Faculdade de Engenharia da Universidade do Porto

Whole-Body End-Pose Planning for High-Degree-of-Freedom Robots on Uneven and Inclined Surfaces

Henrique Manuel Martins Ferrolho

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Rosaldo J. F. Rossetti, PhD

Second Supervisor: Hugo Sereno Ferreira, PhD

External Supervisor: Sethu Vijayakumar, PhD

July 24, 2017

Contact Information:

Henrique Manuel Martins Ferrolho
henrique.ferrolho@gmail.com
https://ferrolho.github.io/

Faculdade de Engenharia da Universidade do Porto
Rua Dr. Roberto Frias
4200-465 Porto
Portugal

School of Informatics, University of Edinburgh
Informatics Forum, 10 Crichton Street
Edinburgh, EH8 9AB
United Kingdom

"Whole-Body End-Pose Planning for High-Degree-of-Freedom Robots on Uneven and Inclined Surfaces"

Whole-Body End-Pose Planning for High-Degree-of-Freedom Robots on Uneven and Inclined Surfaces

Henrique Manuel Martins Ferrolho

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Henrique Lopes Cardoso, PhD
External Examiner: Vladimir Ivan, PhD
Supervisor: Rosaldo J. F. Rossetti, PhD

July 24, 2017

Abstract

During the last few years there have been significant improvements in the field of humanoid robotics. More powerful workstations capable of running more accurate - and therefore more computationally demanding - simulations, and the rise of new generations of humanoid robots with better hardware, have enabled researchers to keep pushing the boundaries and create novel methods to improve the perception and motion of these robots.

Motion planning is the area of robotics which concerns with how and when a robot should move a part of itself, and the execution of such motion. Motion planning has been a thoroughly investigated area, but not all of the challenges related to it are solved yet.

Robots with a fixed base and few Degrees of Freedom (DoF), *e.g.* the industrial robotic arms that revolutionised the automotive industry, have been used as a means to approach the problem of motion planning. Often these type of robots are associated with an isolated environment, in which they do not have to interact with people. Researchers have developed successful motion planning algorithms to operate robots in these environments. Nonetheless, those approaches fall short when humanoid robots are taken into consideration. Applications aimed towards humanoid robots have to take into account the characteristics often associated with them: many DoF, a floating base, and balance and dynamic constraints.

Implementing autonomous solutions with safe human interaction in complex and dynamic environments, considering biped balance and possible external interferences is non-trivial. Our goal is to tackle the problem of high dimensional kinematic and dynamic motion planning. Namely, we will focus on the subproblem of humanoid end-pose planning on uneven terrains.

Acknowledgements

I would first like to thank the colleagues, flatmates, and friends I had the chance to meet during my stay in Edinburgh. My exchange program at the University of Edinburgh changed my openness to experience, and gave me different perspectives concerning innumerous social, political, and ethical issues.

To the folks at the lab G.03 of the Informatics Forum: thank you so much for all the knowledge you shared with me. I was truly lucky to get to know you, and have the chance to learn from you. Yang Yiming and Wang Ruiqiu, your patience is remarkable when it comes to teaching me Chinese. Wolfgang and Vlad, thank you for all the robot banter and weekly doses of social interaction at Teviot.

Professor Sethu Vijayakumar, thank you so much for all the guidance during my thesis. You are an extraordinary role model to look up to.

Finally, I cannot help but mention my favourite English teacher, Sandra Albuquerque. You gave me the tools to understand one of the most spoken languages in the world. I hope the correctness of this thesis lives up to your expectations.

Henrique Ferrolho

Agradecimentos

O meu sincero agradecimento ao Professor Rosaldo Rossetti por todo o apoio, orientação e supervisão durante a escrita desta tese.
Gostava também de agradecer a todos os Professores da Faculdade de Engenharia da Universidade do Porto com quem contactei, por tudo aquilo que me ensinaram durante o meu percurso académico.
Para todos os meus amigos de Viseu e do Porto: todos vocês me marcaram de alguma forma em alguma altura da minha vida. Muito obrigado a todos.
Um enorme obrigado ao meu amigo Hugo Sereno. Desde sempre que foste um exemplo para mim, e alguém com quem eu sempre posso contar. Espero que continuemos a ter a oportunidade de nos encontrarmos um pouco por todo o mundo, e de partilhar umas cervejas enquanto criticamos a vida.
Finalmente, gostava de exprimir a minha mais sincera gratidão para com a minha mãe, Alda Ferrolho, o meu pai, António Ferrolho, e a minha querida irmã, Ana Rita Ferrolho. Sempre me apoiaram durante toda a minha vida, e o meu único desejo é deixar-vos o mais orgulhosos possível em relação às minhas ambições e objectivos pessoais.

Henrique Ferrolho

"We are all apprentices in a craft where no one ever becomes a master."

Ernest Hemingway

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides the reader with an overview of this dissertation. It introduces the context, motivation, and the goals of this project. Finally, it presents the overall structure of this document.

1.1 Context

Robotics is a fascinating research subject. Most people think of robots for industrial purposes only, but robotics has been applied to other subjects lately, such as autonomous vehicles [PR12; Fig+09], human care and assistance [Pet+11], amongst others.

Humanoid robots are very complex systems created to fulfil tasks designed for people. In order to simulate — Shafii et al. [SRR11] have compared two (2) of many humanoid simulators — a human with high accuracy, robots are designed with many Degrees of Freedom (DoF). However, programming these robots is no easy task due to all the conditions that need to be taken into account, *e.g.* constant bipedal balance, safe human interaction, redundant DoF, and adaptation to changes in complex environments.

It is not practical to manually teleoperate a humanoid robot. It is necessary to pursue autonomy or semi-autonomy of complex behaviours on robots. Take National Aeronautics and Space Administration (NASA) Valkyrie as an example: NASA's ultimate goal is to send multiple robots like Valkyrie to Mars, in order to prepare human settlements for future manned missions. Space exploration demands robotic automation due to the delays and limited data rates of interplanetary communication. But there are many other reasons to

automate and push the boundaries of high dimensional humanoid robots: elderly care and assistance, rescue missions, automation of services, construction, etc. Nonetheless, the goal of creating autonomous robots leads to many challenges related to perception, localisation, motion planning, amongst others.

This dissertation is part of the *Open Humanoids Project* [Tea], which is a collaboration between the University of Edinburgh (UoE) and the Massachusetts Institute of Technology (MIT) to develop the NASA Valkyrie and Atlas Humanoid Project. This research was carried out at the Robotics facilities of the School of Informatics at the University of Edinburgh.

1.2 Motivation

One of the automation challenges of humanoid robotics is the planning of whole-body end-poses capable of reaching difficult grasping targets in realistic environments. Currently, in most practical applications, the end-pose is provided manually by an operator because an automated solution is not trivial, especially on uneven or inclined terrains. These end-poses need to be quasi-static balanced, collision-free, and possible to transition to from the robot's initial stance location and configuration. Once a valid whole-body end-pose is found, it can be used as a goal state for other planners, such as walking and motion planners.

Zacharias et al. [ZBH07] introduced the concept of Reachability Map (RM), which stores the reachable workspace regions of a fixed-base robot. Afterwards, Vahrenkamp et al. [VAD13] presented the idea of inverting the RM, which results in a new map that encodes the feasible stances for a given grasping target. Burget and Bennewitz [BB15] extended the Inverse Reachability Map (IRM) concept from [VAD13] to biped humanoids, taking into account constraints like stability and kinematic loop closure. Yang et al. [Yan+16a] addressed the lack of collision avoidance in the previous methods with their inverse Dynamic Reachability Map (iDRM), enabling efficient collision-free whole-body end-pose selection in complex and dynamic environments.

Despite all these advancements concerning the automation of whole-body end-pose planning, the existing methods do not support obstacle avoidance in changing environments for stances on uneven or inclined terrains. End-pose planning on such terrains is still very complicated due to the curse of dimensionality and limited workspace coverage.

Research in this field is motivated by the possibility of one day having a fully fledged suite to control, teleoperate, and coordinate groups of robots. Such suite would allow to do optimal whole-body end-pose and motion planning in very short times. In turn, this would enable researchers with higher level reasoning on more complex problems.

1.3 Goals

By undertaking this research, we focus on solving the key issues of whole-body end-pose planning in order to improve robot autonomy in complex environments. Furthermore, we boost the quality of the results of existing methods and extend humanoid capabilities to more constrained environments which require better and more accurate plans. In addition to our proposed framework, we use existing methods that can already efficiently plan footsteps and whole-body motion to demonstrate the practicality and full integration of our planning methodology.

The main goals regarding the scope of this dissertation can be elaborated as follows:

$\mathcal{G}_1$. Extend previous work concerning end-pose planning to take into account flat supports at different heights;

$\mathcal{G}_2$. Further extend the whole-body end-pose planning framework resultant from $\mathcal{G}_1$ in order to support sloped terrains;

$\mathcal{G}_3$. Evaluate the methodologies proposed to achieve $\mathcal{G}_1$ and $\mathcal{G}_2$ with simulation benchmarks, and conduct test trials in the laboratory's test bed.

1.4 Publications

The work and ideas covered in this dissertation have been used in the writing of scientific publications. Most of the material in Chapter 3 has been submitted and accepted to the IEEE Robotics and Automation Letters (RA-L) journal [Yan+17]. Chapter 4 has been submitted to the Portuguese Conference on Artificial Intelligence (EPIA) in the form of an extended abstract [Fer+17].

- H. Ferrolho, V. Ivan, Y. Yang, W. Merkt, R. J. F. Rossetti, and S.Vijayakumar. Whole-Body End-Pose Planning on Uneven and Inclined Surfaces (extended abstract). In *Portuguese Conference on Artificial Intelligence* (EPIA), 2017. Under review.
- Y. Yang, W. Merkt, H. Ferrolho, V. Ivan, and S.Vijayakumar. Efficient Humanoid Motion Planning on Uneven Terrain Using Paired Forward-Inverse Dynamic Reachability Maps. In *IEEE Robotics and Automation Letters* (RA-L), 2017. In Press.

1.5 Document Structure

The remainder of this document is structured as follows:

- Chapter 2, "Literature Review" (p. 5), provides a literature review covering topics related to motion planning and end-pose planning in robotics. We conclude the chapter by going through the challenges currently being tackled in these areas.

- Chapter 3, "End-Pose Planning on Flat Surfaces at Different Heights" (p. 21), explains in detail our approach to solve the problem of end-pose planning on flat surfaces at different heights.
- Chapter 4, "Extending End-Pose Planning to Inclined Surfaces" (p. 37), addresses our end-pose planning approach in order to consider inclined supports.
- Chapter 5, "Conclusion" (p. 51), concludes this dissertation with an overview of the results accomplished resorting to the end-pose planning methods we propose. This chapter also covers a review of the author's contributions throughout the elaboration of this dissertation. Furthermore, a suggestion of future work to improve and build upon the contributions of our work is given at the very end of this chapter.

Chapter 2

Literature Review

This chapter provides definitions and examples for the basic concepts and methods involved in the problems of whole-body end-pose and motion planning. We explore the evolution and improvements of the methods associated with end-pose planning during the last few years. We also review the state of the art, *i.e.* the best approaches that currently exist. Finally, we look for the problems with the existing methods and niches in the literature that we intend to explore during this dissertation.

2.1 Motion Planning

The creation of autonomous robots[1] raises many problems in the field of Robotics. People usually tend to take for granted most of the motor skills that they have inherently acquired over the years since they were born. In fact, humans are quite good at learning how to abstract and control their limbs without consciously thinking about it since a very young age, and maybe that is the reason why, at first, it seems simple to solve motion planning problems with robots. Nonetheless, motion planning turns out to be an extremely difficult task, making it one of the most important and difficult problems, as well as a fundamental research area in Robotics.

Latombe [Lat12] loosely states the problem of motion planning as follows:

> "How can a robot decide what motions to perform in order to achieve goal arrangements of physical objects?"

[1] An autonomous robot is a robot that performs behaviours or tasks with a high degree of autonomy [Wika].

It is important to note that, contrary to popular belief, the problem of motion planning is not just limited to some sort of collision checking or obstacle avoidance [CS12]. In addition, it also takes into account the planning of collision-free motions in complex environments where objects with irregular geometries can be moving, the coordination of multiple robots that share the same workspace, etc. Temporal, geometrical and physical constraints are all inherent to the problem of motion planning. Given the configuration of an arbitrary robot pose and a desired end-pose, motion planning focuses ultimately on finding a plan and executing the movements required to go from the first pose to the desired pose according to that plan.

End-pose planning in humanoid robotics is the process of finding valid stance locations and balancing collision-free reaching configurations. Once a valid end-pose has been calculated, it can be used as a goal state for other planners, such as walking and motion planners. However, finding a valid end-pose to pass as an input to a motion planning solver is by itself another non-trivial problem. In fact, the more complex the environment, the more difficult the problem of end-pose planning is. Given that motion planning requires a valid end-pose as input, it is paramount to address the problem of end-pose planning. We will cover this topic in the Section 2.2 (p. 10).

There have been various different methods to try to approach motion planning: Fuzzy Logic Control [BC95], Genetic Algorithms [Ger99], Simulated Annealing [Mae+10], and others. Nonetheless, all these approaches have their limitations and often fall into local minima.

A different approach that has been more successful than the ones mentioned previously is Sampling Based Planning (SBP). Elbanhawi and Simic did an extensive review on the subject [ES14]. SBP is done by sampling the Configuration space (C-space), *i.e.* the space of all the valid transformations that can be applied to a robot. This pseudorandom approach provides very fast solutions even for difficult problems, in which the previously mentioned methods often fail. The trade-off of this approach is that the solutions are non-optimal. Notwithstanding, SBP has been extensively adopted [KL94; AW96; Kav+96; LaV98; KL00; LK01]. The reason for this is that in most cases what truly matters is to complete a task as fast as possible in terms of both planning- and execution-wise. Considering this, in many practical scenarios, non-optimal or redundant motions of a SBP solution are irrelevant given that the solutions are obtained much faster than using other methods, and that planning and execution times put together are still faster than the time of the planning stage alone from one of the other methods. This makes SBP extremely attractive for time-critical industrial applications.

The main algorithms used for SBP are the so-called Probabilistic Roadmap (PRM) and Rapidly-exploring Random Tree (RRT) [Kav+96; LaV98]. For the remainder of this chapter, we will explain RRT, RRT-Connect, and PRM. But some concepts need to be introduced beforehand.

2.1.1 Common Concepts

The C-space can be divided into two different regions: the free space, C_{free}, and obstacle space, C_{obs}. This notation prevents the need to explicitly define obstacles.

The robot is represented at any instance by a configuration q. This configuration is a list with length equal to the dimensions of the C-space. Motion planners require as inputs a start and goal configurations, q_{start} and q_{goal}, respectively.

Given two configurations q_a and q_b, a meaningful *metric* can be defined either as a value, or as a cost function, that represents the effort required to transition from one configuration to the next, *i.e.* from q_a to q_b.

A search algorithm known as Nearest Neighbour (NN) uses the previously mentioned metric to find the closest node (from a set of nodes) to a given configuration.

2.1.2 RRT

A Rapidly-exploring Random Tree [LK01] is a probabilistically complete single-query planner. The algorithm works as follows: (1) the search begins from an initial configuration, q_{start}; (2) a random configuration, q_{new}, is selected from the C-space; (3) if this random configuration belongs to C_{obs}, *i.e.* $q_{new} \in C_{obs}$, it means the sampled configuration is in collision, and consequently it is discarded. Otherwise, a Nearest Neighbour search is performed to find the closest sample to q_{new}, *i.e.* q_{near}; (4) these two nodes, q_{new} and q_{near}, are then connected to each other using a local motion planner; q_{new} is discarded if it cannot be reached; (5) a final step is performed to guarantee that the motion responsible for the transitioning between q_{new} and q_{near} is collision free. If the motion is indeed collision free, q_{new} is added to the exploration tree; otherwise, the sample is discarded.

RRT terminates once one of the following conditions is met: (i) the goal configuration, q_{goal}, matches q_{new}, (ii) a specified number of iterations is exceeded, or (iii) a given time period is exceeded.

2.1.3 RRT-Connect

RRT-Connect is based on RRT with a Connect heuristic for single-query path planning problems in high-dimensional C-spaces [KL00]. The method works by incrementally building two RRTs rooted at different configurations: one at the start configuration, q_{start}, and the other at the goal configuration, q_{goal}. The trees each explore space around them and also advance towards each other through the use of a simple greedy heuristic. This heuristic allows rapid convergence to a solution. The combination of the greedy heuristic with the rapid and uniform exploration properties of RRTs avoids the pitfalls of local minima.

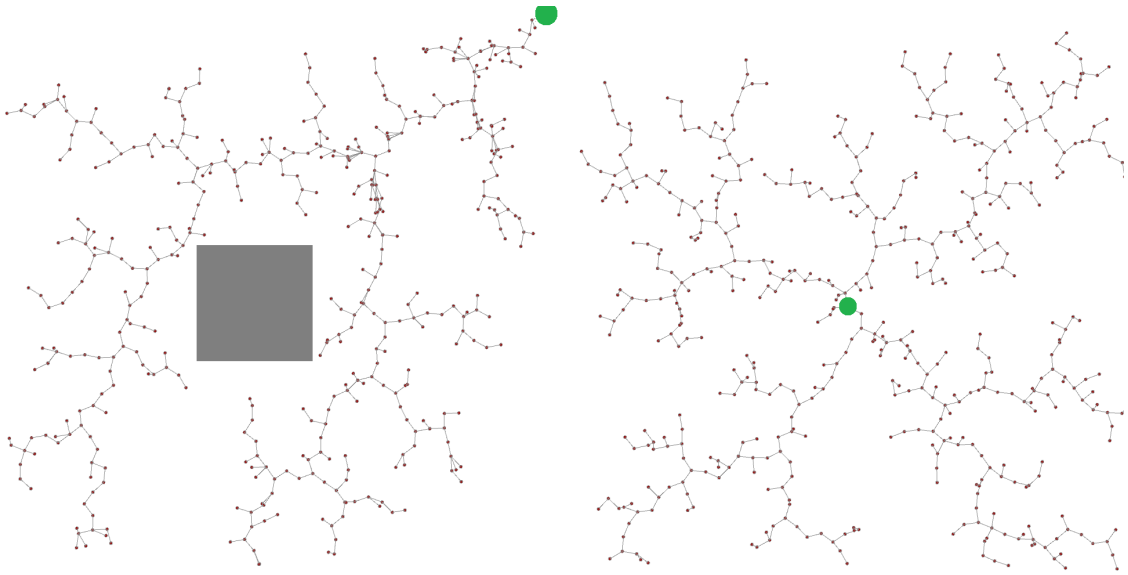


Figure 2.1: RRT exploring an environment with one obstacle (left) and a free space (right) after 500 iterations. The root of each tree is shown as a green bold circle in both cases — on the top right corner (left) and centre (right). Reprinted from [ES14]. Copyright © 2014 by the IEEE.

