Motion planning and synthesis for virtual characters in constrained environments

Steve Tonneau

Thèse

THESE INSA Rennes
sous le sceau de l'Université européenne de Bretagne
pour obtenir le titre de
DOCTEUR DE L'INSA DE RENNES
Spécialité : Informatique

présentée par
Steve Tonneau
ECOLE DOCTORALE : *Matisse*
LABORATOIRE : *IRISA*

Synthèse et planification de mouvement pour des personnages virtuels en environnements contraints

Motion planning and synthesis for virtual characters in constrained environments

Thèse soutenue le 27.02.2015
devant le jury composé de :

Bruno Arnaldi
Professeur à l'INSA Rennes / Président
Jean-Paul Laumond
Directeur de recherche au LAAS-CNRS / Rapporteur
Marcelo Kallmann
Professeur associé à l'université de Californie / Rapporteur
Taku Komura
Professeur associé à l'université d'Edimbourg / Examinateur
Arjan Egges
Professeur associé à l'université d'Utrecht / Examinateur
Julien Pettré
Chargé de recherche à Inria Rennes / Co-encadrant de thèse
Franck Multon
Professeur à l'université de Rennes 2 / Directeur de thèse

Remerciements

(English version follows for the jury members)

Premièrement je voudrais remercier les membres de mon jury: Jean-Paul Laumond, Marcelo Kallmann, Bruno Arnaldi, Taku Komura et Arjan Egges. Je vous suis reconnaissant d'avoir accepté ma requête, et honoré du réel investissement qui a été le vôtre, s'agissant de la relecture du manuscrit, des discussions avant et après la soutenance, ou de votre suivi dans la continuité de mes travaux.

First of all, I would like to thank the members of my jury: Jean-Paul Laumond, Marcelo Kallmann, Bruno Arnaldi, Taku Komura and Arjan Egges. I'm sincerely grateful and honored in many ways: I really appreciated the quality of your feedback on the manuscript, the presentation, as well as the discussions we had before and after the defense.

Ce ne sont pas les étudiants de Mimetic / Hybrid qui me contrediront: des observations menées sur une population masculine de 3 doctorants en informatique, âgés de 23 à 28 ans, se généralisent naturellement à l'ensemble de la population de la planète.

Aussi, mon unique expérience de 3 ans dans un laboratoire de recherche me permet de tirer de grandes conclusions quant à la vie du doctorant, et celle de chercheur en général.

La première, c'est qu'il est plus important de choisir un bon encadrant qu'un sujet de thèse pertinent. Pour celà, une méthode infaillible (100% de réussite sur la population étudiée): écrire un mail à tout l'IRISA en disant qu'on veut faire une thèse, sans donner plus de précisions. Le seul à réagir positivement à ce monument de naïveté s'appellera à coup sûr Julien Pettré, et c'est lui qu'il faudra remercier en premier, pour son amitié, pour la confiance accordée, la patience et l'obstination dont il a fait preuve pour l'obtention de notre bourse, mais aussi pour la patience et l'obstination dont il a fait preuve à mon égard: tu l'as eu ton planner, et c'est pas dommage!

La deuxième conclusion, c'est qu'un bon directeur de thèse, c'est pas mal non plus. Si sur son calendrier, malgré le ciel qui lui tombe sur la tête (au sens propre), et un minimum syndical de 2 réunions à la même heure par jour, il se débrouille toujours pour caler le nom de ces thésards, foncez bille en tête: avec une probabilité de 1, il s'agit de Franck Multon. Merci à lui pour nos interactions (actes à paraitre sous le nom "humour fin de sportif, Star Wars et cinématique inverse").

L'autre certitude avec la thèse, c'est que pour l'accomplir, il faut les meilleurs dans son équipe: Fabien (mon pilier, et pas que de bar, c'est rien de le dire), Ferran (best

party planner ;)), Anne Hélène (Ctrl Alt ... J ???), Teofilo (Go Finland!), Rozenn (je reviens chaque année jusqu'à ce qu'on gagne ce quiz!), Manu (soutien indéfectible à son colloc à terre), Kevin (tout est dans le prénom ;)), Merwan (la force tranquile), Julien B (vainqueur par ko de Ben Stiller au défi défilé), David (chips parce que poivron), Hui-Yin (best puzzlescript award), Ana Lucia (officemate of the year), Huyen (bonne année!), Loiez (mon clone), Nathalie (parce que tout), Marc (je te pardonne), Fabrice (encore merci pour l'arbre), Charles (on va finir par le faire cet article).

Parmi les "extérieurs rennais", mention spéciale à Sylvain (mon deuxième pilier) Fanny, Anna, Hélène, et last but not least Béné.

Pas besoin de remercier mes autres amis, vous ne lirez jamais ces remerciements, et c'est bien pour ça que je vous aime ;).

Merci à mes parents et à mes frères, vous savez pourquoi.

Et merci à Anneli. Ca, je ne le dirai jamais assez.

Contents

Chapter 1

Introduction

With the growing complexity of virtual environments comes the need to provide a virtual character with a larger set of motion capabilities. Additionally to walking, running and jumping, state of the art virtual applications require characters to climb, crawl, pull or push objects... In the video game Assassin's creed ^{TM} for instance, a character, controlled by the player or the computer, can climb among the walls of an apparently large variety of buildings or navigate through the giant trees of a forest (Figure 1.1 - left).

Figure 1.1: Left: Examples of tree navigation and climbing motions in the video game Assassin's creed. © Ubisoft. Right: The bipedal robots Chimp and Schaft at the DARPA challenge. © DARPA challenge.

Similarly, in the robotics field, humanoid robots leave the safety of research laboratories and are required to navigate among complex environments. An illustration is

the DARPA challenge, where robots are expected to perform complex tasks such as removing debris while navigating across uneven terrains (Figure 1.1 - right).

An issue common to robotics and computer graphics is therefore to provide a virtual character (or a robot) with the ability to automatically perform a given motion task in a constrained environment. In this context, the challenges are numerous: addressing this issue requires finding a natural looking, feasible trajectory among the obstacles. The trajectory must prevent the character (robot) from losing balance or colliding with the environment, while creating the contacts necessary to achieve the motion.

This is the issue we address in this thesis, with a focus on computer graphics applications.

Motivation and objectives

The computer graphics and robotics fields have addressed autonomy of motion in different ways, each one reflecting the primary objectives that were pursued.

In an interactive context such as video games, the main concern is performance: a simulation must be reactive to the inputs of the player. Additionally, the quality of the animations is essential. Any unnatural animation or artefact would immediately be noticed and have a negative impact on the user experience.

For these reasons, a majority of techniques in the computer graphics field belongs to the family of *example-based* methods. In this approach a large part of the work is done before the execution of the simulation. Each motion is carefully hand designed by an animator. Another option is to record the motion performed by a real human being using a motion capture system (Figure 1.2). The animation is then replayed when the simulation triggers it. This approach provides animators with a complete control of the motion style, since it is the same as the one recorded. The resulting motion is not necessarily realistic or physically accurate, but it is accepted as **plausible** by the user.

