ is formulated using the same contact model presented in Sec. IV [14].

Additionally, (4) and (5) give a linear mapping from external wrenches to joint torques, which allows including the following constraint into the QP problem

$$\boldsymbol{\tau}_{max} \geq \left| \sum_{i=1}^{\Psi} \boldsymbol{J}_{i,l}^T \boldsymbol{F}_i \right| \tag{17}$$

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

In multi-contact scenarios the contact wrenches $\boldsymbol{F}_i$ are underdetermined due to the branched kinematic chain, which is also known as the wrench distribution problem.

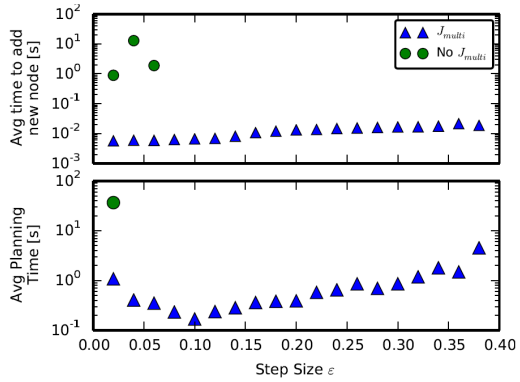


Fig. 5: Average time to add a node and to find a path for 50 runs. In the upper plot, missing points denote that no node was added to the tree, and for the lower plot that no feasible path was found. The success rate of the CBiRRT with the proposed J_{multi} was 100% for all ε, while pure CBiRRt with no J_{multi} has an success rate of 68% only for the lowest ε, otherwise 0%.

As this formulation combines the quasi-static stability and joint torque limits in a QP problem, potentially no solution exists. To detect this case, the QP minimizes the difference between the required force on the CoM to maintain the configuration, $m\boldsymbol{g}_0$, and the force realized on the CoM by the contact forces $\boldsymbol{F}_i$:

$$e_{qp} = \left\| m\boldsymbol{g}_0 - \sum_{i=1}^{\Psi} \boldsymbol{J}_{i,u}^T \boldsymbol{F}_i \right\|_2 \tag{18}$$

where $\boldsymbol{J}_{i,u}$ represents the base part of the end effector Jacobian. This quantifies if the configuration is feasible, in which case this part of the cost function is zero. (18) is then used as an inequality constraint in (16).

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

C. Constrained Rapidly-exploring Random Trees

The planning algorithm presented here is an extension of the CBiRRT algorithm [12]. The CBiRRT has been previously combined with Task Space Regions [12] to specify constraints such as closed chains. Our algorithm, on the other hand, is able to explicitly handle closed chains, as described below.

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

$$0 = \boldsymbol{J}_{multi}\ \dot{\boldsymbol{y}} \tag{19}$$

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

$$\boldsymbol{J}_{multi} = \begin{bmatrix} {}^{rel}\boldsymbol{J}_{1,2}^T & {}^{rel}\boldsymbol{J}_{1,3}^T & \dots & {}^{rel}\boldsymbol{J}_{\psi-1,\psi}^T \end{bmatrix}^T \tag{20}$$

As (19) suggests, 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 way that 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 $\Delta\boldsymbol{q}$ going from $\boldsymbol{q}_{near}$ towards $\boldsymbol{q}_{rand}$ is defined as

$$\Delta\boldsymbol{q} = \varepsilon \frac{(\boldsymbol{q}_{rand} - \boldsymbol{q}_{near})}{\|\boldsymbol{q}_{rand} - \boldsymbol{q}_{near}\|} \tag{21}$$

where $\boldsymbol{\varepsilon}$ is a predefined 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

$$\boldsymbol{q}_{cand} = \boldsymbol{q}_{near} + (\boldsymbol{I} - \boldsymbol{J}_{multi}^{\dagger}\boldsymbol{J}_{multi})\Delta\boldsymbol{q}. \tag{22}$$

Due to drift phenomena caused by the local linearization of the kinematics in $\boldsymbol{J}_{multi}$, an error is accumulated. Therefore, additional constraints must 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 illustrated in Fig. 5.