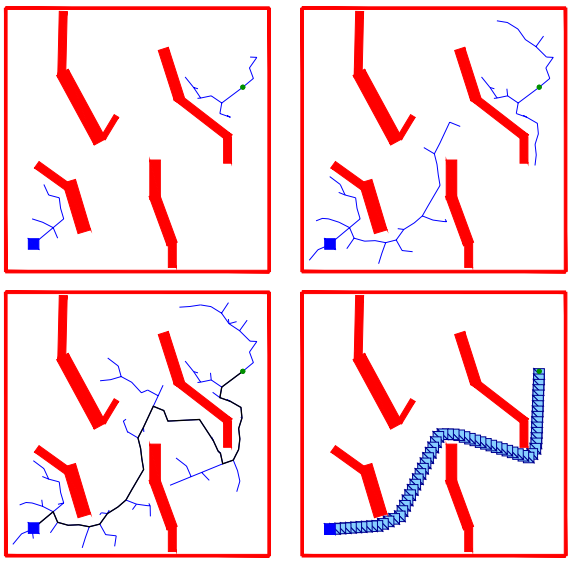


Figure 2.2: RRT-Connect growing two trees towards each other. Reprinted from [KL00]. Copyright © 2000 by the IEEE.

2.1.4 PRM

Probabilistic Roadmap [KL94; AW96; Kav+96] is a probabilistically complete [Bar+97; HLK06] multi-query planner comprised of two main procedures: a *learning* phase, and a *query* phase.

Firstly, the C-space is sampled for a certain amount of time during the learning phase. The samples in the free space, C_{free}, are maintained, whereas the samples in the obstacle space, C_{obs}, are discarded. Figure 2.3 shows an example of a roadmap built by PRM.

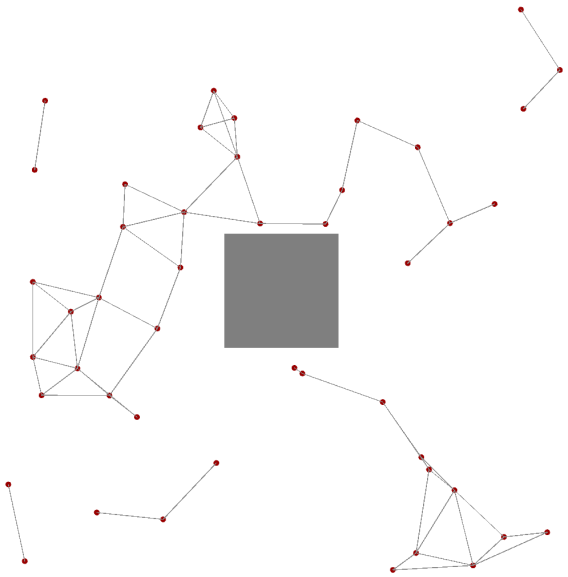


Figure 2.3: Roadmap built in the PRM learning phase. Reprinted from [ES14]. Copyright © 2014 by the IEEE.

Afterwards, during the query phase, the start and goal configurations are defined and connected to the roadmap.

Because PRM maintains the roadmap of the learning phase, and only connects the start and goal configurations during the query phase, it is able to actually solve different queries for the same environment. Planning time is invested in sampling and generating a roadmap so that queries are solved quickly. For this reason, it is referred to as a multi-query planner.

The learning phase is described as follows: (1) a random node, q_{rand}, is sampled from the C-space; (2) if q_{rand} belongs to C_{obs} it is discarded; otherwise, it is added to the roadmap; (3) afterwards, an attempt to connect q_{rand} to its neighbouring nodes — *i.e.* all the nodes within a specific range to q_{rand} — is carried out using a local planner. Each edge connecting two nodes represents a motion plan to transition from one configuration to the other; (4) each connection is then tested for a collision check, and edges associated to motions in collision are disconnected. These steps are repeated until a specific number of nodes have been sampled.

2.1.5 Humanoid Motion Planning

Kuffner et al. [Kuf+01] and Cognetti et al. [CMO15] have shown that humanoid whole-body motion plans can be generated using custom planning algorithms. A different approach consists of dissociating locomotion from upper-body manipulation [Yan+16a]. This separation makes whole-body motion tasks both more simple and practical for complex humanoid robots such as NASA's Valkyrie and Boston Dynamics' Atlas. A complex planning task can therefore be decomposed into three stages:

1. *End-pose planning*: find an appropriate pre-grasp stance location and grasping configuration,
$$\mathbf{p}^*, \mathbf{q}^* = \mathit{EndPosePlan}(\mathbf{p}_s, \mathbf{q}_s, \mathbf{y}^*)$$

2. *Footstep planning and execution*: plan and execute a sequence of footsteps to walk to the pre-grasp stance location,
$$\mathbf{p}_{[0:T]} = \mathit{FootstepPlan}(\mathbf{p}_s, \mathbf{p}^*)$$

3. *Motion planning and execution*: plan and execute a full-body collision-free motion to complete the task,
$$\mathbf{q}_{[0:T]} = \mathit{MotionPlan}(\mathbf{q}^*)$$

... where $\mathbf{p}_s$ and $\mathbf{q}_s$ are the current stance location and robot configuration, and $\mathbf{y}^* = \{\mathbf{y}^*_{lhand}, \mathbf{y}^*_{rhand}\} \in 2 \times SE(3)$ are the desired poses for the left and right hands. An end-pose contains the desired stance location $\mathbf{p}^* = \{\mathbf{p}^*_{lfoot}, \mathbf{p}^*_{rfoot}\} \in 2 \times SE(3)$ and reaching configuration $\mathbf{q}^* \in \mathbb{R}^N$, which can be used as a goal configuration in the motion planning module.

2.2 End-Pose Planning

Addressing the problem of end-pose planning is not trivial, especially on humanoid robots with many DoF. The task becomes even more difficult when complex environments such as staircases or inclined terrains have to be taken into consideration. Traditional Inverse Kinematics (IK) solvers take too long or fail to solve the constraints of such specific problems. In addition, when inclined terrains are considered, the friction of the support surfaces have to be considered, therefore changing the dynamics of the entire system.

The next sections present an in-depth review of the current progress concerning end-pose planning.

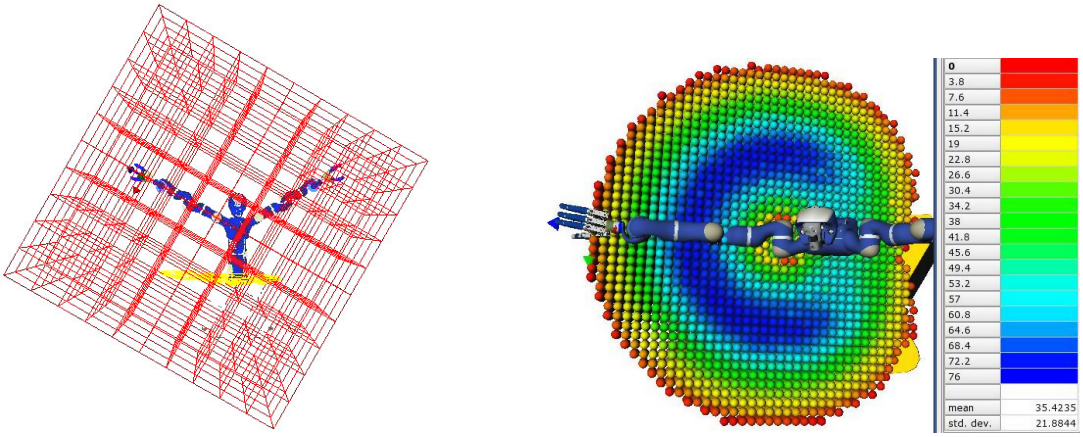
2.2.1 Reachability Map

Humans have developed the ability to unconsciously estimate how far they can reach, and in how many ways they can grasp differently shaped objects. Humanoid robots have arms

with very specific motion constraints: a robot arm can only bend in certain points, and can only reach as far as its structure allows to. Therefore, it is desirable to know in what ways a robot can interact with its workspace, *i.e.* which regions of its surroundings are reachable and in how many different ways, very similarly to what humans do. In robotics, this concept is called the *capability map* [ZBH07] or Reachability Map.

In 2007, Zacharias et al. proposed for the very first time the so-called *capability map*: a representation of kinematic reachability and directional information for the whole Cartesian workspace of a robot arm [ZBH07]. The motivation was to have a visualisation tool for the directional preferences and existing structures in the redundant workspace of the arms of their robot Justin [Ott+06]. In their paper, Zacharias et al. describe the capability map approach by splitting it into the following processes: (i) discretisation, (ii) random sampling, and (iii) analysis and optimisation.

The first step is the discretisation of the workspace of the robot arm. The workspace can be encapsulated by a cube centred at the base of the arm. The size of the cube should be the smallest possible, whilst containing all the possible reaching positions of the arm. A cube that meets such requirements is one with side length of two (2) arm lengths. The result is an overestimation of the maximum workspace of the arm. Afterwards, the cube is subdivided into smaller cubes, also known as voxels (see Figure 2.4a). This discretisation process enables the analysis of specific regions of the workspace of the robot in planning tasks.



(a) The maximum workspace of the right arm is overestimated by the enveloping cube which is further subdivided into smaller cubes.

(b) Reachability spheres across the workspace. The workspace representation was cut in half for better visibility of the structure.

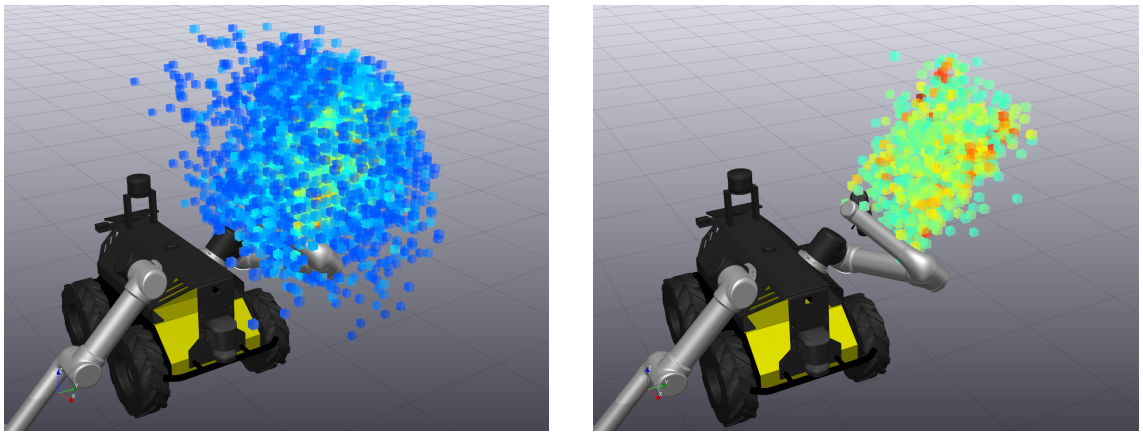
Figure 2.4: Visualisation of information concerning the capability map, as described by Zacharias et al. Reprinted from [ZBH07]. Copyright © 2007 by the IEEE.

After the workspace has been discretised, the C-space of the robot arm is randomly sampled according to a uniform distribution. The position of the Tool Centre Point (TCP) is then calculated using Forward Kinematics (FK) for each of the sampled configurations, and mapped onto the corresponding voxel.

As the authors mention in their paper, one could use the number of random samples mapped onto the same voxel as a measure of reachability for the region covered by that voxel. But that would be misleading: when a configuration corresponds to a robot kinematic singularity, even large changes in link joints result in very small motions in Cartesian workspace. Therefore, a large amount of random samples in singularity might be mapped onto the same voxels, invalidating this as a good measure for workspace reachability.

In order to measure the reachability of a given voxel, Zacharias et al. used what they called *reachability spheres*. They inscribed a sphere in each of the voxels of the workspace, and for each sphere they spawned N points equally distributed on its surface. Moreover, they associated a TCP frame to each of the points, with the z-axis pointing towards the centre of the sphere. Additionally, they rotated the TCP frame of each point around the z-axis according to a fixed step size and, for each resulting frame, they used an IK solver to look for a valid solution. If a solution is found for one of the rotations, they mark that point of the sphere surface in the workspace map. In other words, the method scans the discretised workspace and for each of the reachability spheres checks if it can be grasped from different directions from at least one rotation.

The goal of these reachability spheres is to measure the *reachability index* of each voxel in the workspace of the robot arm. The number of marked points of one of the spheres over the total number of points N on the sphere surface gives the percentage of how reachable the voxel containing that sphere is, *i.e.* the *reachability index*. Afterwards, colouring the voxels of the workspace according to their reachability index provides a good overview of the reachable regions of the arm of the robot (see Figure 2.4b, p. 11). The concept of RMs introduced by Zacharias et al. was the very first method to actually encode the information of *what is the reachable space of a robot when its base is fixed.*



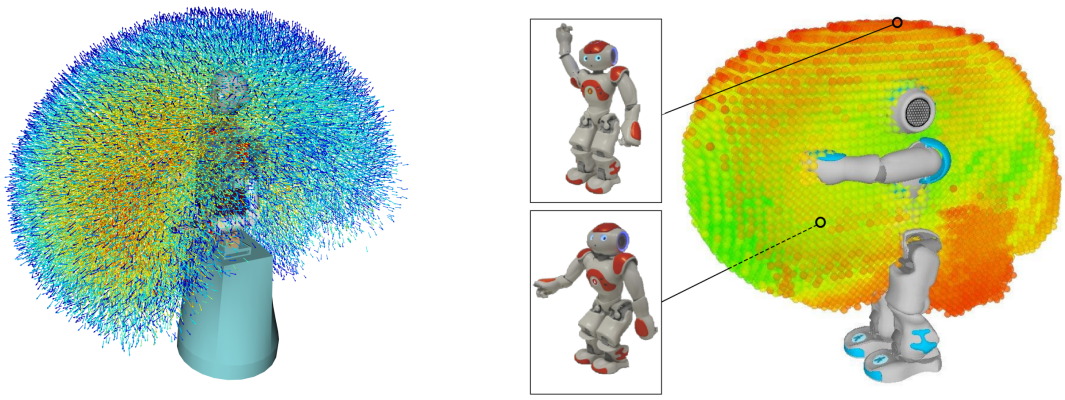
(a) Entire Reachability Map.

(b) Filtered Reachability Map, showing highly manipulable areas.

Figure 2.5: Reachability Map of the left arm of a dual arm Husky robot. Copyright © 2017 by Wolfgang Merkt.

Reachability Maps have some nuisances. To start with, they do not account for collisions with surrounding obstacles. Moreover, the discretisation of the workspace will always have an associated trade-off between resolution and memory: the higher the resolution, *i.e.* the smaller the voxels used to discretise the workspace, the more memory will be required to store the underlying data structure. Finally, the metric used to classify the reachability of the voxels needs to be chosen carefully — the *reachability index* used by Zacharias et al. is an example of a good metric.

After Zacharias et al. published their work, many other researchers have used similar representations to compute the RMs for other robots (see Figure 2.5, p. 12). Furthermore, others have even built other concepts and algorithms on top of RMs (see Figure 2.6). We will take a closer look into some of them in the following sections.



(a) The reachability distribution of the right Tool Centre Point of an ARMAR-III [Asf+06]. Reprinted from [VAD13]. Copyright © 2013 by the IEEE.

(b) Representation of reachable right hand locations from statically stable double support poses of a NAO humanoid. Reprinted from [BB15]. Copyright © 2015 by the IEEE.

Figure 2.6: Vahrenkamp et al. and Burget and Bennewitz used RMs as a starting point for their work in [VAD13] and [BB15], respectively.

2.2.2 Inverse Reachability Map

As explained in the previous subsection, the RM is a very good method to construct a data structure that encodes *where and in what ways a fixed-base robot can reach its workspace.* Notice that RMs use the base of the robot as the frame of reference.

Vahrenkamp et al. [VAD13] extended the potential of RMs by introducing the so-called IRM. They proposed to invert the transformation associated to each entry of the RM, from base-to-TCP to TCP-to-base. This results in a data structure that encodes the opposite information of RM, *i.e. given a grasping target, where can the robot base be placed.*

Zacharias et al. used the previously mentioned *reachability index* as a performance metric for each of the poses contained in their RM. Besides reachability information, manipulability information can also be used as a performance indicator.

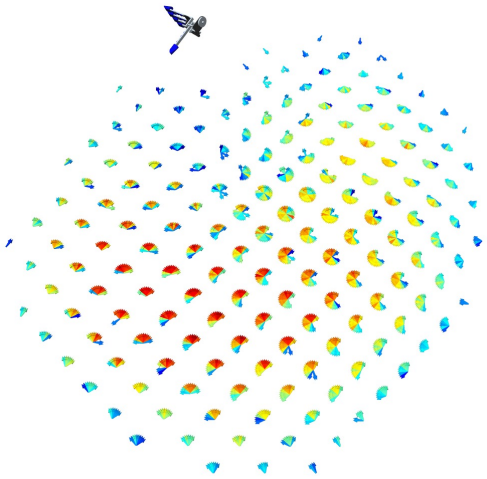


Figure 2.7: The reachability inversion for a given grasp visualised on the ground plane. The potential platform orientations are depicted by arrows which are coloured according to their inverse reachability (red: high, blue: low). Reprinted from [VAD13]. Copyright © 2013 by the IEEE.

For their IRM, Vahrenkamp et al. used an extended manipulability measurement. This measurement consisted of the classical manipulability index [Yos85; Tog86], and a penalty for limited configurations due to the distance between body parts of the robot.

The process used to build the actual IRM can be described as follows: (i) a pass through all the voxels v_i of the RM occurs to build the tuples $(t_i,\ e_i)$, where $t_i \in SE(3)$ is the base-to-TCP transformation and e_i is the corresponding voxel entry; and (ii) the first element of each tuple is inverted from t_i to t_i^{-1}, *i.e.* from a base-to-TCP to a TCP-to-base transformation. The resulting tuples constitute the inverted RM.

The authors' proposed IRM as a method to efficiently find suitable base positions for a mobile base robot given a certain grasping target. Their method, however, did not support bipedal robots.

Burget and Bennewitz [BB15] carried on Vahrenkamp et al. work by extending the IRM to humanoid robots. Their approach takes into account kinematic loop closure and stability constraints, characteristic of humanoid robots. They used a NAO robot to evaluate their approach. In order to build a RM, they start by sampling the C-space of the robot. They split the kinematic structure of the humanoid in two: upper body and lower body. For the sampling they chose to iteratively step through the joint values with a certain increment: 0.3 rad for the upper body joints and 0.2 rad for the lower body joints. For the lower body, they define one of the legs as the *support leg*, and the other as the *swing leg*. Then, they take the chain configuration of the support leg and calculate the FK to get the pose of the hip with respect to the support foot. Afterwards, given the hip pose and the support foot pose, they use an IK solver to look for a valid configuration for the desired pose of the swing leg. If a solution is found, they determine if the whole-body pose is statically

balanced and collision-free. If the pose is correct, it is added to the RM. If the pose is not valid, or no IK solution is found, the algorithm tries to sample a new configuration, thus restarting the whole process. Each of the sampled configurations are mapped onto the voxel containing the spatial location of the end effector for that pose. Moreover, each configuration is evaluated with a manipulability measure, according to the manoeuvrability of the end effector in workspace. Finally, with the RM as input, they generate the IRM by iterating through all the voxels and, for each of the configurations stored in them, inverting the end-effector transformation.