Figure 1.2: Motion capture systems allow to replay the motions of real actors in a virtual environment. © Quantic dreams.

However the drawback of these approaches is that they constrain the design of the environment to fit to the recorded motion. Some motion adaptation methods exist (see for instance [WP95, KGP02]), but they only allow small variations from the original

motion. This can be seen in a state of the art video game such as Assassin's Creed ™ (Figure 1.1): A climbing character can only move along a vertical or horizontal line (no diagonal motions), and the height separating two grasping points is fixed, triggering a specific recorded motion to reach it. In this case the diversity is brought by the work of the artists who managed to design various environments which all respect the same constraints. Similarly, the geometry of the forest trees quickly appears stereotypical.

Providing virtual characters with more autonomy of motion would allow to remove the constraints associated with the design of interactive virtual environments, give more freedom to the artists and enhance the player experience. A valid solution should be fast and produce plausible animations.

Conversely, some robots are ultimately designed to navigate in unknown environments (an extreme example being the exploration of Mars by the robot Curiosity). The main concern of robotics is to provide robots with capacities of reasoning and planning. Robotics contributions have proposed a large variety of motion planners: [KSLO96, LaV98, HBL05, BRL$^+$04, EKMG08]. Motion planners are designed to compute a physically accurate trajectory which allows a robot to navigate through an unknown environment. The main strength of such planners is their genericity: these procedural methods can find solutions in a large variety of environments. However the main drawback is that the resulting motions often look unnatural to a human observer. This makes them hard to adapt to computer graphics applications.

Providing robotics motion planners with means to synthesize more natural looking motions while preserving their robustness in constrained environments would make them suited for computer graphics applications.

Objective examples: We consider two kind of applications, challenging for the existing motion techniques. On one hand we consider real time, reactive contact generation (Figure 1.3); on the other hand we consider global motion planning (Figure 1.4).

In Figure 1.3 for instance, a virtual character is controlled by a human player. His input triggers the task of pushing the cupboard (green overlining). In this context we address the following issue: generating as fast as possible a target contact configuration allowing to achieve the task, even under strong environmental constraints.

The two scenes shown in Figure 1.4 present a climbing wall, and we consider the task of reaching its top. The location of the potential grasps differ from one scene to another.

These examples are challenging when no assumption on the environment can be made, because it is impossible to precompute the motion. Indeed, naive example based approaches could only provide a solution if a motion had been recorded (or hand designed) for each setup, because the sequence of contacts necessary to achieve the objective varies too much. A motion planner might find a solution in every case, but it would not look natural.

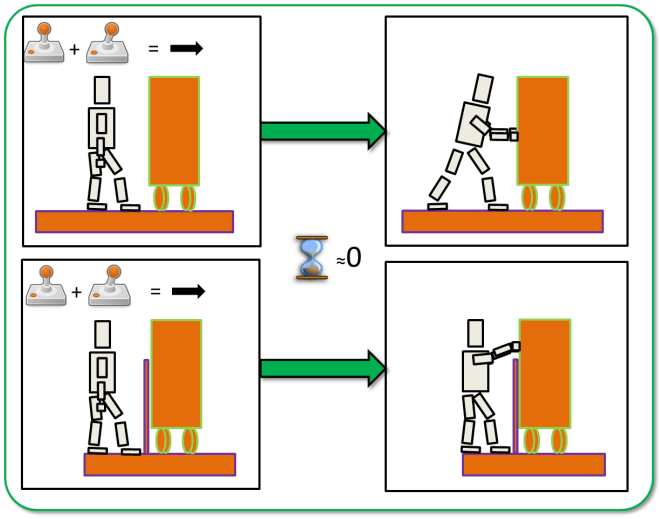

Figure 1.3: We want to address the issue of generating relevant contacts for pushing the cupboard within real time constraints, in spite of the constraints of the environment (bottom).

To sum up, the examples we address are characterized by the following constraints:

- They are achieved along constrained environments, presenting high risk of collision with external elements (a climbing wall, a chair and a table...);
- They require important force exertion (climbing, standing up...);
- They are performed through an acyclic sequence of contacts: contrary to cyclic motions such as walking (for a human, after stepping with one foot, the next step is always performed with the other foot), no assumption can be made about the next contact that must be created.

Therefore in this context, a useful motion planner should be able to:

- Find solutions in unknown constrained environments, with numerous obstacles and narrow passages;
- Generate **task efficient** contact configurations which, in our case, allow to exert important forces (Figure 1.5);
- Compute acyclic, non deterministic sequences of contacts.

Therefore, the objective of this thesis is to propose a planner combining those three properties.

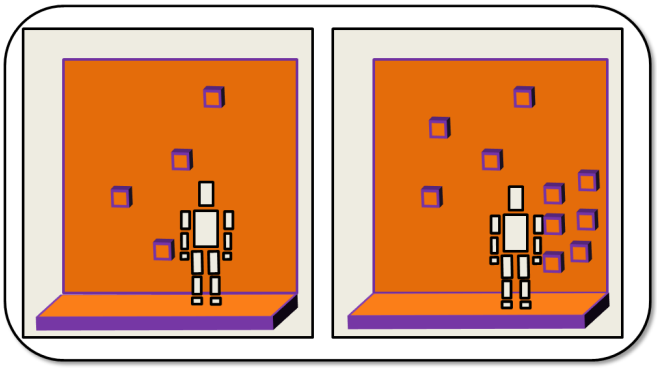

Figure 1.4: A challenging climbing scenario for existing animation methods.

Figure 1.5: The importance of task efficiency is illustrated in this standing up task. We need to propose a heuristic that would allow to choose the configurations for the limbs displayed on the right panel, which seem more suited for a vertical force exertion task.

Contributions

We propose a framework to address our objective examples. In this thesis we consider reactive contact generation (Figure 1.3) as a sub question of the more complex problem of motion planning.

Therefore, our first step is the proposal of a real time generator for contact configurations. Given a current configuration in the environment and a direction of motion, our method computes a suitable contact configuration for the character. The method does not require example motions, and is therefore generic and automatic. To achieve this we propose a new heuristic called the Extended FORce Transmission ratio, or EFORT.

Our second step is the design of a global motion planner built upon this contact generator (Figure 1.6). As an input, the user provides a task, described as a start configuration (Figure 1.6 - purple character) and a goal configuration (Figure 1.6 - blue character). Our framework outputs a contact trajectory described by a sequence of task efficient contact configurations (Figure 1.6 - middle). It is then refined into a dynamically stable contact trajectory allowing the virtual character to perform the task (Figure 1.6 - right).