D. Joint level path generation from a contact sequence

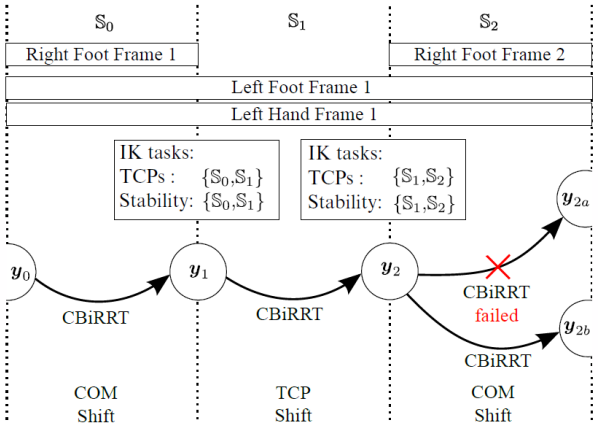


Fig. 6: Robust generation of a feasible path using a contact sequence $[\mathbb{S}_0, \mathbb{S}_1, \mathbb{S}_2]$.

The planning framework so far is able to find a configuration $\boldsymbol{y}_k$ for a desired contact set $\mathbb{S}_0$ meeting a list of constraints (Sec. V-B), and a path between two configurations through the planning algorithm based on CBiRRTs (Sec. V-C). However, to accomplish more complex tasks such as climbing stairs a robust path generation is required, which might involve changes between different contact sets.

A task is then defined with an initial and final configuration, and a feasible joint space path to go from the initial to the final configuration must be found. The motion sequence contains alternating phases of moving an end effector to a new contact and moving the center of mass while maintaining the same contacts (Fig. 6). Two consecutive phases interleave at one configuration $\boldsymbol{y}$ that satisfies the conditions defined by both contact sets. Because the solution at every step in not unique, the planning process can lead to unfeasible configurations. When the inverse kinematics fails, the process is repeated with higher perturbations to detect all local minima. If no path is found between two configurations, a new goal configuration is generated using a different initial condition for the inverse kinematics solver.

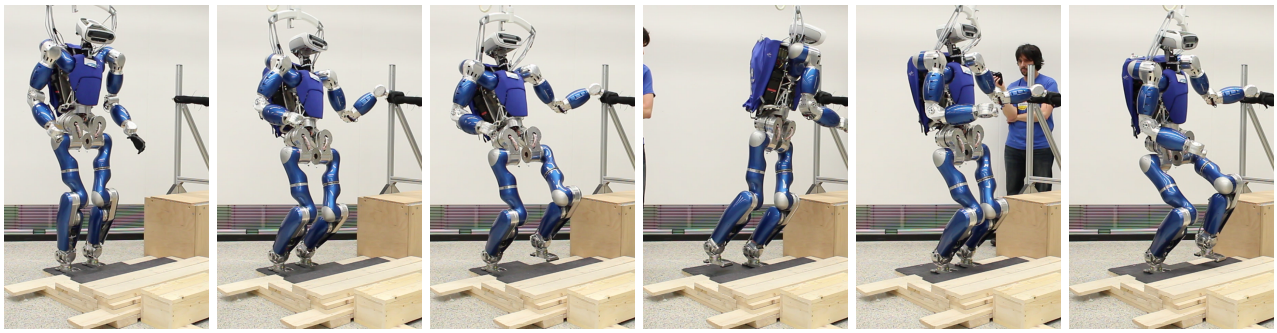


Fig. 7: Snapshots of the experiment climbing stairs. The main challenge of the experiment lies in the correct use of the handrail to support the robot weight, the height of the stairs is $0.05m$. The poses were optimized to have the CoM as close as possible to the supporting foot, which results in quite large motions in the joint space.

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 $\begin{bmatrix}\boldsymbol{y}^T, & \boldsymbol{F}_i, & \ldots\end{bmatrix}^T$. 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 use a simplified version of our controller presented in [14], summarized below. The controller divides the end effectors into two groups: one is actively used for balancing by generating the required contact wrenches, while the other one can be used for performing an interaction task like manipulating an object. The latter group is used here for moving an end effector to a new contact location. Without 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 used for interaction tasks. In order to maintain the balance, the controller stabilizes the CoM position $\boldsymbol{x}_c$ and the hip orientation $\boldsymbol{R}_b$ 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 $\boldsymbol{F}_{\text{int}} = \left(\boldsymbol{F}_{\psi+1}^T \ldots \boldsymbol{F}_{\Psi}^T\right)^T$. The contact wrenches generated with the balancing end effectors $\boldsymbol{F}_{\text{bal}} = \left(\boldsymbol{F}_1^T \ldots \boldsymbol{F}_{\psi}^T\right)^T$ will be determined in the remainder of this section.