With their extended approach, Burget and Bennewitz were able to apply IRM to humanoid robots. However, their method does not take into account collisions with obstacles in the surroundings of the robot. In addition, the lower body configurations are very limited, with the feet positioned parallel relative to each other at all times. Finally, the method assumed the robot was always standing on a flat surface, therefore lacking support for uneven terrains.

2.2.3 iDRM: inverse Dynamic Reachability Map

Yang et al. [Yan+16a] carried on the existing work discussed in the previous sections, further improving end-pose selection with their iDRM. iDRM can find valid end-poses for humanoid robots in complex and dynamic environments[2] by making use of a custom robot-to-workspace occupation list and an online stage to efficiently update the maps and filter end-poses in collision. After a valid end-pose is obtained with iDRM, it can be given as input to footstep and motion planners. The system then generates a walking plan to move the robot to the desired standing location. Finally, it generates a collision-free motion to grasp the goal or goals with stationary feet.

iDRM consists of two main stages: an *offline stage* in which the iDRM is constructed, and an *online stage* in which a valid end-pose is computed according to the environment.

Offline stage

During this stage, the workspace is discretised and represented by a data structure consisting of a set of voxels. Each voxel stores: (i) a list of indices of the robot samples which feet are placed in this very same voxel; and (ii) a so-called *occupation list*, which is a list of indices of all the robot samples that intersect with this voxel in any way. Once this data structure is set up and ready, the next step is to sample whole-body quasi-statically balanced configurations.

$$\mathbf{q}^* = IK(\mathbf{q}_{seed}, \mathbf{q}_{nom}, \mathbf{C})$$

[2]By **complex** environments we mean cluttered environments, with obstacles lying around; and by **dynamic** we mean scenarios where the obstacles and surroundings are prone to change and move around.

In order to do so, a seed pose, $\mathbf{q}_{seed}$, is required as an initial value for the first iteration of a whole-body non-linear optimisation-based IK solver [TD]. This IK solver will then attempt to satisfy all the constraints in the constrain set $\mathbf{C}$ by using a Sequential Quadratic Programming (SQP) solver in the form of:

$$\begin{aligned} \mathbf{q}^* = \arg \min_{\mathbf{q} \in \mathbb{R}^{N+6}} & \|\mathbf{q} - \mathbf{q}_s\|^2_{Q_q} \\ \text{subject to} \quad & \mathbf{b}_l \leq \mathbf{q} \leq \mathbf{b}_u \\ & c_i(\mathbf{q}) \leq 0, \ c_i \in \mathbf{C} \end{aligned} \tag{2.1}$$

... where $Q_q \succeq 0$ is the weighting matrix, $\mathbf{b}_l$ and $\mathbf{b}_u$ are the lower and upper joint bounds, and $\mathbf{C}$ is the constraints set. If successful, it returns the resultant configuration $\mathbf{q}^*$ that satisfies all the specified constraints. The only postures stored in the dataset are the returned poses which are in (i) quasi-static balance, (ii) self-collision-free, and (iii) that reach a region of interest in the front of the robot. In order to entirely explore the robot's reaching capabilities, the authors used the method presented by Kuffner [Kuf04] to uniformly constrain end-effector orientation with the IK solver. The sampling process is repeated with random seed samples until the target number of samples in the dataset are generated.

Afterwards, inverting the end-effector frame results in the stance frame expressed in the end-effector frame of reference:

$$T_n^{stance,eff} = \left(T_n^{eff,world}\right)^{-1} \times T_n^{stance,world} \tag{2.2}$$

Once the offline process finishes, the C-space of the robot has been uniformly explored, and each of the configurations stored in the map according to the voxel which contained the location of that configuration's end-effector, and the occupation lists updated accordingly, as shown in Figure 2.8.

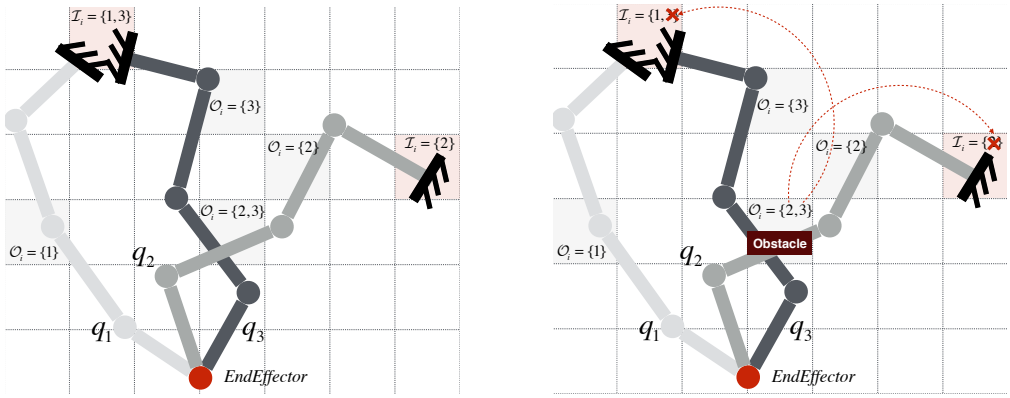


Figure 2.8: iDRM collision update illustration in 2D. The left and right figures show the original iDRM in free space and the updated iDRM respectively. A key feature of iDRM is that updating an occupation list $\mathcal{O}$ affects the reach list $\mathcal{I}$. Reprinted from [Yan+16a]. Copyright © 2016 by the IEEE.

The construction time of iDRM relies upon the chosen voxel resolution and the number of samples to be generated. Nonetheless, the construction time is not very critical considering in needs to be performed offline only once.

Online stage

The selection of valid end-poses occurs during the online stage. The offline stage needs to be completed beforehand.

The first step is to find collision-free samples. In order to do this, iDRM needs to be transformed to the desired end-effector pose in the world frame of reference. This process involves transforming all the voxels, and thus it is indicated to scenarios where there are more obstacles rather than voxels in the discretised workspace. On the contrary, if there are less obstacles than there are voxels — which is usually the case — one can transform the obstacles' frame of reference into the iDRM's frame instead of transforming all the voxels of the workspace. Choosing the appropriate transformation according to the obstacles:voxels ratio will speed up the online update process significantly.

Following, invalid samples, *i.e.* samples that are in collision, are removed. The obstacles in the environment are matched to iDRM's discretised set of voxels, and the samples associated to the ids contained in occupation lists of voxels intercepting obstacles are discarded — see Algorithm 1.

Algorithm 1 Collision update

Require: $\mathbf{y}^*$, Env
Ensure: Q_{free}
1: **if** $size(V) > size(Env)$ **then**
2: $\quad \bar{V} \leftarrow \mathbf{y}^* \times V$
3: $\quad V_{occup} \leftarrow CollisionCheck(\bar{V}, Env)$
4: **else**
5: $\quad \overline{Env} \leftarrow (\mathbf{y}^*)^{-1} \times Env$
6: $\quad V_{occup} \leftarrow CollisionCheck(V, \overline{Env})$
7: **for** $i \in V_{occup}$ **do**
8: $\quad$ **for** $o \in O_i$ **do**
9: $\quad\quad q_o.valid \leftarrow false$
10: $Q_{free} \leftarrow \emptyset$
11: **for** $q_n \in \mathcal{Q}$ **do**
12: $\quad$ **if** $q_n.valid \leftarrow true$ **then**
13: $\quad\quad Q_{free} \leftarrow Q_{free} \cup n$
14: **return** Q_{free}

After this collision update, the filtered iDRM map is a subset of the original, containing only collision-free samples — as show in Figure 2.9 (p. 18). The voxels are coloured according to the amount of collision-free states contained in these very same voxels' reach

lists. Greener voxels contain more samples than red voxels. It is guaranteed that each coloured voxel has at least one collision-free state. The figures located in the middle and at the right show how obstacles affect the set of voxels occupied by obstacles.

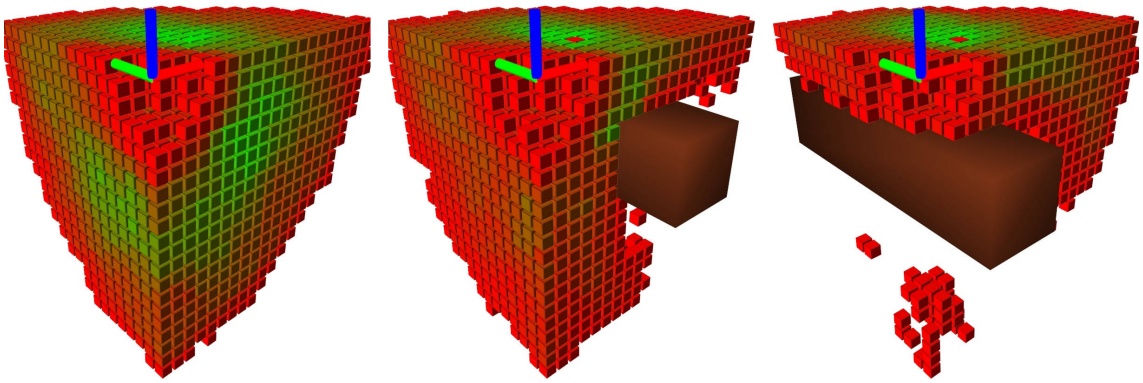
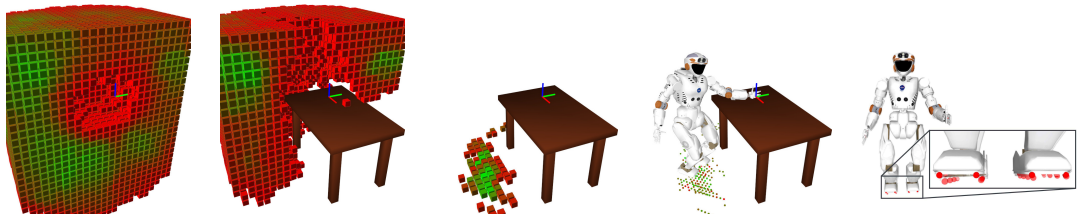


Figure 2.9: Octant view of an iDRM collision update. The axis is the origin of the iDRM, *i.e.* the end-effector pose. Reprinted from [Yan+16a]. Copyright © 2016 by the IEEE.

Once the iDRM map is filtered and contains collision-free samples only, the next step is to check whether those poses are actually feasible samples or not, *i.e.* whether they are physically balanced. Here it is assumed the robot needs only to stand on flat ground with horizontal feet orientation, *i.e.* *roll* and *pitch* of the feet transformation are zero.

Instead of iterating through all the samples which are still being considered as candidate end-poses at this time, the authors discard all the samples whose feet fall into a voxel that does not intersect the ground plane. Some small variations are tolerated in each axis, since there may not exist samples with the exact horizontal feet orientation. These small variations are corrected during a last step for minor posture adjustments. This entire process is depicted in Figure 2.10. The axis widget in the figure represents the desired end-effector pose in the world frame of reference.



(a) Original iDRM (b) Collision update (c) Feasibility update (d) Candidate end-pose (e) End-pose adjustment

Figure 2.10: (a) – (d): iDRM end-pose planning example. The iDRM is transformed into world frame, and the axis indicates desired end-effector pose in the world frame. (e) highlights the final IK adjustment, where the shadowed posture is the candidate $\mathbf{q}_{n^*}$ and the solid one is the final end-pose result $\mathbf{q}^*$. Reprinted from [Yan+16a]. Copyright © 2016 by the IEEE.

Moreover, a list of candidate end-poses is extracted from the samples still being considered — see Algorithm 2.

Algorithm 2 Feasibility update

Require: Q_{free}
Ensure: $Q_{feasible}$
$V_{ground} \leftarrow \emptyset$
for $v_i \in V$ **do**
  if v_i intersects with ground **then**
    $V_{ground} \leftarrow V_{ground} \cup i$
$Q_{feasible} \leftarrow \emptyset$
for $i \in V_{ground}$ **do**
  for $n \in \mathcal{I}_i$ **do**
    if $n \in Q_{free}$ **then**
      $\bar{T}_n \leftarrow \mathbf{y}^* \times T_n^{feet}$
      if $z(\bar{T}_n) < \epsilon_z$ AND $roll(\bar{T}_n) < \epsilon_{roll}$
    AND $pitch(\bar{T}_n) < \epsilon_{pitch}$ **then**
        $Q_{feasible} \leftarrow Q_{feasible} \cup n$
return $Q_{feasible}$

Finally, the last step is to choose one solution from the resultant list of candidate poses. At this point, all the candidates in the list of feasible samples are valid. This is therefore a matter of picking the best candidate sample. Yang et al. scored the samples according to a Jacobian based measure to evaluate the end-effector's manipulability — similarly to [BB15]:

$$g_n = \sqrt{\det J(q_n) J(q_n)^T}, \tag{2.3}$$

...where $J(q_n)$ is the Jacobian matrix of q_n and all the scores are calculated by offline sampling, thus being readily available here. The index of the best candidate pose can therefore be found as:

$$n^* = \underset{n \in Q_{Feasible}}{\arg\max}\ w_m g_n - \|q_n - q_0\|_W, \tag{2.4}$$

...where w_m and W are constant weighting factors, and $\|q_n - q_0\|_W$ is another cost term introduced by the authors to penalise samples far away from their initial configuration. Figure 2.10d (p. 18) shows all the feasible stance locations coloured based on the posture manipulability scores. The visible whole-body robot configuration is the best sample for that specific scenario.

As previously mentioned, the stance pose of the best configuration might not be perfectly aligned with the ground. A whole-body IK solver is used to perform the necessary final adjustments to the configuration. Most of the times these corrections will be very minor changes, required due to the given tolerances on the previous step. In the unlikely event of

the best candidate pose being in collision after the IK adjustment, or becoming unstable, the pose is discarded and the next best candidate will be selected until a valid solution is found.

The authors implemented their planning framework within the EXtensible Optimization Toolset (EXOTica) [Iva+], and used the Flexible Collision Library (FCL) [PCM12] to create the occupation lists and online collision checking queries. Table 2.1 shows the computational times of different components in humanoid motion planning and end-pose planning.

Table 2.1: Computational time of different components in humanoid motion planning (in seconds). The overall time is the sum of end–pose planning (**EP**) and motion planning (**MP**), while the footstep planning is not counted. Algorithms requiring no end-pose planning (marked as –) have a zero **EP** planning time. The planning is a failure (marked as ×) if no solution is found within 100 seconds. Reprinted from [Yan+16a]. Copyright © 2016 by the IEEE.

Algorithms		Easy Task			Medium Task			Hard Task		
EP	**MP**	**EP**	**MP**	**Overall**	**EP**	**MP**	**Overall**	**EP**	**MP**	**Overall**
–	E–space RRT	0	×	×	0	×	×	0	×	×
–	C–space RRT	0	×	×	0	×	×	0	×	×
–	E–space RRT–Connect	0	12.0974	12.0974	0	15.8324	15.8324	0	88.7171	88.7171
RP	C–space RRT–Connect	0.1916	1.5010	1.6926	1.2322	1.8052	3.0374	2.2654	3.2857	5.5511
R-DRM		0.7521		2.2531	2.3273		4.1325	38.8050		42.0907
IRM		0.0440		1.5450	0.9560		2.7612	2.2910		5.5767
IDRM		**0.0553**		**1.5563**	**0.0566**		**1.8618**	**0.0678**		**3.3535**

2.3 Conclusion

Motion planning is still a very challenging area where problems like (a) multi-contact motion planning with collision avoidance during pose transitioning and (b) kinodynamic planning for transitioning between multi-contact poses, amongst many others, are yet to be solved. However, and as we have stated previously, from this point forth our main focus will be on end-pose planning. The goal of the review on motion planning presented here is to contextualise the reader to the problem of end-pose planning.

With regard to end-pose planning, despite all the extensive work and research that has been put into it during the last couple of years, there is still much room for improvement, and many problems to be solved. Namely, end-pose planning on uneven terrains is still very complicated due to the curse of dimensionality and limited workspace coverage. We will be addressing this in Chapter 3 (p. 21) and Chapter 4 (p. 37).

Chapter 3

End-Pose Planning on Flat Surfaces at Different Heights

In this chapter we present a Paired Forward-Inverse Dynamic Reachability Maps approach that extends the iDRM by integrating it with forward reachability maps according to the inherent kinematic structure of the robot. By exploiting the combinatorics of this modularity, greater coverage in each map can be achieved while keeping a low number of stored samples. This enables us to draw samples from a much richer dataset to effectively plan end-poses for single-handed as well as bimanual tasks on uneven terrain.

We demonstrated the method on the 38-DoF NASA Valkyrie humanoid using the whole body to exploit redundancy for accomplishing manipulation tasks on uneven terrain while avoiding obstacles (see Figure 3.1, p. 22).

3.1 Dynamic Reachability Maps

The forward and inverse dynamic reachability maps, *i.e.* Dynamic Reachability Map (DRM) and iDRM, are the mappings from robot C-space to workspace with an efficient indexing technique that updates the collision status of millions of configurations in real-time. DRM and iDRM are defined with respect to the base frame and the end-effector frame respectively. In other words, DRM encodes information of *when fixing the base, what is the reachable space of the end-effector*, whereas iDRM encodes *to reach a desired pose, where to best place*

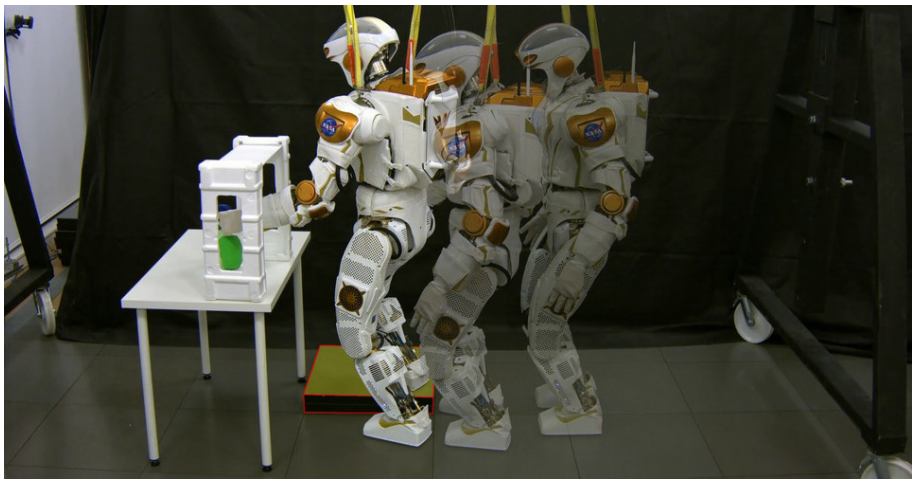


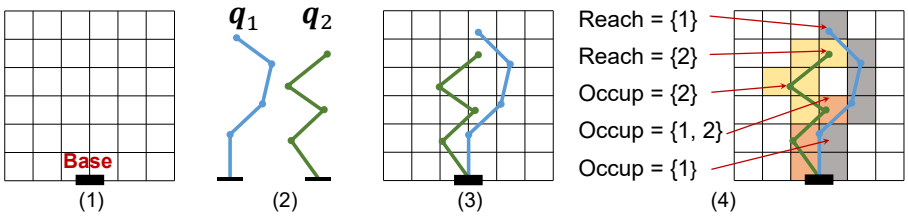
Figure 3.1: Motion planning during a grasping exercise on uneven terrain. The robot automatically chooses appropriate standing locations and grasping configurations on uneven terrains.

the base. However, from an algorithmic perspective, DRM and iDRM are very similar, and both of them have two stages: offline pre-processing and online planning.