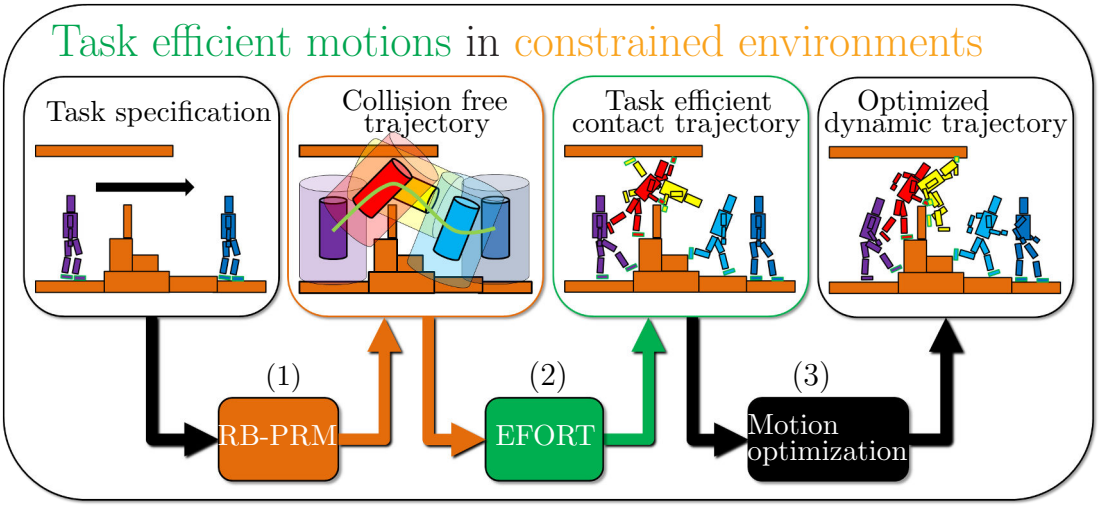

Figure 1.6: A three stage framework for the automatic computation of contact trajectories in constrained environments.

The framework is divided in three stages:

- In the first stage, we propose a motion planning algorithm which rapidly computes a promising trajectory to perform the motion task (Figure 1.6 - 1). The planner is called RB-PRM, which stands for Reachability Based Probabilistic Road Map. Using a character abstraction (represented by the cylinders in the Figure), RB-PRM samples configurations which are close enough to the environment to allow the creation of contacts while avoiding collision. It uses them to compute a collision free path that connects the start and goal configurations. At this stage, no contacts are created. RB-PRM provides our planning solution with the first required property: thanks to its probabilistic approach, it can find solutions in constrained environments. In the rest of this thesis, all figures related to RB-PRM are identified with an orange overline.

- In the second stage the collision free trajectory is transformed into a sequence of **task efficient** contact configurations (Figure 1.6 - 2), using our contact generator. Thanks to this step, our planner is provided with the two other properties: it can compute a sequence of acyclic, task efficient contact configurations. In the rest of this thesis, all figures related to EFORT are identified with a green overline.

- In the third stage this trajectory is used as a guide to an optimization based motion planner method. This allows to produce a smoother and more balanced trajectory (Figure 1.6 - 3). To achieve this last stage, we use the ITOMP optimization framework [PPM12]. This integration is done in the context of an ongoing collaboration with the University of North Carolina (UNC). The key objective is to demonstrate that the contributions we propose can provide such frameworks with the capacity of generating motions in constrained environments.

This framework allows us to propose two contributions:

Contribution 1: With EFORT we provide a method for the real time generation of task efficient contact configurations. This effectively enhances the autonomy of motion of virtual characters controlled by the player. EFORT can be used independently from our framework to be integrated within any animation solution. In a video game application, it could compute appropriate configurations in arbitrary geometries. This would allow more variation in the environments proposed and enhance the user experience.

Contribution 2: We propose a three stage framework to compute contact trajectories in constrained environments (Figure 1.6). By biasing the search of a valid trajectory with simple heuristics we are able to rapidly generate a promising guess in any environment. This allows us to provide plausible motions in constrained environments for virtual characters controlled by the simulation.

Summary of chapters

This thesis is organized as follows:

In Chapter 2 we provide a review of the methods proposed in robotics and computer graphics relative to motion synthesis, with a focus on constrained environments. We consider our objective examples and analyze how state of the art methods perform against them.

In Chapter 3, we give an overview of the methods proposed in this thesis. Each stage is then precisely detailed in their specific chapter.

In Chapter 4, we give several notation conventions and mathematical definitions used throughout the thesis; We also describe in details the character models used.

In Chapter 5, we present a method for the real time generation of task efficient contact configurations. We give the details of the method and explain how it can be integrated within an existing animation system;

In Chapter 6, we present the Reachability-Based Probabilistic Road Map, a motion planner designed to efficiently compute promising collision free trajectories in an arbitrary constrained environments (Figure 1.6 - 1);

In Chapter 7, we detail the second and third stages of our framework. First we detail how the EFORT method is used to transform a collision free trajectory into a contact trajectory (Figure 1.6 - 2); Then we present the trajectory optimization framework used in the final stage (Figure 1.6 - 3).

In Chapter 8, we present the results obtained with our framework, and discuss the ongoing integration with the ITOMP framework;

Finally, this thesis is concluded in Chapter 9. We discuss the advantages and limits of our framework, and present the improvements we intend to develop in future work.

Chapter 2

Related work

Motion synthesis has been addressed by many scientific fields and been investigated for many years, resulting in numerous contributions and formalisms. The selective review we propose in this chapter focuses on automatic synthesis of plausible motions for virtual characters in constrained environments (Figure 2.1). In so doing, we explore contributions made by different fields:

- Research in the robotics field has introduced the notion of configuration space, a useful formalism which allows to characterize mathematically the motion planning problem. Several algorithms were proposed to explore this high dimensional space. Over time, these algorithms have been specialized to address specific issues, such as humanoid motion in constrained environments. In section 2.1, we underline several properties which make this class of algorithms appealing for our problem. However, we also underline the fact that those methods do not consider the naturalness of the resulting motion.
- In section 2.2, we review the computer graphics contributions which address the issue of synthesizing natural motions in constrained environments. The most popular approaches are called "example based methods". Instead of exploring the configuration space, example based approaches restrain the search for a solution to a database of reference motions, considered to be natural. In order to adapt these motions to new environments, they are concatenated, blended or deformed. We show that while they manage to synthesize natural motions, example based approaches do not present the properties of completeness proposed by robotics planners. Therefore they cannot be adapted to constrained environments.
- Finally, in section 2.3 we consider another class of approaches for synthesizing natural motions. As opposed to example based (or data driven) methods, several

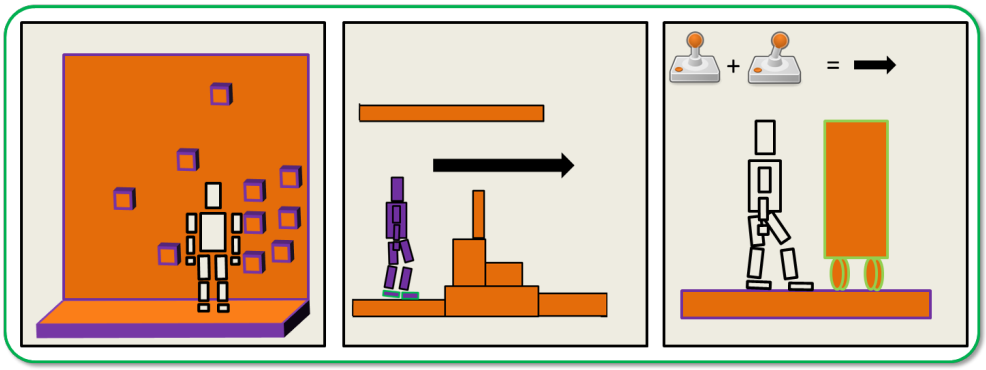

Figure 2.1: Our three objective scenarios: climbing a wall, crossing an obstacle race and pushing a cupboard. In the pushing scenario, the task can be formulated interactively by a player using a joystick.

contributions use a model approach, inspired from biomechanics. We observe that the definition of a natural motion varies with the objectives and review several models for achieving human-like motions. We show that these contributions address cyclic motions or manipulating tasks, but that they are rarely integrated within full body motion planners for constrained environments, which is one of the goals of this thesis.