The desired error dynamics of the closed loop system can be written as

$$\boldsymbol{M}\begin{pmatrix}\dot{\boldsymbol{v}}_c\\ \ddot{\boldsymbol{q}}\end{pmatrix}+\boldsymbol{C}\begin{pmatrix}\boldsymbol{v}_c\\ \dot{\boldsymbol{q}}\end{pmatrix}=\boldsymbol{\tau}_{\text{ext}}-\begin{pmatrix}\boldsymbol{F}_c\\ \boldsymbol{0}\end{pmatrix}-\begin{bmatrix}\boldsymbol{J}_{\text{bal}}^T & \boldsymbol{J}_{\text{int}}^T\end{bmatrix}\begin{pmatrix}\boldsymbol{F}_{\text{bal}}\\ \boldsymbol{F}_{\text{int}}\end{pmatrix} \tag{23}$$

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 (23) with the dynamic model (4) leads to

$$\begin{pmatrix}\boldsymbol{0}\\ \boldsymbol{\tau}\end{pmatrix}=\begin{pmatrix}m\boldsymbol{g}_0-\boldsymbol{F}_c\\ \boldsymbol{0}\end{pmatrix}-\begin{bmatrix}\boldsymbol{J}_{\text{bal,u}}^T & \boldsymbol{J}_{\text{int,u}}^T\\ \boldsymbol{J}_{\text{bal,l}}^T & \boldsymbol{J}_{\text{int,l}}^T\end{bmatrix}\begin{pmatrix}\boldsymbol{F}_{\text{bal}}\\ \boldsymbol{F}_{\text{int}}\end{pmatrix} \tag{24}$$

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

The first line of (24) 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 leads to a force distribution problem. $\boldsymbol{F}_{\text{bal}}$ is obtained by solving

$$\min_{\boldsymbol{F}_{\text{bal}}}\left(\boldsymbol{F}_{\text{bal}}-\boldsymbol{F}_{\text{bal}}^d\right)^T\boldsymbol{Q}\left(\boldsymbol{F}_{\text{bal}}-\boldsymbol{F}_{\text{bal}}^d\right) \tag{25}$$

with respect to the constraint

$$\boldsymbol{0}=m\boldsymbol{g}_0-\boldsymbol{F}_c-\boldsymbol{J}_{\text{bal,u}}^T\boldsymbol{F}_{\text{bal}}-\boldsymbol{J}_{\text{int,u}}^T\boldsymbol{F}_{\text{int}} \tag{26}$$

and to the contact model $\boldsymbol{F}_{\text{bal}} \in \mathcal{F}_{\text{bal}}$ specified by (6) to (9). In the next step, the control torque is computed using the second line of (24):

$$\boldsymbol{\tau}=-\boldsymbol{J}_{\text{bal,l}}^T\boldsymbol{F}_{\text{bal}}-\boldsymbol{J}_{\text{int,l}}^T\boldsymbol{F}_{\text{int}} \tag{27}$$

In order to allow the controller to deal with redundant robots and singular configurations, we added a basic null space controller in [14]

$$\boldsymbol{\tau}'=\boldsymbol{\tau}+\boldsymbol{N}_{\text{null}}\boldsymbol{\tau}_{\text{null}} \tag{28}$$

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 [14], [19].

VII. Experiments

Climbing stairs while using a handrail was chosen as the experiment to validate the approach. The constructed setup, shown in Fig. 7, includes three $0.05m$ high, $0.22m$ deep and $0.7m$ wide steps. The step depth is only $0.02m$ deeper than the length of the robot feet. The associated handrail is placed $0.75m$ left of the robot center and is $1.2m$ high. The planned CoM Trajectory for climbing the stairs is shown in Fig. 8.

In terms of contacts, the feet can generate forces and torques subjected to the contact model (6) to (9). In contrast to the unilaterality of the feet, the hands are modeled as bilateral contacts. Furthermore, it is assumed that the hands can only transmit forces and no torques.