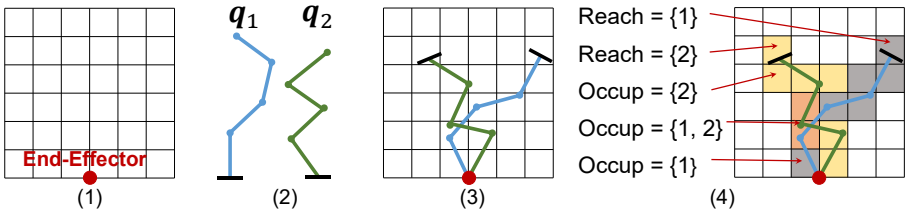
3.1.1 Offline Pre-Processing

The offline pre-processing phase contains four major steps for both DRM and iDRM, as highlighted in Figure 3.2 (p. 23). First, the workspace is discretised into a bounded 3D voxel grid $\mathbb{V}$. The grid of DRM is defined with respect to the base frame while the grid of iDRM has its origin in the end-effector frame. Throughout this dissertation, we use *root link* to refer to the reference link, *i.e.* the base link for DRM and the end-effector link for iDRM. Also, we use *tip link* to refer to the end-effector link for DRM and the base link for iDRM. Both DRM and iDRM can only have one root link but multiple tip links. Next, we generate N number of valid samples[1], which are then transformed to the origin of the corresponding map. The last step generates the reach list R_v and occupation list O_v for each grid voxel $v \in \mathbb{V}$. The reach list R_v stores the indices of samples whose tip link falls into this voxel v. For a robot with K tip links, the reach list stores a list of paired values specifying both sample and tip indices, *i.e.* $R_v = \{(n, k) \dots\}$, where $n \in N$ is the sample index and $k \in K$ is the tip index. Note that in Figure 3.2 (p. 23), we use a robot model with only one tip link for clarity. Finally, the occupation list O_v is generated storing the list of samples that intersect with voxel v.

[1] A valid sample has to satisfy a combination of robot's kinematic joint limits, be self-collision-free, balanced, etc.

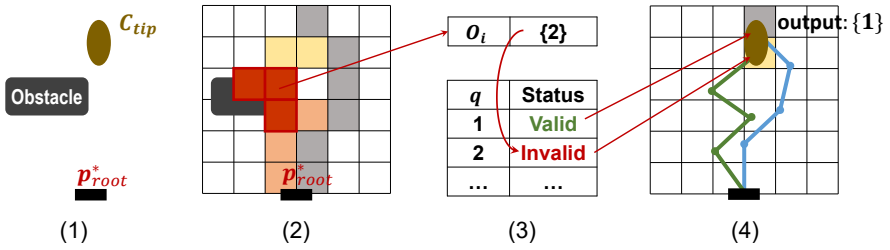


(a) Forward Dynamic Reachability Map (DRM)

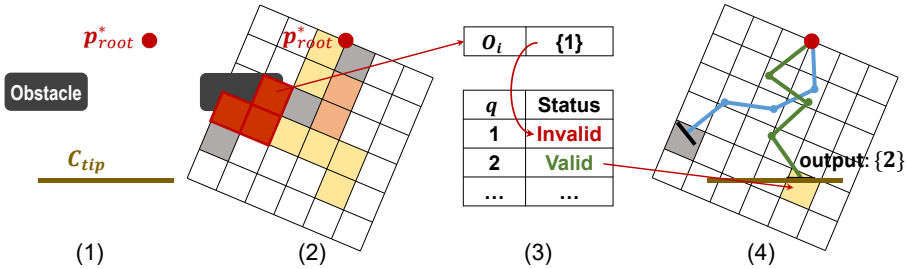


(b) Inverse Dynamic Reachability Map (iDRM)

Figure 3.2: Examples of DRM and iDRM offline map construction. From left to right: (1) discretised space, (2) generate valid samples, (3) transform samples to map origin, and (4) generate reach and occupation lists.



(a) Forward Dynamic Reachability Map (DRM)



(b) Inverse Dynamic Reachability Map (iDRM)

Figure 3.3: Examples of DRM and iDRM online updates. From left to right: 1) problem setup; 2) transform map to root pose; 3) validate collision status; and 4) check tip pose constraints and find valid samples.

3.1.2 Online Update

During the online update phase, our goal is to find samples that are collision-free and satisfy tip link constraints C_{tip} given the root pose $\mathbf{p}^*_{root}$, as highlighted in Figure 3.3 (p. 23). C_{tip} defines valid position and orientation regions for different tip links. Firstly, the DRM/iDRM map is transformed to $\mathbf{p}^*_{root}$. Conventional collision checking is then deployed to identify the colliding voxels, then iteratively invalidate samples in the occupation list O_v of all colliding voxels. Finally, we check the reach lists of candidate voxels to find valid samples that satisfy collision-free and C_{tip} so the output samples are guaranteed to be collision-free. For example, in Figure 3.3 (p. 23), two samples from the DRM satisfy the tip pose constraint, but only sample 1 was selected since the other sample was invalidated during the collision update step. In the iDRM case, sample 1 was excluded from the result as it was in collision and violated the tip pose constraint.

3.2 Whole-Body Kinematic Split

The iDRM can be used directly for humanoid end-pose planning with the constrained positions of two feet [Yan+16a], which is limited to flat ground only. As the iDRM can have multiple tip links, a direct and naïve approach is to create an iDRM with one root link and three tip links, where one hand is selected as the root and the rest three limbs are treated as tip links. However, this significantly increases the dimensionality of the problem, *i.e.* the number of samples has to increase exponentially with each tip link to cover the high dimensional space (see Section 3.4). Consequently, the required memory size is so large that it becomes infeasible to run on any commodity hardware.

To plan end-poses on uneven terrain while keeping a manageable number of samples and memory size, we take advantage of the robot's inherent structure to treat upper-body and lower-body separately. We separate the robot at the torso pelvis joint, as illustrated in Figure 3.4 (p. 25). We create an iDRM for the upper-body and a DRM for the lower-body. We choose one hand as the root of the upper-body iDRM, and the other will become a tip link. We could further split the kinematic structure to obtain more but smaller components, *i.e.* further split the upper-body into left and right arms. However, as we will show later in 3.4.3, the proposed splitting approach is more efficient considering the trade-off between success rate and planning time. In the rest of this section, we will discuss how to create the two maps, and combine them to plan end-poses on uneven terrains.

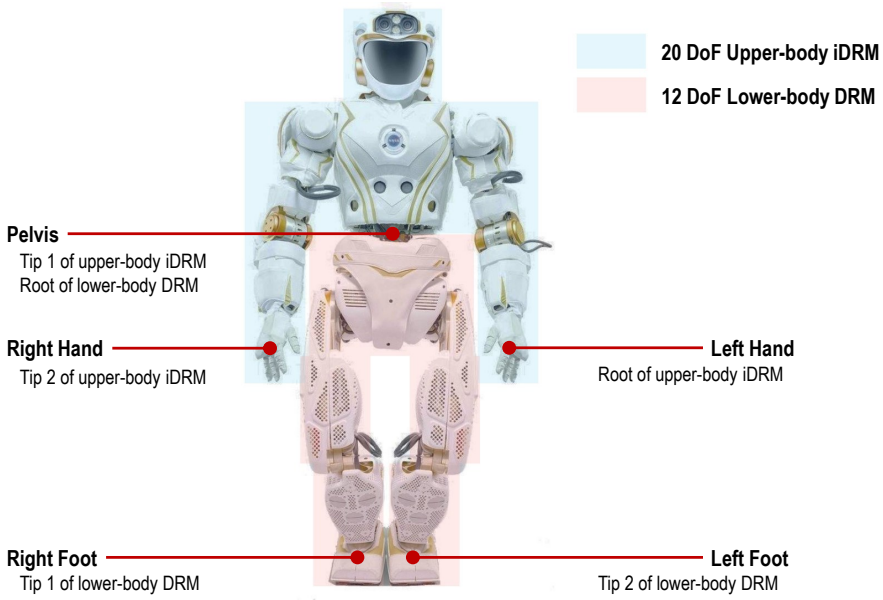


Figure 3.4: Kinematic split of NASA's Valkyrie 38-DoF humanoid robot. Concerning the DoF of the individual body parts, each leg has six (6), each arm has seven (7), the robot torso has three (3), and the neck has three (3). The pelvis represents an extra 6-DoF virtual joint that connects the robot to the world.

3.3 End-Pose Planning for Bi-manual Tasks on Uneven Terrain

3.3.1 Constructions of DRM/iDRM for Humanoids

Upper-Body iDRM

In this case study, the left hand is selected as the root link of the upper-body iDRM, and the right hand and pelvis are treated as two tip links. Several iDRM datasets with different number of samples (all with 10cm workspace voxel resolution) are generated for the 20-DoF upper-body of Valkyrie. Traditionally, samples of an inverse reachability should cover the whole C-space, *i.e.* for the case of a humanoid, samples of the map should reach behind the robot. However, since the robot's sensor are predominantly facing forward, we want to express a preference for stable stance locations that give us reasonable manipulability. We adopt a heuristic in our method, where we only store samples with both hands reaching comfortable manipulation poses in front of the robot, as shown in Fig. 3.5. Note that the robot can still manipulate objects that are currently far away or behind the robot by walking to an appropriate pre-action stance location, which is the key point of end-pose planning.

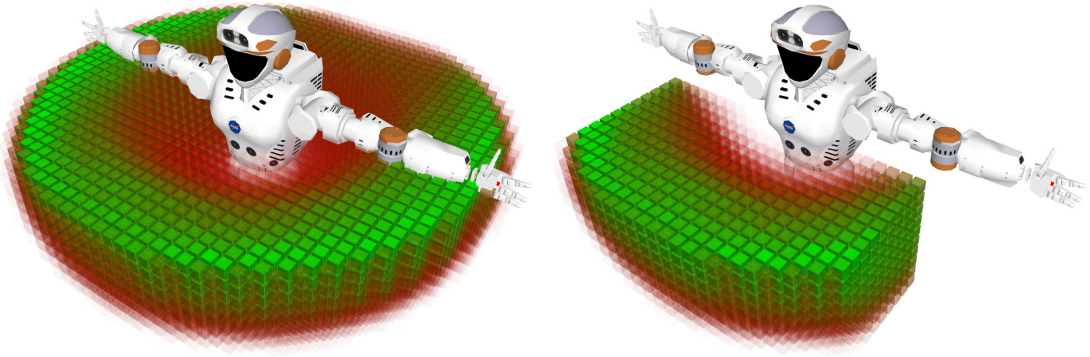


Figure 3.5: Left: the upper-body's full Reachability Map; right: the Reachability Map constrained to the front of the robot. All coloured voxels are reachable by the robot and greener voxels are regions with high reachability scores. Only part of the map is plotted for clarity (the shape of the whole map is similar to the one of a sphere).

Lower-Body DRM

The lower-body of Valkyrie has 12-DoF (6-DoF per leg). Though the legs have a large range of motion, the manifold of balanced configurations is much smaller even on uneven terrain. Therefore, we have reduced the "*reachability*" map for the lower-body so that the legs have the range to adapt to the uneven terrain but they won't reach most unnatural poses[2]. To this end, we generate lower-body configurations with two feet placed in a region below the pelvis ($0.8 - 1.1$ meter for Valkyrie), as shown in Fig. 3.6. This ensures that the lower-body DRM has sufficient samples to adapt to uneven terrain without demanding extra memory for storing poses that can't provide support for the robot, *e.g.* poses where the feet reach above the pelvis.

3.3.2 End-Pose Planning

Let M_{upper} be the upper-body iDRM and M_{lower} be the lower-body DRM. Given a task $\mathbf{y}^* = (\mathbf{y}^*_{lhand}, \mathbf{y}^*_{rhand})$, start states $\mathbf{p}_s, \mathbf{q}_s$ and the environment Env, the end-pose planner needs to find an end-pose that contains $\mathbf{p}^* = (\mathbf{p}^*_{lfoot}, \mathbf{p}^*_{rfoot})$ and $\mathbf{q}^*$. Firstly, we create two tip pose constraints $C = \{C_{pelvis}, C_{rhand}\}$ for the upper-body iDRM, where C_{pelvis} constrains the pelvis link to be inside a feasible height region and approximately perpendicular to the ground (*i.e.* upright), and C_{rhand} constrains the right hand to be near $\mathbf{y}^*_{rhand}$. Algorithm 3 highlights our proposed end-pose planning method for bimanual tasks on uneven terrain, where in lines 1-7 M_{upper} is used to find collision-free upper-body configurations that satisfy the constraints C, such that two hands can reach the goal $\mathbf{y}^*$ with the pelvis pose T_{pelvis}.

[2]Though a metric of being "unnatural" appears to be subjective, it has meaningful implications for achieving such poses on a real robot in terms of joint range and sustainable power. In our work, we define the terms *natural* and *comfortable* as the distance in the C-space from a chosen nominal configuration derived from the posture shown in Fig. 3.4.

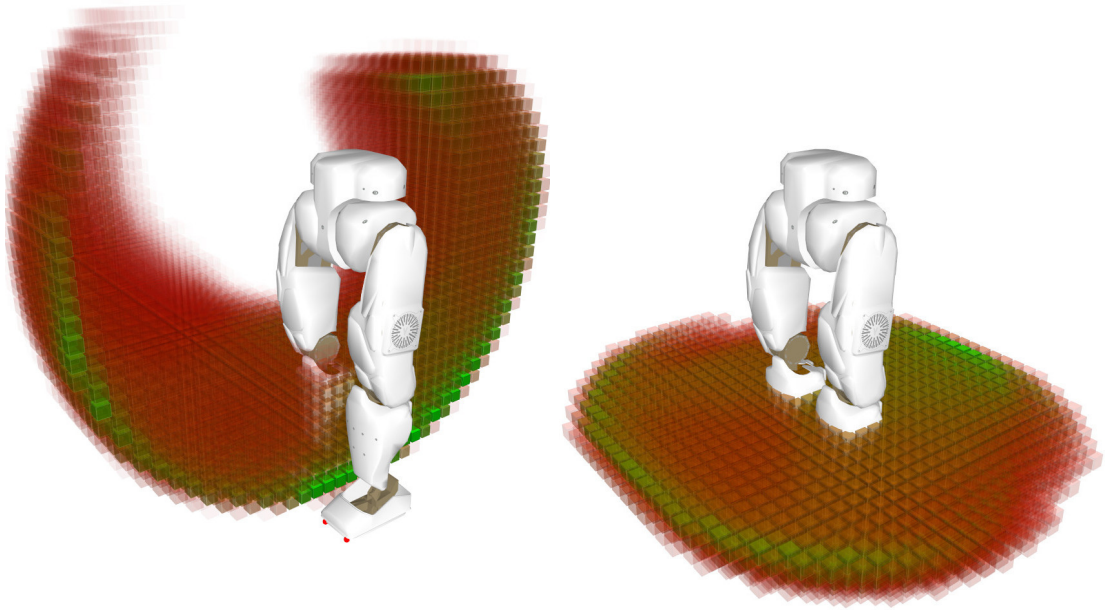


Figure 3.6: Left: the lower-body's unconstrained Reachability Map; right: the Reachability Map constrained to feet placed below the pelvis. Only part of the map is plotted for clarity.

Algorithm 3 Humanoid End-Pose Planning

Require: $\mathbf{y}^*_{lhand}$, C
Ensure: $\mathbf{p}^*_{lfoot}$, $\mathbf{p}^*_{rfoot}$, $\mathbf{q}^*$

$\mathbf{y}^*_{root} \leftarrow \mathbf{y}^*_{lhand}$
Transform M_{upper} **to** $\mathbf{y}^*_{root}$ // *Fig. 3.3b(2)*
CollisionUpdate(M_{upper}) // *Fig. 3.3b(2-3)*
$\mathbf{Q} \leftarrow \emptyset$
for $\forall \mathbf{q}_n \in$ collision-free subset of M_{upper} **do**
    $T_{pelvis}, T_{rhand} \leftarrow$ TipGlobalPoses($\mathbf{q}_n, \mathbf{y}^*_{root}$)
    if SatisfyConstraint(T_{pelvis}, T_{rhand}, C) **then** // *Fig. 3.3b(4)*
        Transform M_{lower} **to** T_{pelvis} // *Fig. 3.3a(2)*
        CollisionUpdate(M_{lower}) // *Fig. 3.3a(2-3)*
        for $\forall \mathbf{q}_m \in$ collision-free subset of M_{lower} **do**
            $\mathbf{p}_{lfoot}, \mathbf{p}_{rfoot} \leftarrow$ TipPoses $(T_{pelvis}(\mathbf{q}_n), \mathbf{q}_m)$
            if ValidTerrainContact($\mathbf{p}_{lfoot}, \mathbf{p}_{rfoot}$) **then** // *Fig. 3.3a(4)*
                $\mathbf{q} \leftarrow \{\mathbf{q}_n, \mathbf{q}_m\}$
                if $\mathbf{q}$ is balanced **then**
                    $\mathbf{Q} \leftarrow \mathbf{Q} \cup (\mathbf{p}_{lfoot}, \mathbf{p}_{rfoot}, \mathbf{q}, f(\mathbf{q}))$
$\mathbf{p}^*_{lfoot}, \mathbf{p}^*_{rfoot}, \mathbf{q}^* \leftarrow$ LowestCost $(\mathbf{Q})$
return $\mathbf{p}^*_{lfoot}, \mathbf{p}^*_{rfoot}, \mathbf{q}^*$

It is worth emphasising that, given an upper-body configuration $\mathbf{q}_n$, the global pose of a link can be calculated by FK, but it is not necessary since we can retrieve these poses directly from iDRM. For each tip link, *i.e.* pelvis and right hand, the iDRM reach pose is referenced in the root (left hand) frame. Given the desired root pose $\mathbf{y}^*_{lhand}$, the global pose of a tip link is

$$T_n^{tip,world} = \mathbf{y}^* \times T_n^{tip,root} \tag{3.1}$$

... where $T_n^{tip,world}$ and $T_n^{tip,root}$ represent the tip pose of sample n in global and root frames accordingly. Here $T_n^{tip,root}$ is pre-computed for each sample during offline processing and $\mathbf{y}^*$ is given for each task. Hence, computing the global poses of the pelvis and the right hand is very efficient in our approach.