In section 2.4, this chapter is concluded with a summary of the reviewed contributions, a comparison of their properties regarding the requirements of our problem.

2.1 Robotics contributions for synthesising motions in constrained environments

To synthesize motion in constrained environments, it is necessary to compute a collision free trajectory between a starting position and a goal position. This issue is commonly known as the motion planning problem. A classic illustration is given by the "piano movers" problem, formulated as follows in [SS83]: "Given a body B, and a region bounded by a collection of "walls", either find a continuous motion connecting two given positions and orientations of B during which B avoids collision with the walls, or else establish that no such motion exists." The complexity of the problem stems from the infinity of postures that a character can take in a given environment. In this section we review the solutions proposed in the robotics field, based on the configuration space formalism. We start by explaining the formalism, before describing the early geometric algorithms proposed to explore it, upon which modern contributions are based. We then review how these algorithms were modified to specifically address motion synthesis in constrained environments. We finally present the contributions which explored the configuration space in order to synthesize balanced trajectories for humanoid robots.

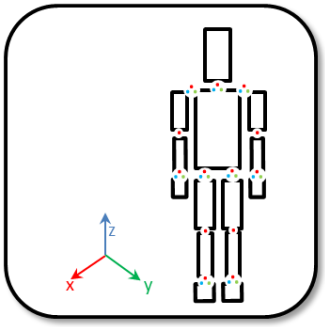

Figure 2.2: A humanoid robot with 31 + 6 = 37 degrees of freedom. Each colored dot represents a degree of freedom around an axis.

2.1.1 Geometric motion planning

The most common formulation of the motion planning problem is proposed by Lozano-Pérez, who introduces the notion of configuration space in [LP83]. We consider a kinematic chain composed of N degrees of freedom, that we call a **robot**, or a **virtual character**. An example of robot is given in Figure 2.2. The position and orientation of all the segments of the robot, which we call a **configuration**, can be described by a point $\mathbf{q}$ in the configuration space $C \subset R^N$. We define $C_{free} \subset C$ as the space of configurations that do not collide with the environment (Figure 2.3 - left). Similarly, $C_{obs} \subset C$ is the space of colliding configurations (Figure 2.3 - right).

With these definitions, the geometric issue of finding a collision free trajectory in a 3D environment (or workspace), is formulated into the search of a trajectory $q(t) \in C_{free}$ between two points in the configuration space (Figure 2.4).

Based on this notion of configuration space, a lot of algorithms have been developed, of which Lavalle's book [LaV06] stands as the most recent survey. Probabilistic algorithms have demonstrated their ability to deal with complex environments, comprising numerous obstacles and narrow passages. The most famous probabilistic planners are the Probabilistic Road Maps (PRM) [KSLO96] and the Rapidly exploring Random Trees (RRT) [LaV98]. One strong property of probabilistic planners is the completeness of their algorithm: if a solution exists, the probability of finding it converges to 1, given an infinite amount of time.

Probabilistic methods generate random configurations in C_{free} and connect them within a graph structure, where the nodes represent the sampled configurations (Figure 2.5). Two nodes are connected according to a given proximity predicate, usually called a "local steering method". It uses basic means such as linear interpolation to assert that a continuous, collision free path exists between the connected configurations.

To find a path between a start configuration and a goal configuration, the graph is searched. If a solution exists, it is returned as a discrete sequence of configurations which indicates how to traverse the graph.

Probabilistic methods can be used to find paths in unknown environments. However,

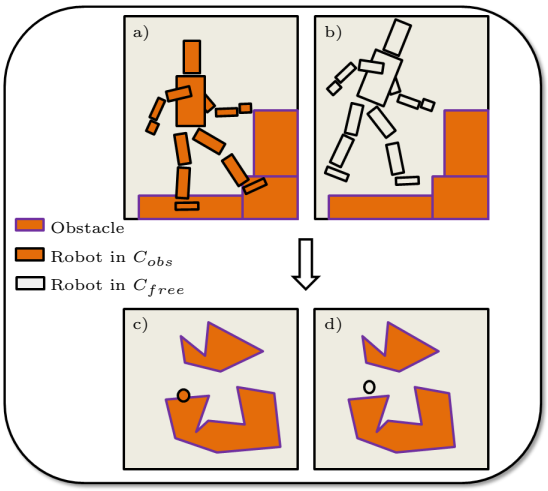

Figure 2.3: a) The configuration of the orange robot belongs to C_{obs} because it collides with the obstacles; b) the configuration of the grey robot belongs to C_{free} because it does not collide with the obstacles; c),d) an abstract 2D representation of the high dimensional configuration space of the robot.

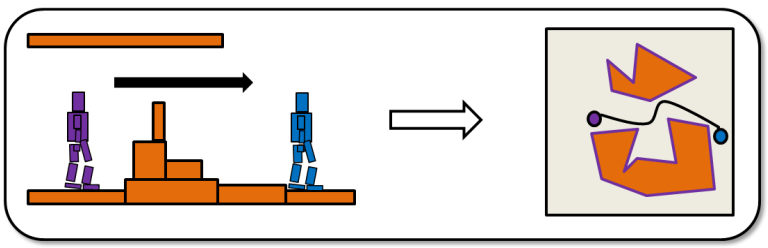

Figure 2.4: Motion planning can be viewed as the search of a collision free path between two points of the configuration space.

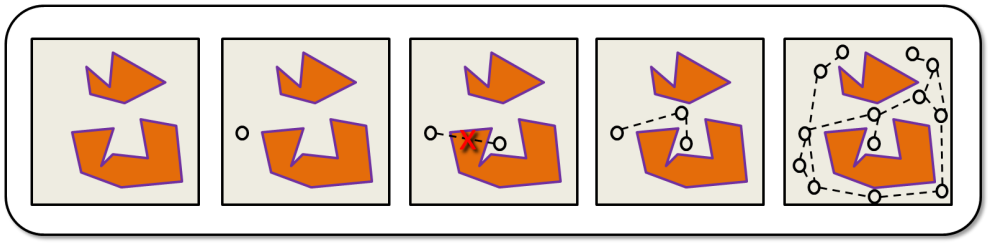

Figure 2.5: An example of the generation of a PRM graph. Configurations are randomly sampled until a given condition is met (such as a target number of configurations). Two configurations are connected together if a collision free path can be found between them.

they need adaptation to be efficient in constrained environments: in the presence of a lot of obstacles, performances of uniform sampling strategies can be dramatically affected due to the high probability of generating configurations in collision with the environment.