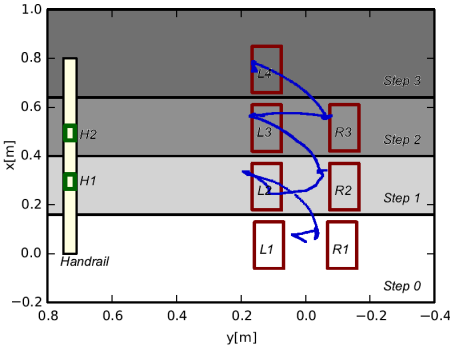


Fig. 8: Planned CoM Trajectory for the stair setup. The contact sequence is $[L1, R1]$, $[L1, R1, H1]$, $[R1, H1]$, $[L2, R1, H1]$, $[L2, H1]$, $[L2, R2, H1]$, $[L2, R2]$, $[L2, R2, H2]$...

VIII. Final discussion

This paper presented a full pipeline to allow a humanoid robot to acquire information of the surrounding terrain, plan a multi-contact solution to go from an initial to a final stance, and execute the generated trajectory using a passivity-based controller. During execution of the trajectories, contact friction was found to be one of the major limiting factors for achieving a good performance. Despite using a conservative value for the friction coefficient in the controller, end effectors in contact often started to slide in situations with low normal forces. The main reason for this behavior was the high joint friction combined with required configurations close to singularities. Both can cause a deviation between the real contact wrenches and the ones commanded by the controller. Additionally, the base state estimation relies on the assumption that the contacts do not slide and are totally rigid. A sliding contact can lead to a constant deviation of the robot position in the world, while the real contact stiffness might cause oscillations within the control loop.

When trying the full trajectory for the stairs in one shot, the base state estimation accumulates some centimeters of error, which can lead to the foot striking one step of the stair. However, the soft impedance control lets the robot slide the frontal part of the foot over the edge of the step to overcome this problem. It is important to have a rather soft stiffness so the reaction forces generated do not destabilize the overall balance.

The RGB-D 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.

Much work has been done in the field of on-line planning with reduced models, yet it makes sense to focus 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 unilateral contacts based on perception data using all end effectors. Furthermore, as a future work we want to explore planing methods that provide more dynamic trajectories between statically stable configurations.

References

[1] M. Fallon, S. Kuindersma, S. Karumanchi, M. Antone, T. Schneider, T. Dai, C. D'Arpino, R. Deits, M. DiCicco, D. Fourie, T. Koolen, P. Marion, M. Posa, A. Valenzuela, K. Yu, A. Shah, A. Iagnema, R. Tedrake, and S. Teller, "An architecture for online affordance-based perception and whole-body planning," *J. Field Robotics*, vol. 32, no. 2, 2014.

[2] M. Johnson, B. Shrewsbury, S. Bertrand, T. Wu, D. Duran, M. Floyd, P. Abeles, D. Stephen, N. Mertins, A. Lesman, J. Carff, W. Rifenburgh, P. Kaveti, W. Straatman, J. Smith, M. Griffioen, B. Layton, T. de Boer, T. Koolen, P. Neuhaus, and J. Pratt, "Team IHMC's lessons learned from the DARPA robotics challenge trials," *J. Field Robotics*, vol. 32, no. 2, 2014.

[3] C. Atkeson, B. Babu, N. Banerjee, D. Berenson, C. Bove, X. Cui, M. DeDonato, R. Du, S. Feng, P. Franklin, M. Gennert, J. Graff, P. He, A. Jaeger, J. Kim, K. Knoedler, L. Li, C. Liu, X. Long, T. Padir, F. Polido, G. Tighe, and X. Xinjilefu, "No falls, no resets: Reliable humanoid behavior in the DARPA robotics challenge," in *IEEE-RSJ Int. Conf. Humanoid Robots*, 2015, pp. 623–630.

[4] 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.

[5] 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.

[6] 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, pp. 4408 – 4413.

[7] 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, pp. 932 – 937.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] D. Berenson, S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *Int. J. Robotics Research*, vol. 30, no. 12, pp. 1435–1460, 2011.

[13] 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.

[14] 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*, DOI: 10.1177/0278364916653815, 2016.

[15] 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.

[16] B. Henze, A. Werner, M. A. Roa, G. Garofalo, J. Englsberger, and C. Ott, "Control applications of TORO - a torque controlled humanoid robot," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2014, pp. 841 – 841.

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

[18] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke, "Real-time plane segmentation using RGB-D cameras," in *Robot Soccer World Cup*. Springer, 2011, pp. 306–317.

[19] 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. 1, no. 2, pp. 700 – 707, 2016.