After retrieving the global poses, we can then check if the configurations satisfy the pelvis and right hand constraints. For a candidate upper-body configuration $\mathbf{q}_n$, we transform M_{lower} to T_{pelvis} and find valid lower-body configurations, *i.e.* collision-free and valid contacts with the terrain, as shown in lines 8-12 of Algorithm 3. To check foot contacts, we first extract the step regions from the environment. Similar to Eq. 3.1 with T_{pelvis} as the $\mathbf{y}^*$, we can obtain the tip (foot) poses in the global frame and check if the foot is within the step regions. If the lower-body configuration has valid contacts, we then combine the candidate upper and lower body configurations to acquire the whole-body configuration. Since multiple valid end-poses may exist, we iterate though M_{upper} and M_{lower} to find the best candidate based on the cost function $f(\mathbf{q})$. Different cost functions can be defined for different tasks and environments. In general, for humanoid robots, it is desirable to have an end-pose with minimum travelling distance that is close to the start/nominal configuration. The following cost function is used in our implementation,

$$f(\mathbf{q}) = \|T_{pelvis}(\mathbf{q}) - T_{pelvis}(\mathbf{q}_s)\|_{W_1} + \|\mathbf{q} - \mathbf{q}_s\|_{W_2}, \tag{3.2}$$

where W_1, W_2 are weights.

After end-pose planning, the last step is to refine the output and ensure all necessary constraints are satisfied, *e.g.* the hand(s) need to precisely reach the target, the feet need to be perfectly in contact with the terrain, and the pose needs to be statically balanced. A non-linear optimisation-based solver [TD] is used to adjust the candidate end-pose with respect to these constraints by applying a SQP solver in the same form of Equation 2.1. If the solver fails or the solution is in collision, the optimisation is repeated with the next best candidate end-pose[3].

[3]Please keep this in mind as it will be relevant to explain an important detail of our implementation in one of the following sections of this chapter — *cf.* 3.4.3.

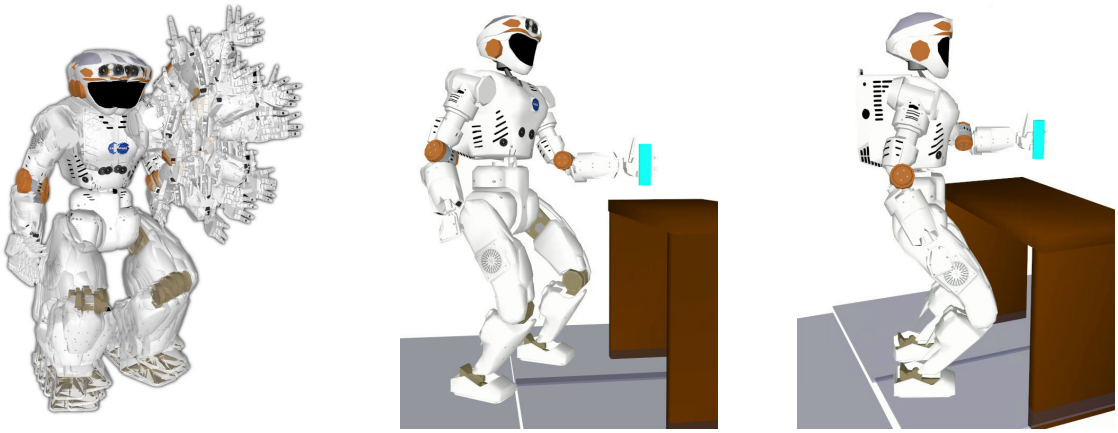


Figure 3.7: The first figure highlights the upper-body iDRM and lower-body DRM samples, followed by two examples of selected end-poses in different scenarios.

3.3.3 Footstep and Motion Planning

After finding the end-pose, a footstep planner is invoked to plan a set of footsteps to enable walking from current stance location $\mathbf{p}_s$ to pre-grasp stance location $\mathbf{p}^*$, followed by a motion planner to generate a valid whole-body trajectory to realise the end-pose $\mathbf{q}^*$. Footstep and motion planning are not the main focus of this work, and any suitable algorithms could be used. The footstep planner from [DT14] and the whole-body motion planner from [Yan+16b] are implemented here.

3.4 Evaluation

3.4.1 Construction of Dynamic Reachability Maps

We have generated maps with different root/tip links and number of samples to analyse how different splitting of the map affects the performance:

- Φ_1: An upper-body iDRM with the left hand as the root, pelvis and right hand as the tips. Three datasets are generated with different number of samples: $100,000(\Phi_{1a})$, $1,000,000(\Phi_{1b})$ and $4,000,000(\Phi_{1c})$.
- Φ_2: An upper-body iDRM with the left hand as the root, pelvis and right shoulder as the tips. Three datasets are generated with different number of samples: $10,000(\Phi_{2a})$, $100,000(\Phi_{2b})$ and $1,000,000(\Phi_{2c})$.
- Φ_3: A right arm DRM with right shoulder as the root and right hand as the tip. Three datasets are generated with different number of samples: $10,000(\Phi_{3a})$, $100,000(\Phi_{3b})$ and $1,000,000(\Phi_{3c})$.

- Φ_4: A lower-body DRM with the pelvis as the root, left and right feet as the tips. Four datasets are generated with different number of samples : $1,680(\Phi_{4a})$, $44,400(\Phi_{4b})$, $227,400(\Phi_{4c})$ and $742,560(\Phi_{4d})$.

Table 3.1: Map construction analysis.

Map		No. samples	Construction time (min)	File size (MB)
Upper-body two arms	Φ_{1a}	10^5	28.8	108
	Φ_{1b}	10^6	289.7	1,082
	Φ_{1c}	4×10^6	1090.8	4,352
Upper-body left arm	Φ_{2a}	10^4	0.25	9
	Φ_{2b}	10^5	2.61	91
	Φ_{2c}	10^6	25.0	879
Right arm	Φ_{3a}	10^4	0.05	2
	Φ_{3b}	10^5	0.58	22
	Φ_{3c}	10^6	6.19	217
Lower-body two legs	Φ_{4a}	$1,680$	0.24	1
	Φ_{4b}	$44,400$	6.15	33
	Φ_{4c}	$227,400$	30.0	160
	Φ_{4d}	$742,560$	103.5	535

All datasets are created with 10cm workspace grid resolution. The construction time and file size are highlighted in Table 3.1. The construction time of Φ_1 maps are relatively longer because many of the samples are discarded and only these with both hands fall into the region of interest are kept. The Φ_1 maps are also expensive to store since the kinematic structure includes the entire upper-body with two arms. It is worth emphasising that the file size of Φ_1 is similar to Φ_2 and Φ_3 combined with same number of samples, *e.g.* $\Phi_{1b} \approx \Phi_{2c} + \Phi_{3c}$.

The proposed end-pose planning method can be obtained by combining Φ_1 and Φ_4, for example, combining Φ_{1a} and Φ_{4a} gives a dataset with a theoretical $10^5 \times 1680 = 168$ million whole-body configurations; combining Φ_{1c} and Φ_{4c} gives a dataset with a theoretical 909.6 trillion whole-body configurations. A further split method can be obtained by combining Φ_2, Φ_3 and Φ_4, for example, combining Φ_{2c}, Φ_{3c} and Φ_{4c} gives a dataset with a theoretical 2.274×10^{17} whole-body configurations. It is clear that the total number of whole-body configurations increases exponentially with the number of components. However, combining these maps significantly slows down the on-line planning (see Section 3.4.3).

3.4.2 End-Pose Planning Benchmarking Setup

We have crated a set of benchmark problems by passing random hands and feet pose constraints, as well as quasi-static balance constraint, into the whole-body IK solver to

obtain a random but balanced configuration. The configurations are filtered for self-collisions. We then populate spherical obstacles into the free environment randomly but not colliding with the robot until a required number of obstacles is reached. Finally, we can extract the height and position of each foot from the generated configuration and create terrain areas accordingly. A valid end-pose planning problem is thereby generated. We also store the desired poses for both hands, collision environments and terrain areas. Note that the robot configurations are generated to ensure the problem is solvable with at least one solution. The configuration is not known to the candidate algorithm, and the algorithm is allowed to find a different but valid solutions if multiple solutions exist. In our benchmarking, we created 1000 random problems, each of which contains 20 spherical obstacles with 15-20cm radius.

3.4.3 Simulation Benchmarking

Reducing the Candidate Set

Our first approach to select the final end-pose from the candidate set was to traverse the list in a linear fashion. However, this is a very time-consuming process due to the large number of whole-body samples it contains for the majority of the planning requests.

In a first attempt to reduce the time spent in this process, we changed our implementation to select only the first K samples from the candidate set. In our implementation, we chose to make $K = 20$, but this is a program variable which can be easily reconfigured, and should reflect the trade-off between the *time* and the *amount of samples* we allow the planning request to explore before quitting. We soon discovered that this technique was not the best match for our approach when we benchmarked it, as it did not produce very good success rates. Indeed, when we create a subset of the entire set of candidates, $\mathbf{Q}$, by selecting the first K samples in a linear fashion, we are selecting neighbouring samples – very similar whole-body configurations – which lead to the same IK collision result. Now, if the first candidate sample of the subset does not satisfy all the problem constraints during the final collision check, it is very likely that its neighbours will also fail to meet those constraints. Filtering the candidates list in this manner might discard poses with relatively high cost – which are further away from the head of the candidate list – but valid nonetheless.

In an effort to improve the success rate results of our planning approach, we changed the method used to create the subset of the entire list of candidates: instead of selecting the *first* K samples of the list, we select K equally spaced candidates *across* the whole candidate set, *e.g.* for a set containing 20 samples and for $K = 4$, the subset would contain the 1st, 5th, 10th, and 15th samples, therefore prioritising the exploration of the list of candidates in its entirety. This technique results in much better planning success rates, which we will be presenting in the following sections.

Different Lower-Body Datasets

As we have mentioned, the lower-body is used for maintaining balance rather than for maximum reachability. Thus, we should use a dataset that contains enough samples which is sufficient for finding balanced configurations rather than having a dataset with millions of samples that consumes huge amount of memory and slows down on-line computation. We combine Φ_{1b} with different Φ_4 maps to analysis the affects different lower-body maps might introduce and therefore select the suitable one for other experiments. We also evaluated the performance by directly applying the non-linear IK without using DRM/iDRM. Table 3.2 shows the success rate and average planning time using different methods. The map success rate is the rate of DRM/iDRM reports finding valid candidate end-poses, which is then passed to the IK adjustment function. The IK success rate is the rate of non-linear IK successfully adjusted the candidate poses and satisfy all constraints. The pose is then passed to a collision checking function, a final success is reported if the pose is collision-free.

Table 3.2: End-pose planning performance across different lower-body datasets and using the non-linear full-body IK.

Method	Map success rate	IK success rate	Final success rate	Avg. time(s)
$\Phi_{1b} + \Phi_{4a}$	72.7%	71.8%	71.4%	0.08 ± 0.02
$\Phi_{1b} + \Phi_{4b}$	73.7%	72.8%	72.5%	0.09 ± 0.03
$\Phi_{1b} + \Phi_{4c}$	80.7%	79.0%	78.7%	0.13 ± 0.10
$\Phi_{1b} + \Phi_{4d}$	86.3%	84.8%	84.2%	0.23 ± 0.33
Non-Linear IK	-	99.8%	59.3%	0.03 ± 0.01

We notice that these methods can not achieve 100% success rate, which is caused by several factors: firstly, although we have created each map with millions of configurations, it is still inefficient to cover the high dimensional whole-body C-space (38 dimension for Valkyrie); secondly, and in the interest of time, we only allow the method to try 20 different poses from the set of candidates, $\mathbf{Q}$, due to the reasons we mentioned in 3.4.3; lastly, some valid poses which are not in collision may get invalidated due to aliasing of the occupancy grid. Such artefacts can be reduced by using a finer workspace grid, but they can't be completely eliminated. This is a common issue with all grid-based methods.

It is interesting that the final success rate is very close to the initial map success rate, which means that once the DRM/iDRM maps find candidate end-poses, those poses are very likely to be valid. On the other hand, the direct non-linear IK method reports a 99.8% success rate, but only 59.3% is finally valid, *e.g.* collision-free. The result suggests that using only the non-linear IK is inefficient in cluttered environments, and the proposed method is indeed improving the success rate.

The benchmarking was done in randomised and complex environments designed to fully evaluate different approaches. Although the methods do not achieve 100% success rate in the benchmarking, as we will show later in Section 3.4.4, they are sufficient for

solving practical problems. Based on the result we conclude that the success rate as well as planning time increase with the number of lower-body samples. We use the lower-body dataset Φ_{4c} for the rest of the experiments. However, other datasets with more samples might be used depending on the different demands between success rate and planning time.

Different Map Combinatorics

Table 3.3: Analysis of end-pose planning performance with different upper-body datasets and the same lower-body dataset. Considering the trade-off between success rate and planning time, the method $\Phi_{1c} + \Phi_{4c}$ is used for hardware experiments.

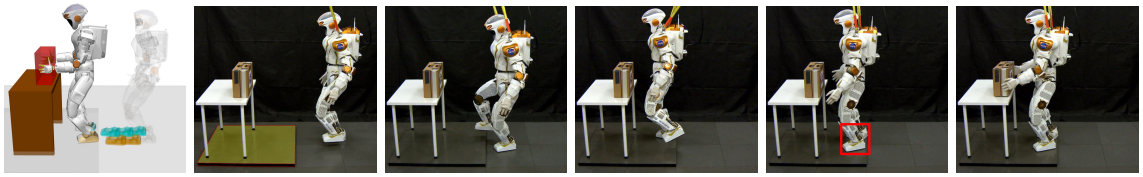
Method	Total No. samples	Map success rate	IK success rate	Final success rate	Avg. time (s)
$\Phi_{1a} + \Phi_{4c}$	2.274×10^{10}	57.9%	57.1%	56.8%	0.04 ± 0.01
$\Phi_{1b} + \Phi_{4c}$	2.274×10^{11}	80.7%	79.0%	78.7%	0.13 ± 0.10
$\Phi_{1c} + \Phi_{4c}$	9.096×10^{11}	88.6%	85.7%	**85.1**%	0.40 ± 0.37
$\Phi_{2a} + \Phi_{3a} + \Phi_{4c}$	2.274×10^{13}	70.0%	65.1%	63.7%	0.10 ± 0.05
$\Phi_{2b} + \Phi_{3b} + \Phi_{4c}$	2.274×10^{15}	91.3%	83.5%	80.4%	0.56 ± 0.39
$\Phi_{2c} + \Phi_{3c} + \Phi_{4c}$	2.274×10^{17}	96.9%	85.0%	81.2%	8.08 ± 4.68

We choose to split the humanoid robot into two parts at pelvis. However, one can further split the upper-body into smaller parts, *e.g.* left body part (Φ_2) and right arm (Φ_3). Table 3.3 shows the end-pose planning result of using different upper-body maps, where the success rate and planning time increases with the number of samples as expected. However, the further splitting ($\Phi_2 + \Phi_3 + \Phi_4$) leads to a much longer planning time while the success rate is not significantly improved comparing to the proposed splitting ($\Phi_1 + \Phi_4$). Furthermore, in the case of using further split method with maps $\Phi_{2c} + \Phi_{3c} + \Phi_{4c}$, the final success rate is lower than using proposed split method with maps $\Phi_{1c} + \Phi_{4c}$. Note that the map reports a 96.9% success rate, but dropped to 85.0% after IK adjustment, most of which were caused by failure to satisfy balance constraint. This means further splitting the body leads to higher chance of violating the balance constraint of the whole-body. Splitting the upper- and lower-body at the pelvis link is thereby proved to be the most practical approach considering the trade-off between coverage, planning success rate, and algorithm runtime. We use the proposed split method with datasets Φ_{1c} for upper-body and Φ_{4c} for lower-body for the following experiments on robot hardware.

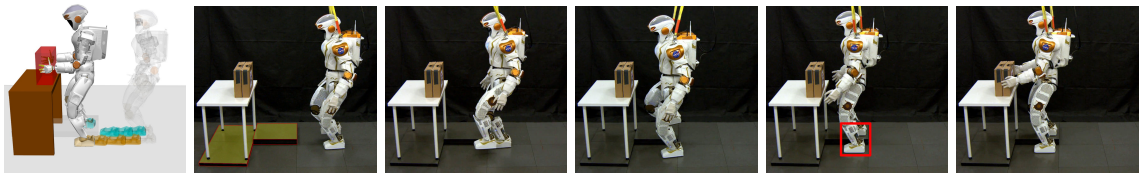
3.4.4 Hardware Experiments

To demonstrate the capability of end-pose planning on uneven terrain, we created three bimanual box-picking tasks with different terrain types. In the first scenario $B1$ (Fig. 3.8a), the robot has to walk onto a higher floor, which in theory can be found by classic iDRM as well; in the second case $B2$ (Fig. 3.8b), the robot has to stand on surfaces at two different heights; in the last scenario $B3$ (Fig. 3.8c), the robot needs to avoid a collision between its right leg and a large obstacle during the picking task. Our method is capable of finding different collision-free end-poses in these environments. We found that the possible pelvis

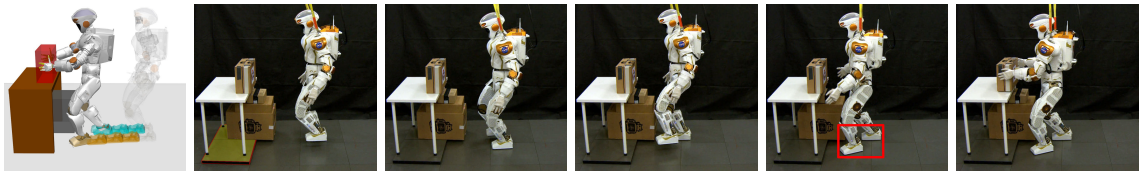
poses are quite limited in practice for bimanual tasks, *i.e.* the robot has to stand directly in front facing the box in order to pick it up with two hands. Nevertheless, our DRM/iDRM hybrid method provides a valid solution for the robot to perform bimanual picking tasks in presence of uneven terrain.



(a) $B1$: Pick up a box from a higher terrain.



(b) $B2$: Pick up a box by placing the right on a higher terrain.

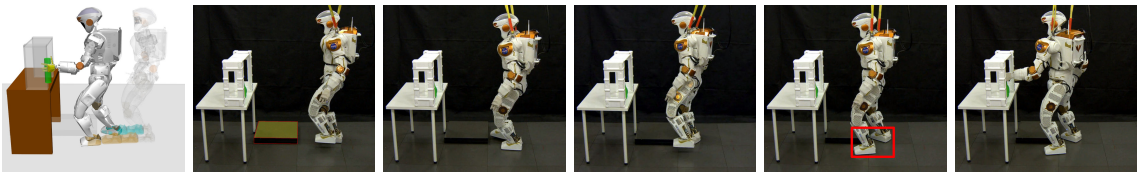


(c) $B3$: Pick up a box while the right leg position is restricted by a large obstacle.