2.1.2 Planning in constrained environments

Interesting variations of the generic probabilistic motion planners algorithms have been developed to address specific cases and applications. Specifically, several contributions have been proposed for manipulator robots. Such robots are deployed in factories, where they move their robotic arm to perform assembly tasks. In these constrained environments, the risk of collision is high. But additionally to finding a collision free path between the effector and a target location, manipulation planners have to consider the objects being manipulated [SLCS04]. Object manipulation is a hard problem with several specificities, and is not studied in this work.

In [KM04], Kallmann et al. introduce the dynamic roadmap approach for manipulator robots. The idea is to generate a roadmap in the reachable workspace of a given limb of a robot, independently of the environment. At runtime, when a path from a configuration to another is requested, the roadmap is updated accordingly to the current situation: upon request if the chosen path is colliding with the environment, the roadmap is updated by removing the incriminated nodes and a new request is performed. This approach allows for good performances and addresses dynamic situations, such as moving obstacles.

When there are many obstacles in the environment, uniform sampling of the workspace is not appropriate: the probability of generating collision free configurations in narrow passages is low. Several approaches propose a way to bias the sampling to obtain a higher proportion of interesting configurations [HLM97].

In this case, sampling near obstacles configurations is an efficient way to improve the coverage of the configuration space [HST94, AW96, YTEA]. In [AW96] Amato et al. introduce the Obstacle Based PRM (OBPRM). As shown in Figure 2.6, the idea of OBPRM is to explicitly generate colliding configurations, which are then rotated randomly, and finally translated in a random direction until a collision free contact

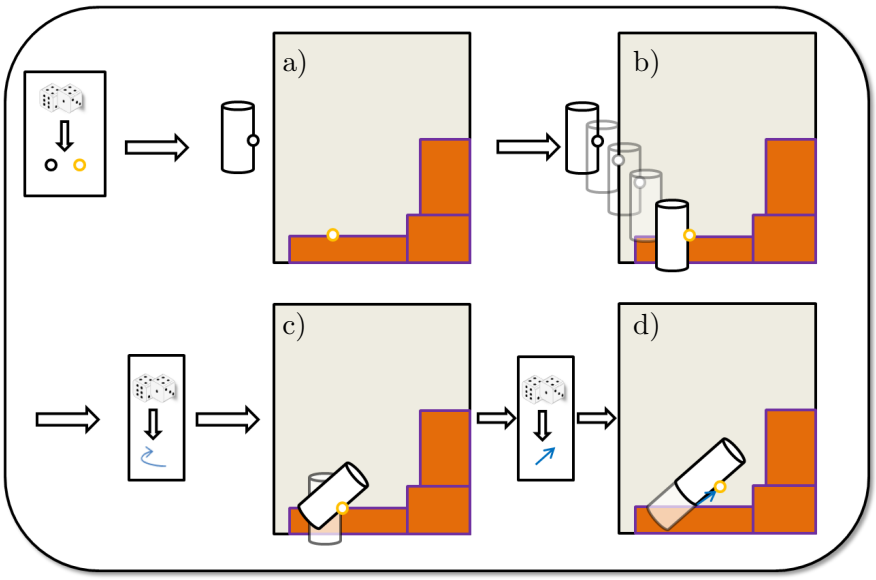

Figure 2.6: The configuration generation process of the OBPRM algorithm. A point on the robot surface is selected, as well as another point in one of the obstacle's surface (a). The robot is translated so that those two points coincide (b). The robot is then randomly rotated (c), before being translated in a random direction until the robot is in contact and collision free(d).

configuration is obtained. The authors also provide an analysis of the different methods available to bias the sampling while trying to maintain the genericity of PRM based approaches, in [ABL+98].

Although they are designed for constrained environments, the methods presented in this section do not address directly the full body motion synthesis of virtual characters, rather focusing on the complex manipulation problem. Constraints such as balance are not considered in these purely geometric approaches, but have been studied in locomotion planning contributions.

2.1.3 Motion planning for humanoid robots

Synthesising locomotion requires considering other aspects than finding collision free configurations. During human walk, for instance, contacts have to be generated alternately between each foot and the environment, while balance must be maintained throughout the motion. Previously presented probabilistic methods, i.e. geometric approaches, do not account for these specificities and therefore require to be extended.

Rather than sampling configurations in C_{free}, the objective of locomotion planners is to generate configurations in $C_{Contact} \subset C_{free}$, the subset of all the configurations where a creature is in contact with the environment (Figure 2.7).

To do so, two kinds of approaches have been proposed in robotics, depending on the complexity of the environment. If the environment can be navigated with motions such

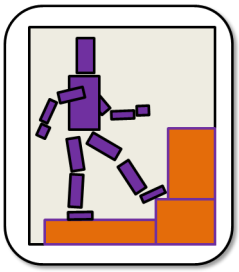

Figure 2.7: An example of contact configuration for both feet of a humanoid creature.

as walking or running, simplifications can be made in the sampling process, and the final motion can be achieved by replaying recorded motions such as motion capture data. In constrained environments, these simplifications cannot be made and new algorithms are required to generate contacts.

2.1.3.1 Strong heuristics for cyclic motions in simple environments

Human walk is a stereotypical cyclic motion. Thanks to its predictability in open environments, deterministic algorithms and strong simplifications of the environment are an efficient way to obtain natural looking solutions.

For instance in [PLS03, KJ98], the authors propose a 2 stage planner: first the environment is reduced to 2 dimensions (height is ignored), and a PRM planner is used to find a collision free trajectory. In Pettré et al., as shown in Figure 2.8, the robot legs are abstracted to a bounding cylinder to accelerate the collision checking step. In Kuffner et al. the whole body is abstracted. Secondly, once a 2D trajectory has been computed, motion capture clips are used to generate the actual walking motions.

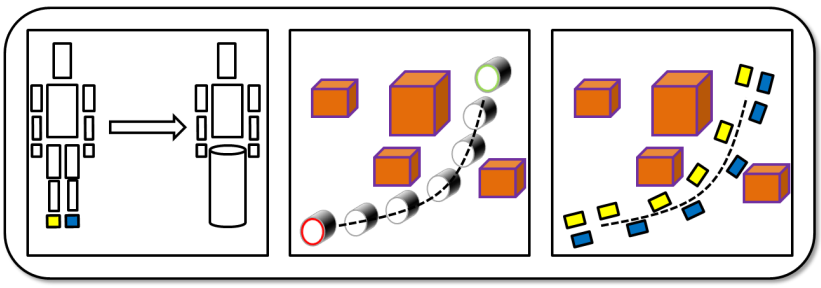

Figure 2.8: Pettré et al. use an abstraction of the character to plan a path in a 2D environment, before using motion capture to recreate the footsteps [PLS03].

In [CLS03], Choi et al. also use a PRM, this time to generate footprints in a 3D environment. Connections between the nodes of the PRM are defined using similarity to motion capture clips (Figure 2.9).