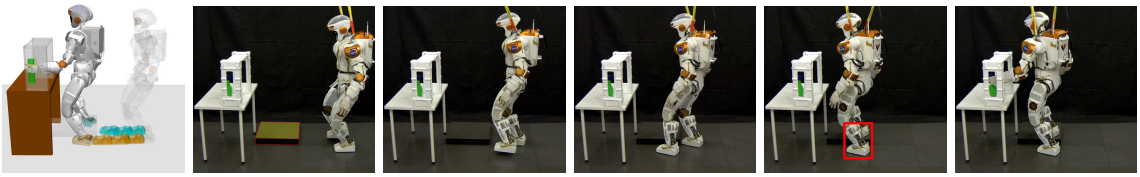
Figure 3.8: Bimanual box-picking tasks on terrains at different heights. The robot is able to automatically find appropriate standing locations and whole-body configurations.

We further validated two single-arm grasping tasks where the target was placed at different locations, as shown in Fig. 3.9. An upper-body iDRM is created with the left hand as root link and pelvis as tip link. The right arm joints are set to a pre-defined nominal configuration for all samples, as shown in Fig. 3.7. The constrain set C then contains pose constraints only for the pelvis but not for the right hand. In the first scenario $S1$ (Fig. 3.9a), the target was placed at the edge of the table, where the robot could easily grasp without being too close. So, the robot could stay away from the high surface, while keeping the target at a reachable distance. Whereas in the second task $S2$ (Fig. 3.9b), the target was placed further away from the edge of the table and enclosed by the obstacle. The end-pose planner found a feasible configuration to place two feet on different surfaces so the robot was close enough for grasping the target.

We would like to highlight that with the modular and combined forward inverse dynamic reachability maps presented in this work, we are able to find end poses which include lunging body or taking a sidestep (in scenarios *B3* and *S1*) for increasing the reachable workspace by leveraging the advantage of the legged system. This is in contrast with the prior work [BB15; Yan+16a] which limited the foot poses to a constant distance and planning for the mid-feet point. A supplementary video can be found at `https://youtu.be/o-05EHf-gg8`.



(a) *S1*: Grasp a target placed at the edge of the table.



(b) *S2*: Grasp a target placed deeper on the table.

Figure 3.9: Single-handed grasping tasks on terrains at different heights. Case I: the target is easily reachable, so the robot does not need to be too close to the table; Case II, the robot needs to be closer to the table by placing the right foot on the uneven terrain.

3.5 Summary

We presented a novel end-pose planning algorithm that combines the DRM and iDRM, which allows humanoid robots to automatically find appropriate end-poses in presence of uneven terrain. Using NASA's Valkyrie humanoid as a test bed, we demonstrated the effectiveness of the proposed method in planning end-poses for both single-arm and bimanual tasks on uneven terrains.

Chapter 4

Extending End-Pose Planning to Inclined Surfaces

In this chapter we suggest an extension to our previous work concerning end-pose planning from flat supports at different heights (Chapter 3) to inclined supports. Our hypothesis is that end-pose planning on inclined terrains can be achieved by:

(a) integrating into the pipeline a method to compute the quasi-static balance of kinematic structures for arbitrary contact geometries. Namely, one that takes into account the static friction between the robot contact points and the environment, and;

(b) adjusting the sampling method of the lower body, to one that explores the C-space of the robot's lower body support limbs.

Ultimately, the goal of this chapter is to enable efficient planning of collision-free balanced whole-body robot end-poses, provided (i) a set of support regions, (ii) a list of the obstacles present in the robot's surroundings, and (iii) a single- or multi-handed grasping target(s). Fig. 4.1 exemplifies an obstacle-free scenario populated with multiple inclined green tiles — which serve as support regions for the robot — and a feasible whole-body posture which satisfies two given grasping goals (the small red and green cubes).

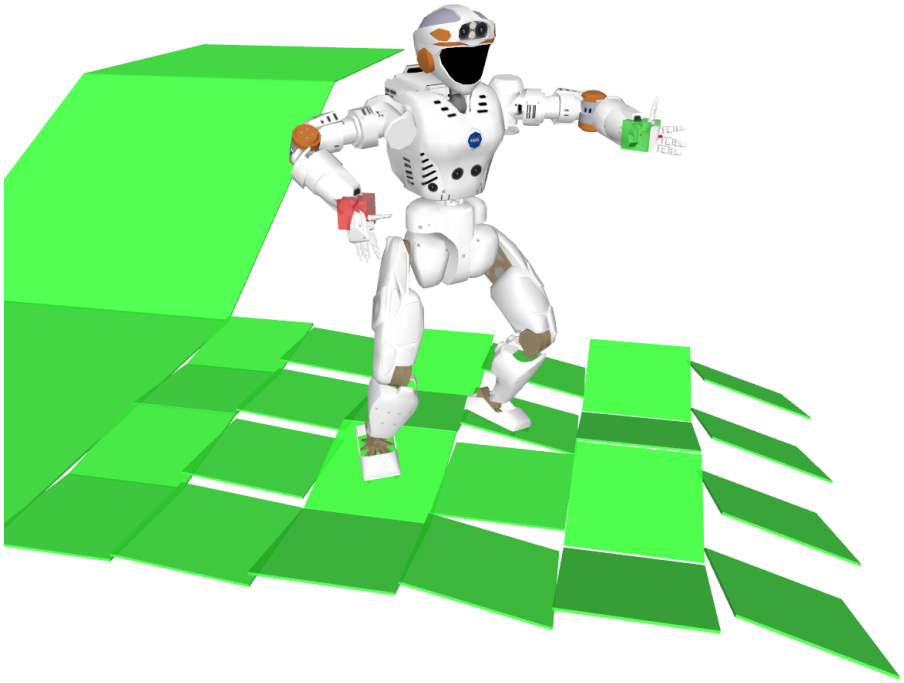


Figure 4.1: End-pose planning taking into account the inclination of support regions.

4.1 Robust Static Equilibrium

A system in contact with a flat ground is said to be in static equilibrium if and only if the vertical projection of its Centre of Mass (CoM) lies within the convex hull of the support polygon — a shrunk area of the contact polygon:

$$c^{xy} \in \text{Hull}(\{p_i^{xy}\}), \tag{4.1}$$

... where c^{xy} is the CoM projection, and $\text{Hull}(\{p_i^{xy}\})$ is the convex hull of the contact points on the ground, that is a convex 2D polygon.

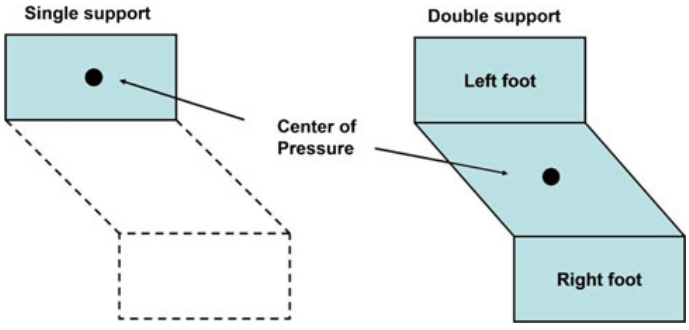


Figure 4.2: Centre of pressure and support polygon. Reprinted from [Di +12]. Copyright © 2012 by the IJIDeM.

The support polygon of a biped humanoid robot standing on its left foot only (left) and on both its feet (right) is illustrated in Fig. 4.2. The system is in static equilibrium as long as the CoM — the bold black circle — lies within the coloured area of the support polygon. Furthermore, this condition still holds true for flat sloped grounds, as long as the friction at the contact points is large enough not to cause the system to slip. Should this happen, checking whether a system is in static equilibrium is not so straightforward.

Figure 4.3: Close-up photo of the feet of NASA's Valkyrie. From NASA – Past and Present Dreams of the Future. Copyright © 2016 by Benedict Redgrove.

Many robustness measures to test for static equilibrium have already been proposed. Caron et al. [CPN15] suggested to check for the capacity of a system to generate $x - y$ CoM accelerations within a polytope while maintaining the derivative of the angular momentum null. Barthélemy and Bidaud [BB08], [Qiu+11] proposed a robustness measure based on the radius of the largest hypersphere centred at the Gravito-Inertial Wrench (GIW) and fully contained inside the GIW cone. Del Prete and Mansard [DM15] studied the control of legged robots with focus on the robustness to torque-tracking errors — an uncertainty source that affects the contact-force tracking.

Later on, Del Prete et al. [DTM16] proposed to account for robustness to errors in the contact-force tracking, *i.e.* to prevent the forces necessary to maintain equilibrium from being too close to the boundaries of the friction cones. The authors presented two algorithms to test equilibrium that are faster than previous approaches, and proposed a method to test robust equilibrium that allows the robot not to lose contact in case of bounded force-tracking errors. In their tests, the authors solved the Linear Programming (LP) problems with *qpOases* [Fer+14] — a C++ parametric active-set solver for Quadratic Programming (QP) that also supports LP. For the polytope projection they used the C++ *cdd* library [FP96]. Subsequently, the authors wrote a C++ implementation of the

algorithms they presented in [DTM16] to check the robust equilibrium of a system. In the next subsection we will explain how we integrated this library into our codebase and into our framework for whole-body end-pose planning.

4.1.1 Robust Static Equilibrium Applied to Whole-Body End-Poses

The utility classes implemented by Del Prete et al. to compute the robust static equilibrium of a system require the following inputs:

- A list of contact points
- A list of contact normals
- The contact friction coefficient
- The number of generators used for the linear approximations of the friction cones
- The mass of the system
- The CoM position of the system

In our implementation, each foot of the robot has four contact points associated to it. In turn, each of those contact points is coupled to a contact normal whenever a support region and a foot's centre of pressure frame touch each other. Furthermore, we assume the contact friction coefficient to be 0.3, which is comparable to the friction between *polystyrene* and *steel* on clean and dry surfaces. This is an under-approximation of what we believe to be the true friction coefficient between the robot's foot sole material and the material on which it will be operating most of the times[1]. We use four generators for the linear approximations of the friction cones. Finally, the mass of NASA's Valkyrie is approximately 135.906 kg, 77.976 kg of the upper-body and 57.93 kg of the lower-body, according to the kinematic split described in Section 3.2 (p. 24).

After having integrated the aforementioned utility classes, we are now able to compute the robustness of the equilibrium of any given whole-body configuration — one needs only to extract the CoM, and the list of contact points and their respective normals from the whole-body pose, in order to pass them as inputs to the library. The implementation returns a quantitative measure of the robustness of the system's equilibrium. Negative values mean the system can not be in equilibrium. Positive values mean the system is in equilibrium, and the greater the returned value is, the more robust is the state of the system's equilibrium. Fig. 4.4 contains two examples of whole-body end-poses. The image on the left is an example of a system standing in a feasible upright fashion. The image on the right is an example of the robot leaning over such that it would not be able to be in quasi-static balance. The red arrows located at the vertices of the sole of each foot are

[1] The kinetic or sliding frictional coefficient — applicable only when there is a relative motion between the surfaces — between *rubber* and *dry concrete* is in the range from 0.6–0.851, and between *rubber* and *wet concrete* is in the range from 0.45–0.75. Without motion the values are somewhat higher. [Too]

visual markers representing the contact normals between the feet and the green support regions. The contact points are located at the root of the arrows.

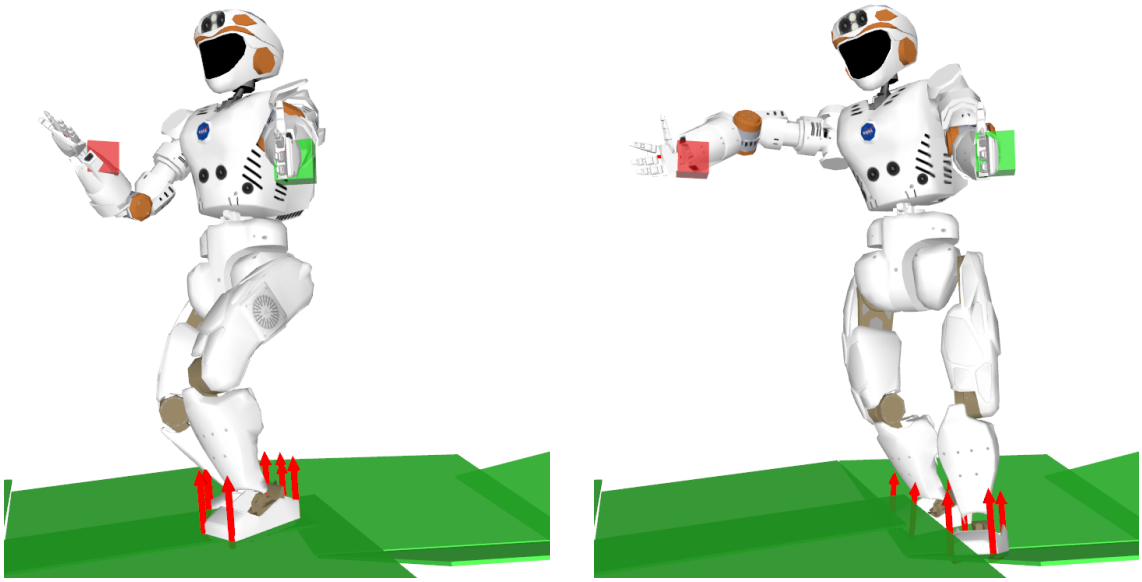


Figure 4.4: A robust (left) and a non-robust (right) whole-body end-poses.

4.1.2 Computing the Robustness of Lower-Body Samples

Heretofore, we have a method to compute the robustness of whole-body configurations. Nevertheless, we have yet to generate an extended and improved lower-body dataset containing lower-body samples which enable planning of stances on inclined terrains. To that end, it is desirable to be able to compute the robustness of stand-alone lower-body samples. Using a similar approach to what we discussed previously, we can use the same library to compute the robustness of a given lower-body sample by supplying the CoM position and the mass of the lower-body alone. This is indeed a valid approach. Notwithstanding, the robustness evaluation of a system consisting only of the lower-body will carry very different results of that very same lower-body sample combined with an upper-body sample.

A better alternative, which we have decided to use in our implementation, is to simulate a dummy mass, just above the pelvis link[2], with the same mass as the upper-body of the robot's kinematic structure. This is a rough approximation, after all, depending on the upper-body configuration, its CoM will be positioned somewhere else other than at 30 cm above the pelvis link. However, do note that this approximation results in a more accurate robustness estimation rather than not considering an upper-body sample at all.

At this stage we have a way to compute the robustness of individual lower-body samples by approximating the upper-body with the usage of a dummy mass. While analysing

[2]More specifically, 30 cm above the pelvis link in our implementation.

this approach, we noticed that occasionally some configurations would be classified as having a robustness of two, and sometimes three, orders of magnitude greater than all the other samples. Henceforth, we will denote these abnormal samples as *extremely robust* configurations. Fig. 4.5 shows the visual representation of two lower-body samples for which the library returns an unexpectedly high robustness measurement.

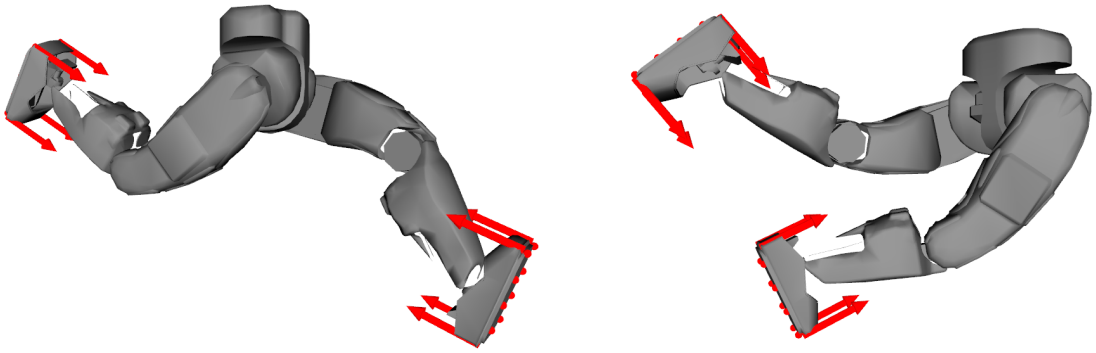


Figure 4.5: Two lower-body samples evaluated as *extremely robust*. Both samples are unfeasible and would cause the system to collapse. Thus, we discard such samples and consider them as outliers. The red arrows are visual markers representing the contact normals between the feet and fictitious support regions coplanar to the sole of each foot.

These outliers are evaluated as having a robustness orders of magnitude above the expected results, but it can be easily seen that they do not represent valid balanced configurations. We did not investigate this issue further. To account for these outliers, we discard samples evaluated with a robustness above 500.

4.2 Lower-Body DRM Construction

Previously, in 3.3.1 (p. 26) we generated lower-body configurations with the two feet placed below the pelvis and with neither *roll* nor *pitch*. This was based on the assumption that the terrains would always be flat, even though they could be positioned at different heights. For more complex terrains, namely sloped surfaces, this no longer holds true, and as a consequence the need of a new method to generate a lower-body DRM arises.

We used two different lower-body sampling methods: (a) random sampling, and (b) selective sampling. For each method, we generated three datasets of different sizes — 1000, 5000, and 10000 samples. Following, we explain these new datasets in more detail.

4.2.1 Sampling Methodology

Random Sampling

We generated three random lower-body datasets, without discarding any of the generated samples. Each dataset contains 1000, 5000, and 10000 lower-body samples.

Selective Sampling

This method samples random configurations, just like the previous method. However, in order for a generated sample to be kept in the dataset, it must meet the following criteria: (i) the angle between the normal of the foot and the z-axis must not exceed 30°; (ii) the horizontal distance between the feet must not be greater than 0.5 m; (iii) the z-distance between the feet must not be greater than 0.2 m; (iv) the angle between the headings of each foot must not exceed 60°; (v) the robustness evaluation must be greater than or equal to 10. The sampling process ends when the target number of samples to be generated is reached.

The constraints (i) – (iv) are set due to inherent hardware limitations of the robot, namely joint and torque limits. Constraint (v) is set as a robustness threshold to ensure that only balanced configurations are kept in the dataset[3].

4.2.2 Lower-Body Datasets Robustness

In order to compare the different sampling methods described in 4.2.1 in terms of equilibrium robustness, we used the method from 4.1.2 (p. 41) to calculate the robustness of each sample of every dataset, and for each dataset we calculated the average robustness of its samples. Table 4.1 shows the average robustness for each of the generated datasets.

Table 4.1: Average robustness of the samples contained in each of the generated lower-body datasets.

Method	No. samples	Avg. robustness
Random	1000	-109.99
	5000	-111.44
	10000	-42.01
Constrained	1000	16.82
	5000	17.19
	10000	11.99

In order to better understand the content of each dataset in terms of equilibrium robustness, we have also plotted histograms of the frequency of each robustness measurement for each of the datasets. Fig. 4.6 contains those histograms. The three histograms on the top row correspond to the *randomly sampled* datasets. The other three histograms on the bottom row correspond to the datasets generated using the *selective sampling* methodology. Balanced samples — values greater than or equal to 0 — are represented in blue bins. Non-robust samples — negative values — are represented in red bins.

A quick analysis of the histograms on the top row clearly shows that the majority of the samples stored in the datasets are not balanced, and therefore are useless. Furthermore,

[3]In fact, a threshold of greater than or equal to 0 would have been enough to ensure this. However, increasing the threshold to 10 further improves the robustness quality of the overall dataset.

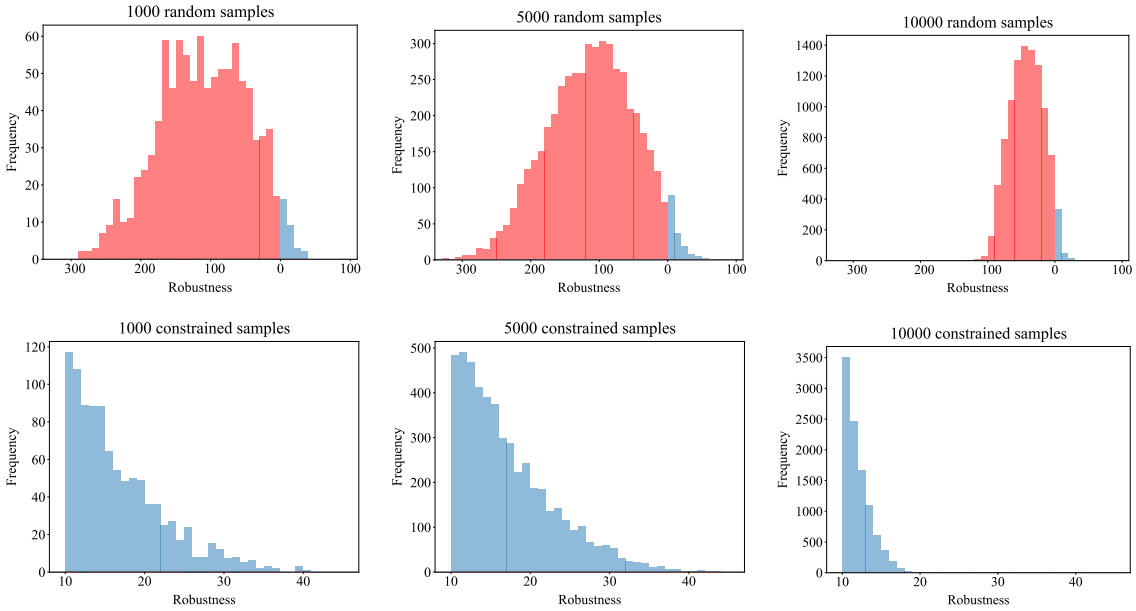


Figure 4.6: Robustness histograms of the benchmarked lower-body datasets. Negative robustness means the system can not be in equilibrium (red bins).

as the number of samples contained in the dataset increases, the more the curve of the histogram resembles a bell curve, *i.e.* a graph of a normal (Gaussian) distribution, which is expected. On the other hand, the histograms of the bottom row only contain balanced samples whose robustness is greater than or equal to 10. This is due to constrain (v), defined in the *selective sampling* method. The shape of the curve of these histograms appears to look like the right end of a normal distribution.

As anticipated, the *selective sampling* methodology is better than a *random sampling*, producing datasets comprised only of balanced configurations.

4.3 End-Pose Planning on Inclined Terrain

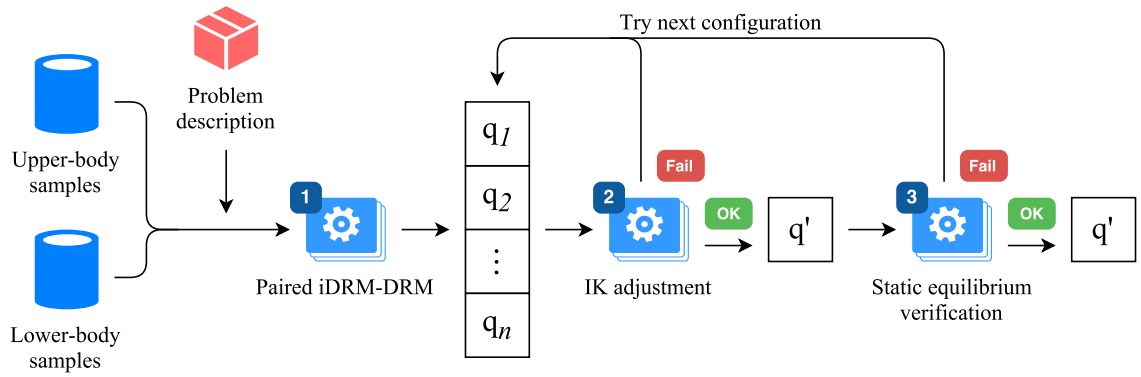


Figure 4.7: Pipeline overview of the proposed planning framework. The numbered blocks with a gear represent the key *stages* along the pipeline.

First of all, the pipeline of our existing planning framework requires two datasets generated offline: one containing upper-body samples and another containing lower-body samples. After having generated new lower-body datasets which contain samples for inclined supports, our goal is now to be able to actually do whole-body end-pose planning on inclined terrains. Ideally, we would like to have an environment like the one shown in Fig. 4.8 and be capable of finding a valid solution when a grasping task is requested. For that purpose, we need to adapt the current pipeline of our planning framework. A flowchart of the new adapted version is shown in Fig. 4.7 (p. 44).

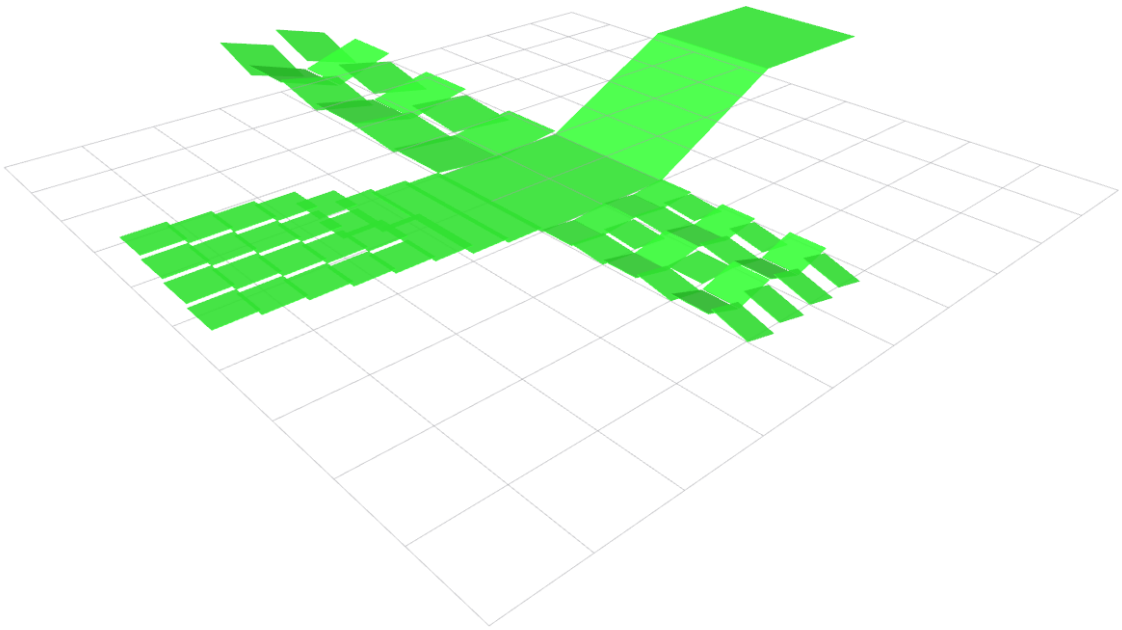


Figure 4.8: Test environment generated with our custom terrain parser[4].

When the system receives a request, such as a grasping target, the environment information is supplied to the algorithm pipeline — including scene obstacles and support regions. At the first stage — denoted by the blue rectangle with a gear, and labeled with a (1) on the top left corner in the flowchart of the framework — a combination of an upper-body iDRM and a lower-body DRM is used to find a list of whole-body samples that are collision-free and satisfy the task and the problem constraints. The sample at the head of the list proceeds in the pipeline, while the remainder of the list is kept in memory. Subsequently, the selected sample q is adjusted during stage two (2) using a non-linear optimisation-based IK solver [TD], resulting in sample q'. This adjustment is required to position the tip links *exactly* where they should be, *i.e.* the hands at the desired grasping

[4]We developed a very simple custom terrain parser during this part of the project in order to test our approach in multiple different inclined terrain scenarios with ease.

pose, and the feet perfectly aligned with the support regions. The initial and final states of an example of such a correction are shown in Fig. 4.9.

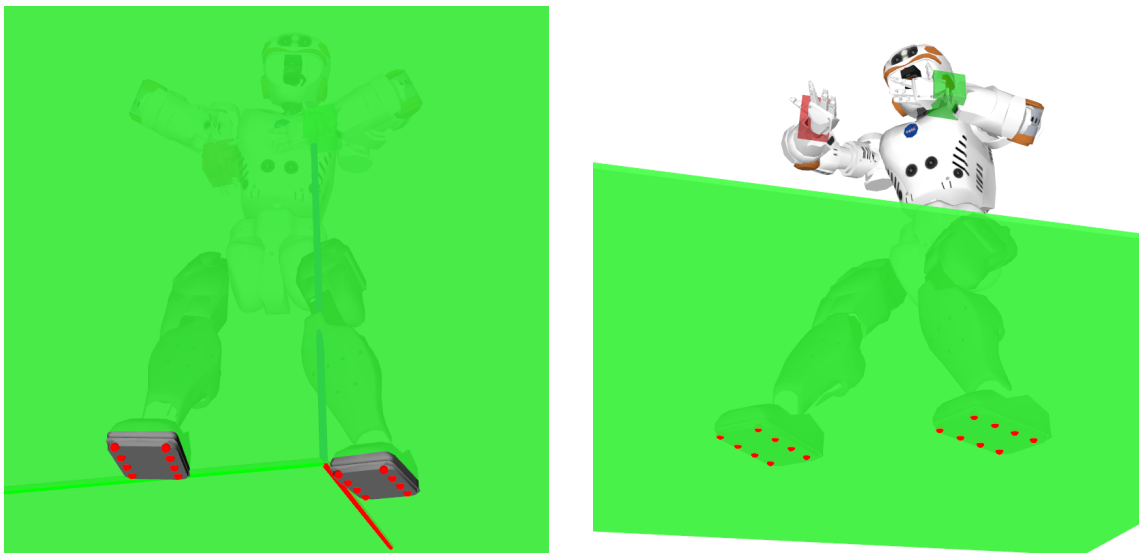


Figure 4.9: End-pose planning candidate solution before the final IK adjustment(left) and the resultant final solution after the IK corrections (right). Notice the correction of the position of the feet.

Finally, the static equilibrium robustness of the adjusted whole-body configuration q' is computed during stage three (3). This final robustness check is required due to the combinatorial nature of our planning approach, *i.e.* even though the lower-body sample of the solution is guaranteed to be balanced (otherwise it would have not been kept in the dataset during the selective sampling method described in 4.2.1, p. 43), the whole-body posture itself might not be in static equilibrium once the upper-body is combined with the lower-body sample. Once stage three (3) is finished and the whole-body configuration is confirmed to be in robust static equilibrium, the framework returns this configuration as the result to the given end-pose planning problem.

Sporadically, the IK solver may fail to perform the necessary adjustments to correct the sample q into q', whether due to unfeasible linear constraints or due to an unbounded problem objective, for example. In such cases, the sample is discarded and the flow goes back in the pipeline to immediately after stage (1), where the next sample of the previously saved list is popped out and then passed onto the pipeline. The same procedure occurs if q' is found out not to be in equilibrium at stage three (3) — the flow goes back in the pipeline and the next candidate pose in the list of candidates is passed on to the remainder of the pipeline.

4.4 Evaluation

4.4.1 End-Pose Planning Benchmarking Setup

Like we have previously described in 3.4.2 (p. 30), we wrote an auxiliary problem generator in order to effortlessly benchmark our method. The generator creates a set of problems with a grasping goal for each end-effector, feasible support regions for each leg, and multiple spherical obstacles. The problem generator takes in three arguments: (1) the number of problems to be generated, (2) the number of obstacles to be spawned, and (3) the minimum distance from the robot's kinematic structure at which the obstacles are allowed to be spawned.

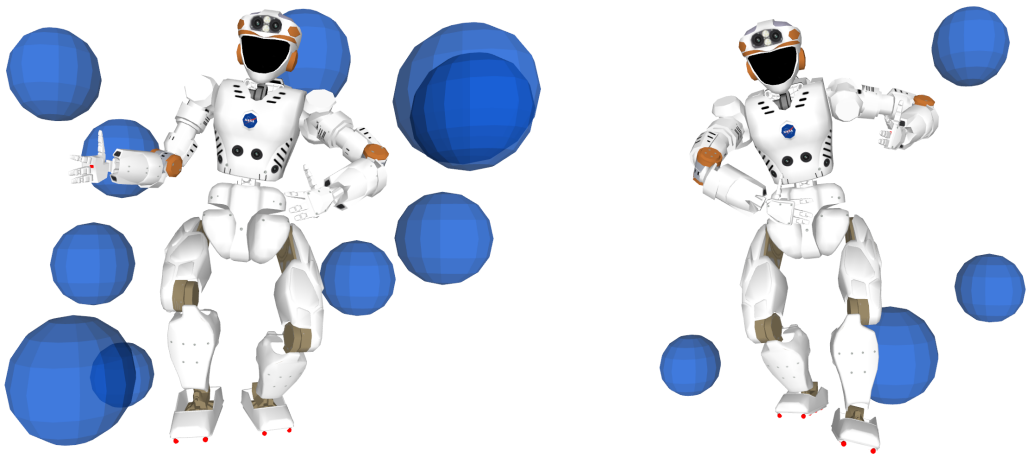


Figure 4.10: Example of two generated end-pose planning problems. Notice the increased number of obstacles of the problem on the left, which poses a greater planning challenge.

4.4.2 Simulation Benchmarking

To evaluate the performance of our end-pose planning framework on uneven and inclined terrains, we used the aforementioned problem generator to create six sets of problems, each set containing 1000 problem scenarios. The six problems fall into one of two categories: (a) problems where the obstacles are guaranteed to be at least 10 cm away from the robot kinematic structure used to generate that problem, and (b) problems where the obstacles are guaranteed to be at least 5 cm away from the robot. Category (b) is more challenging than (a), given that the solution's C-space is more constrained. Furthermore, for each category there exist 3 different problem sets: one containing problems with 5 spawned obstacles, another containing problems with 10 obstacles, and yet another consisting of problems with 20 obstacles.

Table 4.2 shows the success rate and the average planning time of each stage along the pipeline for each of the problem sets. The benchmarking was carried out using an upper-body dataset containing one million samples and a lower-body dataset containing ten thousand samples. The upper-body dataset was generated in the same manner as for

Chapter 3. The lower-body dataset was generated using the *selective sampling* approach described earlier in this chapter. The subset of the candidate list was created by selecting the first K samples of the set. Due to time constraints we did not have the chance to try the technique which improved the results of the former chapter as described in 3.4.3 (p. 31). The results are very poor when compared to the success rates obtained in that chapter.

Table 4.2: End-pose planning performance across different benchmark trials.

Min. distance to robot	No. obstacles	Success rate	Avg. iDRM time	Avg. IK time	Avg. REL time
10 cm	5	30.2 %	0.1270 s	7.0083 ms	0.8614 ms
	10	23.5 %	0.1953 s	6.5170 ms	0.8537 ms
	20	17.5 %	0.3114 s	6.6180 ms	0.8757 ms
5 cm	5	27.1 %	0.1246 s	6.9481 ms	0.8863 ms
	10	18.1 %	0.2028 s	6.7264 ms	0.8700 ms
	20	7.4 %	0.2661 s	6.6694 ms	0.8732 ms

As expected, we can see that the planning success rate decreases as the number of obstacles in the problem increases. Furthermore, by analysing the problems with the same number of obstacles but different minimum distance allowed between the robot and the obstacles during the problem generation, we can conclude that the success rate is always lower when the obstacles are allowed to be spawned closer during environment setup. Both these correlations are expected. After all, spawning more obstacles and allowing those obstacles to be closer to the configuration used to generate the problem, will further constraint the feasible regions of the solution to that problem.

Concerning the times of the different components, we can conclude that the iDRM runtime is the main bottleneck of the total planning time. Furthermore, the time spent by iDRM increases along with the number of obstacles in scene. The times required by both the IK solver and the robust equilibrium library (REL) — the method described in 4.1.1 — are reasonably consistent: the IK takes approximately 7 ms and the robustness evaluation of the whole-body poses 0.9 ms .

Table 4.3: Average robustness of the whole-body end-pose solutions to the benchmarking trials.

Min. distance to robot	No. obstacles	Avg. robustness
10 cm	5	20.73
	10	19.21
	20	19.84
5 cm	5	20.30
	10	19.47
	20	19.88

We have also collected information with regard to the robustness of the valid solutions returned by our planner. We can conclude that the average robustness does not vary

much depending on the problem settings. However, the results seem to indicate that the robustness of the selected candidate decreases slightly as the number of obstacles increases.

4.5 Summary

In this chapter we proposed an extension to our existing end-pose planning framework in order to not only support horizontal terrains at different heights, but also support inclined surfaces. In order to account for the additional circumstances characteristic of sloped surfaces, we generated a new lower-body dataset containing samples necessary to plan stances on inclined supports. Furthermore, we integrated utility classes into our framework to compute the static equilibrium robustness of whole-body end-poses on inclined contact geometries, given that this is substantially different from computing the quasi-static balance of stances on flat grounds.

Chapter 5

Conclusion

This chapter presents the conclusions of this research: we overview the work covered in this dissertation, followed by the list of the contributions resultant of this work, and finally, we review problems and issues yet to be solved, as well as some guidelines on how to tackle them.

5.1 Overview

In this dissertation we first reviewed the current state of the art in motion planning and end-pose planning. We then tried to approach the goals initially defined in chapter 1.3.

$\mathcal{G}_1$. **Extend previous work concerning end-pose planning to take into account flat supports at different heights.**

In Chapter 3 we proposed a whole-body kinematic split at the pelvis link in order to have a paired forward-inverse dynamic reachability map. The strength of both DRM and iDRM enabled us to explore the combinatorics of the modularity resultant from the kinematic split. By exploiting the combinatorics of this modularity the coverage of the whole-body C-space increased. As a result, it enabled us to effectively plan end-poses for grasping tasks on environments with support regions at different heights.

$\mathcal{G}_2$. **Further extend the whole-body end-pose planning framework resultant from $\mathcal{G}_1$ in order to support sloped terrains.**

In Chapter 4 we integrated utility classes to compute the static equilibrium robustness of whole-body configurations. This integration was necessary because the method

used to check for quasi-static balance of whole-body end-poses on a flat ground is substantially different from the method used to calculate the same on inclined contact geometries. As a result, we were able to generate a new and improved lower-body dataset, by biasing the sampling process with the new robustness metric. In turn, and in addition to our framework's previous capabilities, the extended lower-body dataset enabled us to find whole-body balanced configurations on sloped surfaces.