Other approaches were proposed, using finite state machine or similar graph ap-

Figure 2.9: Choi et al. combine a PRM planner with a motion capture database to plan a sequence of footsteps in a 3 dimensional environment [CLS03].

proaches. In [KNK+03, YLvdP07], the authors use forward dynamics to plan a sequence of steps in a discretized environment using a footstep transition graph or a finite state machine.

While these methods achieve natural-looking results, they only apply to deterministic cyclic motions (such as walking or running), in relatively open environments. Therefore they do not qualify for motion synthesis in constrained environments. A typical such application is the climbing scenario illustrated Figure 2.10, with two climbing walls comprising different sets of grasps. Considering the right-side wall, it does not seem reasonable to start climbing by creating a contact with one foot, whereas considering the left-side wall, it is probably how the climber would proceed.

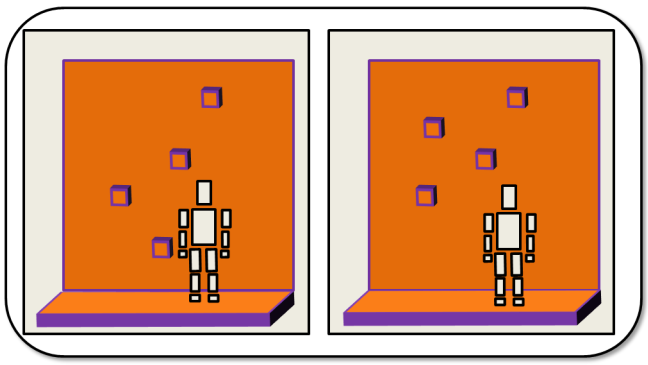

Figure 2.10: In two different constrained environments, replaying a deterministic sequence of contacts is sometimes impossible. A sequence of valid contact configurations has to be planned for each new environment.

2.1.3.2 Contact discretization for constrained environments

While the methods proposed in [KNK+03, LCH03] are restricted to walking motions, in [KvdP01] the authors combine motion planning and finite state machine approaches to plan motions in a discretized constrained environment (Figure 2.11). However, the space of valid motions remains limited by the finite state machine, restricted to walking, crawling, swinging and climbing motions. Hence, unexpected variations of the

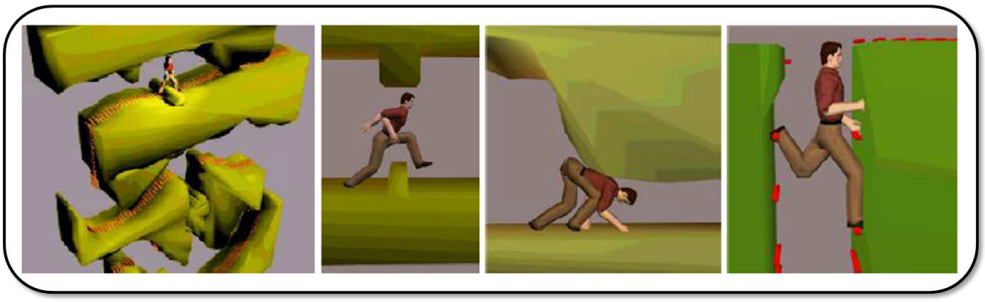

Figure 2.11: Kalisiak et al. combine motion planning and finite state machines to synthesize motion in 2.5D constrained environments. Potential contact positions are manually discretized [KvdP01].

environment cannot be handled by this approach.

To address our objective examples, a solution must be able to generate non stereotypical sequences of contacts, referred to as acyclic motions. *Contact before motion* methods were introduced in [HBL05, BRL$^+$04]. The idea is to generate incrementally an adjacency graph from the current configuration to the goal configuration. Two configurations are adjacent if one and exactly one contact differs from the initial configuration and its successor (Figure 2.12). A local PRM method is used to connect two adjacent configurations once the adjacency graph has been generated. This approach is efficient because the probability of finding a path between two adjacent configurations is high. A major drawback of this method however, is to require that the potential contacts in the environment be user-inputted. In [EKMG08] Escande et al. remove the need to specify the contacts manually by introducing a potential-field based algorithm to generate contact postures. However, to avoid reaching local minima or generating complicated paths, their method relies on a manually specified initial rough trajectory. In [BELK09] a method is proposed to automatically compute this input trajectory. This is achieved using a probabilistic planner to generate configurations in C_{free}, later projected in $C_{Contact}$ using inverse kinematics with priorities (Figure 2.13).

Nevertheless, as the name states, *contact before motion* approaches generate contact configurations independently from the trajectory, which can result in contacts being generated that are unadapted to the task being performed. To overcome this limitation, in [MTP12] Mordatch et al. introduce the Contact Invariant Optimization (CIO) method, a motion planner which optimizes simultaneously a trajectory and the contacts created along it. The trajectory is discretized into time windows of 0.5 seconds, during which contacts are set and cannot be changed. Along the process,

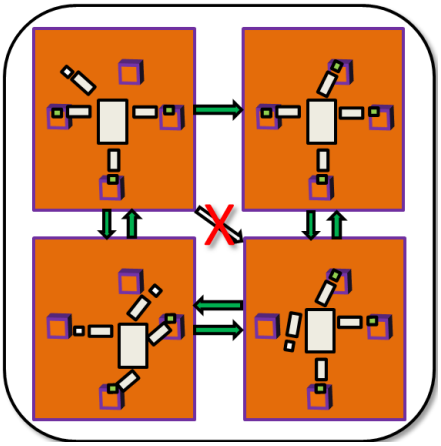

Figure 2.12: In *contact before motion* approaches, a transition between two states exists only if one and only one contact differs between the two states (effectors in contact are represented in green).

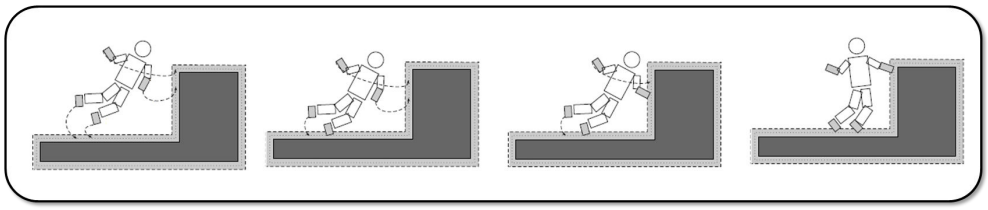

Figure 2.13: Bouyarmane et al. propose a method to automatically compute input trajectories for *contact before motion* planners. Contact configurations are obtained by applying inverse kinematics to near obstacle collision free configurations [BELK09].

an end-effector is guided towards the nearest surface satisfying dynamic constraints. However, similarly to potential field approaches, a relevant input trajectory must be produced for the CIO to work efficiently. In the constrained environments we address, the optimization method is hard to parameterize and can easily get stuck in local minima or fail to converge. Contrary to probabilistically complete planners, the CIO might fail to find a solution which exists. Furthermore, produced trajectories do not satisfy dynamic balance. Al Borno et al. also propose a full-body trajectory optimization method that does not require explicit contact definition, but still requires to specify which obstacles an effector should be in contact with [AdLH12].

Another issue of *contact before motion* planners and optimization methods is that the performance is usually low: computing a motion usually requires from several minutes to hours.