$\mathcal{G}_3$. **Evaluate the methodologies proposed to achieve $\mathcal{G}_1$ and $\mathcal{G}_2$ with simulation benchmarks, and conduct test trials in the laboratory's test bed.**

We developed a custom terrain parser and a problem generator in order to run extensive benchmarks and simulations to test the methods we proposed. We proved that we can indeed do end-pose planning on terrains at different heights, and on sloped surfaces. We conducted hardware experiments to prove the practicality of our approach using the 38-DoF NASA's Valkyrie in our test bed.

5.2 Contributions

The main contributions of this research include:

- A review of (i) the main SBP algorithms for motion planning, (ii) the theoretical concepts of RM, IRM and their respective dynamic versions, DRM and iDRM, and (iii) the state-of-the-art iDRM end-pose planning algorithm for complex and cluttered environments;
- An extension of iDRM, by combining RM for the lower-body with IRM for the upper-body, which allows the robot to select stance locations and whole-body configurations in complex and cluttered environments, namely with uneven flat terrains, in real time;
- Yet another extension to the iDRM end-pose planning algorithm, by improving the generation of lower-body samples and integrating a new technique to compute the static equilibrium robustness of whole-body poses on arbitrary contact geometries, which further extends the capability of the robot to plan end-poses on sloped terrains.

5.3 Future Work

5.3.1 Parallelisation

Since we decompose the kinematic structure of the humanoid into upper- and lower-body parts, the planner needs to combinatorially check each candidate pelvis pose with the lower-body DRM, which is time consuming and grows exponentially with the dataset size. Hence our future work involves using highly-parallelised architectures such as GPUs or other customised hardware to parallelise the combinatorial component in order to further reduce online planning time.

5.3.2 Compression

A current limitation of our method is the amount of memory required for storing the maps, *e.g.* 4.5GB for Valkyrie using the datasets Φ_{1c} and Φ_{4c}, considering the methodology presented in Chapter 3 (p. 21). Our future work involves investigating new methods of encoding the configuration-to-workspace mapping for better memory efficiency. This will allow us to increase the resolution of the voxel grid and improve the success rate of our method.

5.3.3 Selective Sampling Threshold

Regarding the static equilibrium robustness metric of the *selective sampling* described in 4.2.1, we chose to only keep samples in the dataset above a certain threshold — in our implementation we used 10 as threshold. Further work should be carried out to examine whether this is necessary or not. Moreover, if a threshold is required, a search for the optimum value for this threshold should be investigated.

5.3.4 "Extremely Robust" Lower-Body Samples

Concerning the abnormal equilibrium robustness classification of some lower-body samples discussed in Chapter 4 (p. 37), an investigation should be carried out in order to understand why those samples are being evaluated with such a high robustness measurement.

Acronyms

C-space Configuration space. 6, 7, 9, 11, 14, 16, 21, 25, 26, 32, 37, 47, 51, 54

CoM Centre of Mass. 38–41, 54, 57

DoF Degree of Freedom. 1, 10, 21, 25, 26, 52, 54, *Glossary:* degree of freedom

DRC DARPA Robotics Challenge. 54

DRM Dynamic Reachability Map. 21–24, 26, 29, 30, 32, 34, 35, 42, 45, 51, 52, 54

E-space End-effector space. 54

EPIA Portuguese Conference on Artificial Intelligence. 3, 54

EXOTica EXtensible Optimization Toolset. 20, 54

FCL Flexible Collision Library. 20, 54

FEUP Faculdade de Engenharia da Universidade do Porto. 54

FK Forward Kinematics. 11, 14, 28, 54

GIW Gravito-Inertial Wrench. 39, 54

iDRM inverse Dynamic Reachability Map. 2, 15–18, 21–26, 28, 29, 32–35, 45, 48, 51, 52, 54

IK Inverse Kinematics. 10, 12, 16, 19, 20, 31, 45, 46, 48, 54

IRM Inverse Reachability Map. 2, 13–15, 52, 54

LP Linear Programming. 39, 54

MATLAB MATrix LABoratory. 54

MIQCQP Mixed-Integer Quadratically-Constrained Quadratic Program. 54

MIT Massachusetts Institute of Technology. 2, 54

NASA National Aeronautics and Space Administration. 1, 2, 10, 21, 25, 35, 39, 40, 52, 54

Acronyms

NN Nearest Neighbour. 7, 54

PRM Probabilistic Roadmap. 6, 9, 54

QP Quadratic Programming. 39, 54

RA-L IEEE Robotics and Automation Letters. 3, 54

RM Reachability Map. 2, 11–15, 26, 27, 52, 54

ROS Robot Operating System. 54

RRT Rapidly-exploring Random Tree. 6–8, 54

SBP Sampling Based Planning. 6, 52, 54

SDP Semidefinite Programming. 54

SQP Sequential Quadratic Programming. 16, 28, 54

TCP Tool Centre Point. 11–14, 54

UoE University of Edinburgh. 2, 54

Glossary

curse of dimensionality The various phenomena that arise when analysing and organising data in high-dimensional spaces (often with hundreds or thousands of dimensions) that do not occur in low-dimensional settings such as the three-dimensional physical space of everyday experience [Wikb]. 2, 20, 54

degree of freedom The number of independent parameters that determine the state of a physical system [Wikc]. 1, 54

end effector The device at the end of a robotic arm, designed to interact with the environment. The exact nature of this device depends on the application of the robot [Wikd]. 15–18, 54, 57

jacobian The matrix relating joint velocities to end effector velocities. 19, 54, 57

kinematic singularity A point within the robot's workspace where the robot's jacobian matrix loses rank — in practice it is a point in the workspace where the robot loses its ability to move the end effector in some direction no matter how it moves its joints. It typically occurs when two of the robot's joints line up, making them redundant [HQ]. 12, 54

pose The combination of a *position* and *orientation*. 6, 14–16, 18, 54

quasi-static balance A robot standing on flat ground is quasi-statically balanced if its CoM projection lies within the support polygon with no velocity and acceleration along any axis. 16, 40, 49, 54

SE(2) 2D rigid transformations represented by a linear transformation on homogeneous 3-vectors. 54

SE(3) 3D rigid transformations represented by a linear transformation on homogeneous 4-vectors. 10, 14, 54

SO(2) 2D rotations represented by a 2D rotation matrix. 54

SO(3) 3D rotations represented by a 3D rotation matrix. 54

support polygon The convex hull of the set of contact points of a robot configuration. 54, 57

References

[Asf+06] Tamim Asfour, Kristian Regenstein, Pedram Azad, J Schroder, Alexander Bierbaum, Nikolaus Vahrenkamp, and Rüdiger Dillmann. ARMAR-III: An integrated humanoid platform for sensory-motor control. In *Humanoid robots, 2006 6th ieee-ras international conference on*. IEEE, 2006, pages 169–175 (Cited on p. 13).

[AW96] Nancy M Amato and Yan Wu. A randomized roadmap method for path and manipulation planning. In *Robotics and automation, 1996. proceedings., 1996 ieee international conference on*. Volume 1. IEEE, 1996, pages 113–120 (Cited on pp. 6, 9).

[Bar+97] Jérôme Barraquand, Lydia Kavraki, Jean-Claude Latombe, Rajeev Motwani, Tsai-Yen Li, and Prabhakar Raghavan. A random sampling scheme for path planning. *The international journal of robotics research*, 16(6):759–774, 1997 (Cited on p. 9).

[BB08] Sébastien Barthélemy and Philippe Bidaud. Stability measure of postural dynamic equilibrium based on residual radius. *Advances in robot kinematics: analysis and design*:399–407, 2008 (Cited on p. 39).

[BB15] Felix Burget and Maren Bennewitz. Stance selection for humanoid grasping tasks by inverse reachability maps. In *Proceedings of IEEE ICRA*, 2015, pages 5669–5674 (Cited on pp. 2, 13–15, 19, 34).

[BC95] Hee Rak Beom and Hyung Suck Cho. A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning. *Ieee transactions on systems, man, and cybernetics*, 25(3):464–477, 1995 (Cited on p. 6).

[CMO15] Marco Cognetti, Pouya Mohammadi, and Giuseppe Oriolo. Whole-body motion planning for humanoids based on CoM movement primitives. In *Humanoid robots (humanoids), 2015 ieee-ras 15th international conference on*. IEEE, 2015, pages 1090–1095 (Cited on p. 10).

[CPN15] Stéphane Caron, Quang-Cuong Pham, and Yoshihiko Nakamura. Leveraging Cone Double Description for Multi-contact Stability of Humanoids with Applications to Statics and Dynamics. In *Robotics: science and systems*, 2015 (Cited on p. 39).

[CS12] SAM Coenen and M Maarten Steinbuch. Motion planning for mobile robots – A guide. PhD thesis. Master Thesis, Mechanical Engineering, Eindhoven University of Technology, Eindhoven, 2012 (Cited on p. 6).

[Di +12] Giuseppe Di Gironimo, Luigi Pelliccia, Bruno Siciliano, and Andrea Tarallo. Biomechanically-based motion control for a digital human. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 6(1):1–13, 2012 (Cited on p. 38).

[DM15] Andrea Del Prete and Nicolas Mansard. Addressing constraint robustness to torque errors in task-space inverse dynamics. In *Robotics, sciences and systems 2015*, 2015 (Cited on p. 39).

[DT14] Robin Deits and Russ Tedrake. Footstep planning on uneven terrain with mixed-integer convex optimization. In *Humanoid robots (humanoids), 2014 14th ieee-ras international conference on*. IEEE, 2014, pages 279–286 (Cited on p. 29).

[DTM16] Andrea Del Prete, Steve Tonneau, and Nicolas Mansard. Fast algorithms to test robust static equilibrium for legged robots. In *Proceedings of IEEE ICRA*, 2016, pages 1601–1607 (Cited on pp. 39, 40).

[ES14] Mohamed Elbanhawi and Milan Simic. Sampling-based robot motion planning: A review. *Ieee access*, 2:56–77, 2014 (Cited on pp. 6, 8, 9).

[Fer+14] Hans Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Georg Bock, and Moritz Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical programming computation*, 6(4):327–363, 2014 (Cited on p. 39).

[Fer+17] Henrique Ferrolho, Vladimir Ivan, Yiming Yang, Wolfgang Merkt, Rosaldo J. F. Rossetti, and Sethu Vijayakumar. Whole-Body End-Pose Planning on Uneven and Inclined Surfaces (extended abstract). In *Portuguese Conference on Artificial Intelligence (EPIA) (under review)*, 2017 (Cited on p. 3).

[Fig+09] Miguel C Figueiredo, Rosaldo JF Rossetti, Rodrigo AM Braga, and Luis Paulo Reis. An approach to simulate autonomous vehicles in urban traffic scenarios. In *Intelligent transportation systems, 2009. itsc'09. 12th international ieee conference on*. IEEE, 2009, pages 1–6 (Cited on p. 1).

[FP96] Komei Fukuda and Alain Prodon. Double description method revisited. *Combinatorics and computer science*:91–111, 1996 (Cited on p. 39).

[Ger99] Michael Gerke. Genetic path planning for mobile robots. In *American control conference, 1999. proceedings of the 1999*. Volume 4. IEEE, 1999, pages 2424–2429 (Cited on p. 6).

[HLK06] David Hsu, Jean-Claude Latombe, and Hanna Kurniawati. On the probabilistic foundations of probabilistic roadmap planning. *The international journal of robotics research*, 25(7):627–643, 2006 (Cited on p. 9).

[HQ] Carl Henshaw and Quora. Robotics: What is meant by kinematic singularity? — Quora. [Online; accessed 04-July-2017]. URL: https://www.quora.com/Robotics-What-is-meant-by-kinematic-singularity/answer/Carl-Henshaw (Cited on p. 57).

[Iva+] Vladimir Ivan, Yiming Yang, Michael Camilleri, Wolfgang Merkt, and Sethu Vijayakumar. EXOTica: an extensible optimization toolset for motion planning and control prototyping and benchmarking. [Online; accessed 15-June-2017]. URL: https://github.com/openhumanoids/exotica (Cited on p. 20).

[Kav+96] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Ieee transactions on robotics and automation*, 12(4):566–580, 1996 (Cited on pp. 6, 9).

[KL00] James J Kuffner and Steven M LaValle. RRT-connect: An efficient approach to single-query path planning. In *Robotics and automation, 2000. proceedings. icra'00. ieee international conference on*. Volume 2. IEEE, 2000, pages 995–1001 (Cited on pp. 6–8).

[KL94] Lydia Kavraki and J-C Latombe. Randomized preprocessing of configuration for fast path planning. In *Robotics and automation, 1994. proceedings., 1994 ieee international conference on*. IEEE, 1994, pages 2138–2145 (Cited on pp. 6, 9).

[Kuf+01] James Kuffner, Koichi Nishiwaki, Satoshi Kagami, Masayuki Inaba, and Hirochika Inoue. Motion planning for humanoid robots under obstacle and dynamic balance constraints. In *Robotics and automation, 2001. proceedings 2001 icra. ieee international conference on*. Volume 1. IEEE, 2001, pages 692–698 (Cited on p. 10).

[Kuf04] James J Kuffner. Effective sampling and distance metrics for 3D rigid body path planning. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*. Volume 4. IEEE, 2004, pages 3993–3998 (Cited on p. 16).

[Lat12] Jean-Claude Latombe. *Robot motion planning*. Volume 124. Springer Science & Business Media, 2012 (Cited on p. 5).

[LaV98] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning, 1998 (Cited on p. 6).

[LK01] Steven M LaValle and James J Kuffner Jr. Randomized kinodynamic planning. *The international journal of robotics research*, 20(5):378–400, 2001 (Cited on pp. 6, 7).

[Mae+10] Takashi Maekawa, Tetsuya Noda, Shigefumi Tamura, Tomonori Ozaki, and Ken-ichiro Machida. Curvature continuous path generation for autonomous vehicle using B-spline curves. *Computer-aided design*, 42(4):350–359, 2010 (Cited on p. 6).

[Ott+06] Ch Ott, Oliver Eiberger, Werner Friedl, B Bauml, Ulrich Hillenbrand, Ch Borst, Alin Albu-Schaffer, Bernhard Brunner, H Hirschmuller, S Kielhofer, et al. A humanoid two-arm system for dexterous manipulation. In *Humanoid robots, 2006 6th ieee-ras international conference on*. IEEE, 2006, pages 276–283 (Cited on p. 11).

[PCM12] Jia Pan, Sachin Chitta, and Dinesh Manocha. FCL: a general purpose library for collision and proximity queries. In *Robotics and automation (icra), 2012 ieee international conference on*. IEEE, 2012, pages 3859–3866 (Cited on p. 20).

[Pet+11] Marcelo Petry, Antonio Paulo Moreira, Luis Paulo Reis, and Rosaldo Rossetti. Intelligent wheelchair simulation: Requirements and architectural issues. In *11th international conference on mobile robotics and competitions, lisbon*, 2011, pages 102–107 (Cited on p. 1).

[PR12] José LF Pereira and Rosaldo JF Rossetti. An integrated architecture for autonomous vehicles simulation. In *Proceedings of the 27th annual acm symposium on applied computing*. ACM, 2012, pages 286–292 (Cited on p. 1).

[Qiu+11] Zhapeng Qiu, Adrien Escande, Alain Micaelli, and Thomas Robert. Human motions analysis and simulation based on a general criterion of stability. In *International symposium on digital human modeling*, 2011, pages 1–8 (Cited on p. 39).

[SRR11] Nima Shafii, Luis Paulo Reis, and Rosaldo JF Rossetti. Two humanoid simulators: Comparison and synthesis. In *Information systems and technologies (cisti), 2011 6th iberian conference on*. IEEE, 2011, pages 1–6 (Cited on p. 1).

[TD] Russ Tedrake and the Drake Development Team. Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems. [Online; accessed 15-June-2017]. URL: http://drake.mit.edu/ (Cited on pp. 16, 28, 45).

[Tea] Open Humanoids Project Development Team. Open Humanoids Project: University of Edinburgh and MIT's Valkyrie and Atlas Humanoid Project. [Online; accessed 15-June-2017]. URL: https://github.com/openhumanoids/ (Cited on p. 2).

[Tog86] Masaki Togai. An application of the singular value decomposition to manipulability and sensitivity of industrial robots. *Siam journal on algebraic discrete methods*, 7(2):315–320, 1986 (Cited on p. 14).

[Too] The Engineering ToolBox. Friction and Friction Coefficients. [Online; accessed 26-June-2017]. URL: http://www.engineeringtoolbox.com/friction-coefficients-d_778.html (Cited on p. 40).

[VAD13] Nikolaus Vahrenkamp, Tamim Asfour, and Rüdiger Dillmann. Robot placement based on reachability inversion. In *Robotics and automation (icra), 2013 ieee international conference on*. IEEE, 2013, pages 1970–1975 (Cited on pp. 2, 13, 14).

[Wika] Wikipedia. Autonomous robot — Wikipedia, The Free Encyclopedia. [Online; accessed 04-July-2017]. URL: https://en.wikipedia.org/wiki/Autonomous_robot (Cited on p. 5).

[Wikb] Wikipedia. Curse of dimensionality — Wikipedia, The Free Encyclopedia. [Online; accessed 04-July-2017]. URL: https://en.wikipedia.org/wiki/Curse_of_dimensionality (Cited on p. 57).

[Wikc] Wikipedia. Degrees of freedom (mechanics) — Wikipedia, The Free Encyclopedia. [Online; accessed 04-July-2017]. URL: https://en.wikipedia.org/wiki/Degrees_of_freedom_(mechanics) (Cited on p. 57).

[Wikd] Wikipedia. Robot end effector — Wikipedia, The Free Encyclopedia. [Online; accessed 04-July-2017]. URL: https://en.wikipedia.org/wiki/Robot_end_effector (Cited on p. 57).

[Yan+16a] Yiming Yang, Vladimir Ivan, Zhibin Li, Maurice Fallon, and Sethu Vijayakumar. iDRM: Humanoid motion planning with realtime end-pose selection in complex environments. In *Proceedings of IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2016, pages 271–278 (Cited on pp. 2, 10, 15, 16, 18–20, 24, 34).

[Yan+16b] Yiming Yang, Vladimir Ivan, Wolfgang Merkt, and Sethu Vijayakumar. Scaling Sampling-based Motion Planning to Humanoid Robots. *Arxiv preprint arxiv:1607.07470*, 2016 (Cited on p. 29).

[Yan+17] Yiming Yang, Wolfgang Merkt, Henrique Ferrolho, Vladimir Ivan, and Sethu Vijayakumar. Efficient Humanoid Motion Planning on Uneven Terrain Using Paired Forward-Inverse Dynamic Reachability Maps. In *IEEE Robotics and Automation Letters (in press)*, 2017 (Cited on p. 3).

[Yos85] Tsuneo Yoshikawa. Manipulability of robotic mechanisms. *The international journal of robotics research*, 4(2):3–9, 1985 (Cited on p. 14).

[ZBH07] Franziska Zacharias, Christoph Borst, and Gerd Hirzinger. Capturing robot workspace structure: representing robot capabilities. In *Proceedings of IEEE IROS*, 2007, pages 3229–3236 (Cited on pp. 2, 11–13).