2.1.4 Conclusion

Robotics approaches for motion planning are primarily geometric methods. This makes the issue of synthesising natural looking motions a hard one. Most of the work presented here focuses in the first place on the hard problem of synthesising a feasible trajectory of stable contact configurations. Because efficient planners rely on random generation to find valid trajectories, the solutions they produce have to be refined to produce natural results. The first class of improvements brought to probabilistic planners consists in biasing the sampling process to improve the probability of generating interesting configurations. In constrained environments, stable motions are successfully generated thanks to *contact before motion* approaches. However, they still fail to generate natural looking motions. Optimization approaches such as the CIO show promise for improving the quality of these procedural methods, but need additional work to be applied to complex cases, as they are subject to local minima, performance and convergence issues.

2.2 Example based methods for synthesizing natural motions in constrained environments

In the previous section, we present motion planning methods, which can solve motion planning problems in arbitrary environments, thanks to their completeness property. Conversely, in this section we review contributions which primarily focus on generating natural motions in constrained environments. The contributions presented belong to the "example based" family. These approaches explore the configuration space in a restricted manner. They consider a database of preexisting motions, considered to be natural. Naturalness can be subjective (the motions have been designed by a professional animator) or objective (the motions have been recorded on real human actors using motion capture). The search of a trajectory in the configuration space is performed in the neighbourhood of these motions, which serve as a basic vocabulary: motions are blended, deformed, and finally concatenated to achieve a solution motion. The main advantage of this approach is that the solutions proposed are natural in the sense that they are similar to the motion database. However, we show that example based approaches do not provide the completeness of motion planning approaches, even when hybrid solutions are proposed, which makes them unadapted to constrained environments.

2.2.1 Motion editing techniques

The main issue of example based methods in constrained environments is that they are designed to be played in a predefined setup. If the position of the character varies a little, or if the environment is slightly modified, the recorded animation is no more relevant. As an illustration, taking a recorded motion of a character climbing up a staircase, altering the number of steps and replaying the animation will result in an unfit motion. Therefore, the reference motion must be adapted to fit the variations.

Three kinds of methods exist to adapt an existing motion to the constraints of a task or the environment: blending, warping and concatenation. Motion blending consists in creating new animations by combining two or more existing animations and assigning weights to them [MMKA04]. A survey of the different blending techniques can be found in [FHKS12]. Motion warping was proposed by Witkin et al. in [WP95]. It consists in deforming a motion by manually assigning spacetime constraints to segments of a virtual creature. The original trajectory is approximated into a curve that matches the constraints while following the original motion as much as possible (Figure 2.14). Motion concatenation consists in playing sequentially different motion clips to form a complex motion. Blending can be used between two motions to improve transitions between states.

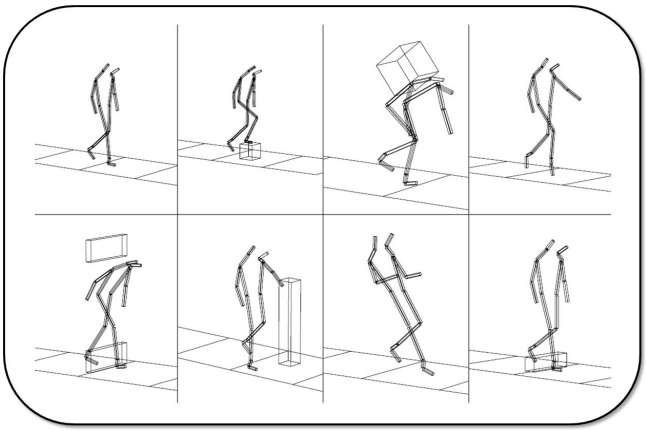

Figure 2.14: By specifying the right space time constraints, Watkin et al. manage to generate various motions from the initial walking motion (upleft) [WP95].

By combining blending, warping and concatenation, it is possible to generate new motion sequences adapted to environmental variations.

A more recent approach is proposed by Al-Asqhar et al. in [AAKC13], which adapts automatically the motions, and allows more important deformations of the environment and characters. The authors propose to formulate a motion in a constrained environment using a spatial relationship-based representation. The reference motions are expressed in terms of their relative position to sampled points of the environment. A similar approach is used in [PMM$^+$07]. In [HKT10] it is used to adapt distance relationships between several characters in close interaction with each other. This allows preserving the aspect of the original motion while varying parameters such as the height and weight of the character or the size of a car in which it enters. It allows important variations of the environment and the virtual character, as long as the topology is not changed. Limitations include: transposing a motion from a human to another animal or adding obstacles.

In [KMA05], Kulpa et al. also consider a different representation of the motion, which describes constraints intrinsically linked to the motion such as feet contacts with the ground. This representation allows to use a single reference motion with different

characters of various morphologies, but not to adapt it to new environments.

A common tool used in computer animation is Inverse Kinematics [BB04, LP12]. It can be combined with example based motion to provide more natural results. In his master thesis [Joh09], Johansen develops the locomotion system to adapt walking motions to height variations of the terrain, by combining motion warping with inverse kinematics. A reference walking motion is decomposed for each leg of the creature, according to the phases where the effector lands on the ground, is lifted, or is in the air. Given a terrain deformation, warping is automatically applied to adapt the trajectory of the effector to the new constraint. The locomotion system can be manually parameterized to avoid too important deformations of the motion (Figure 2.15). In [vBPE10], van Basten et al. address the same issue by formulating it as a search in the step space, extracted automatically from a motion capture database. They combine time warping and inverse kinematics on the result of a closest neighbour search to animate sequences of steps. Unfortunately these two contributions remain restricted to walking motions.

Figure 2.15: The locomotion system is a tool available to the industry which adapts walking motions to uneven terrain by combining motion warping and inverse kinematics (left). It is possible to adjust several parameters to avoid too important deformations of the original motion and unnatural postures (right). © Unity 3D.

Motion editing techniques can be seen as a set of methods enabling to extend the vocabulary made available by a reference motion database. They propose variations of these motions, allowing them to adapt to new environments, provided they have the same overall topology.

However, they do not avoid a strong simplification of the search space, because they assume a predetermined way to address a motion, expressed as a sequence of contacts. This assumption does not hold in constrained environments. In [RBC98] Rose et al. propose a "verb" and "adverb" formulation which illustrates this. A "verb" describes a control behavior such as walking, or crawling, and an "adverb" describes the way the "verb" is achieved: they are the control parameters of the motion. When the "verb" describes a stereotypical motion such as walking or running, the possible "adverbs" are

various but often similar: walking is always achieved through a stereotypical sequence of foot contacts, therefore editing the motion is relatively easy. But considering a verb such as "climbing", there are many "adverbs" possible, and combining them becomes much more challenging. Given a reference motion, and a climbing wall where the grasps' locations are randomly generated (Figure 2.10), there is no guarantee that the recorded sequence of contacts is achievable on the new wall. However, it would be possible to record additional climbing motions for various setups. In this case, choosing between the possibilities requires using motion planning techniques.

2.2.2 Combining motion planning and example based methods in constrained environments

As hinted in Section 2.1.3, motion capture is often used in the final stage of global motion planners to synthesize natural looking walking motions, as in [CLS03, PLS03, KJ98]. In these situations, the motion capture data has to be organized into specific structures to request the most appropriate motion for a given situation [LL04].

Some contributions extend the application of such hybrid methods to motion in constrained environments. In [YKH04], given a collision free manipulation trajectory computed by a sampling based planner, Yamane et al. apply inverse kinematics to achieve the motion. The inverse kinematics process is biased using a motion capture database so that the resulting motion looks as similar as possible to a requested reference motion.(Figure 2.16). In [HMK11], Huang et al. address a similar problem, but use motion blending to synthesize the motion. A sampling based planner is used, and the sampling is performed in the space of the blendable example motions. These methods allow to produce human-like motions because their results remain similar to the reference motion, but they are limited by the considered motion database.

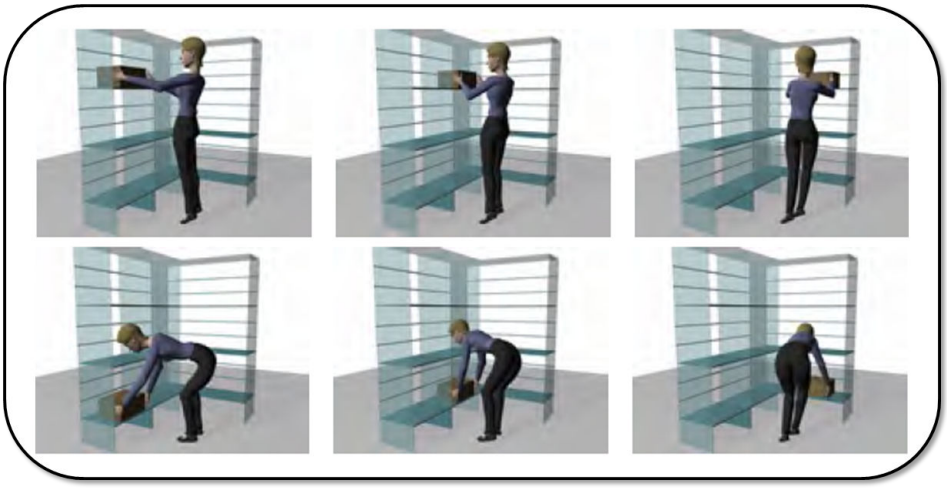

Figure 2.16: Yamane et al. address the manipulation problem by combining motion planning and inverse kinematics. The inverse kinematics step is biased using a motion capture database which leads to more natural motions [YKH04].

The reverse approach can be used efficiently in interactive applications. The motion capture data can be connected to finite state machines, trees or graphs, which will be requested to find a feasible trajectory. By requesting such motion graphs, in [KGP02] Kovar et al. concatenate several sequences of motion to achieve a new, global motion. In [SH07] Safonova et al. combine the A* pathfinding algorithm with motion graphs. This allows them to introduce heuristics to find a sequence of motions that will limit the deformations necessary to achieve the motion (Figure 2.17).

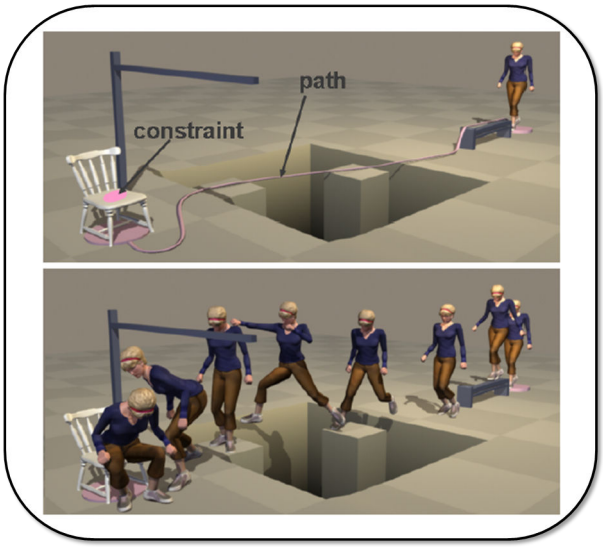

Figure 2.17: Safonova et al. combine motion graphs with pathfinding techniques to generate an optimized sequence of motions [SH07].

In [LK06], Lau et al. introduce precomputed search trees. From a finite state machine describing the possible transitions between recorded motions, they create a tree which describes all the possible transitions from an initial state until a fixed depth (Figure 2.18). The overall approach is similar to the dynamic roadmap method proposed in [KM04]. At runtime, given the current state and position of a virtual character in the environment, an ideal, collision free path is chosen from the tree and the sequence of motion is played. This approach allows adaptation to dynamic environments, and, contrary to the work of Safonova et al., provides real time performances. The approach is extended to unstructured motion graphs in [MK12], where the authors remove the need to manually design a state machine guiding the search.

Combining motion planning techniques with example based methods makes it possible to extend the possibilities of synthesizing natural motions in constrained environments, by decomposing the global motion into a sequence of motion capture clips. Still, constrained examples remain challenging for these methods (Figure 2.10), because the set of possible motions remains limited by the database.

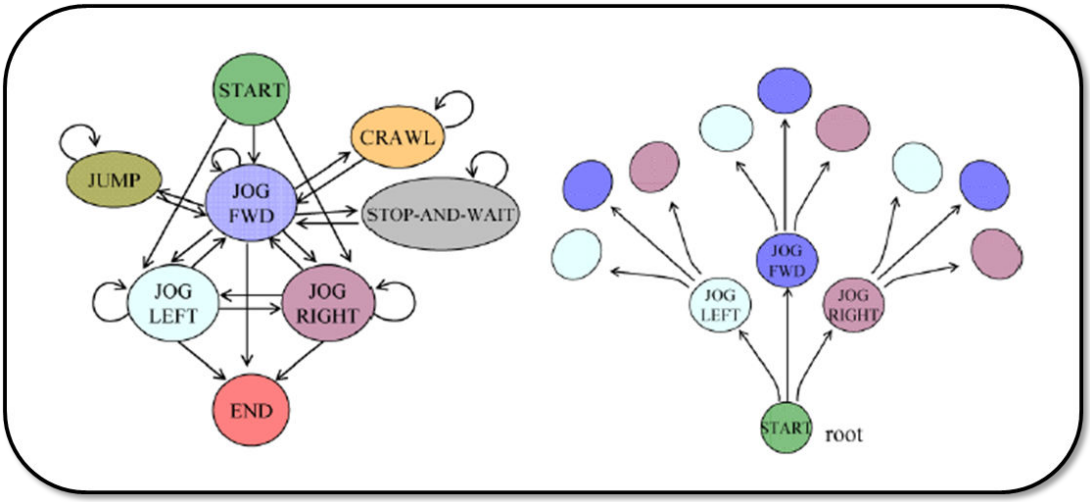

Figure 2.18: From a finite state machine describing the motion capabilities of a virtual character, Lau et al. create a tree that expands all the possible transitions from a given state up to a fixed depth. At runtime, the motions corresponding to the paths of the trees are evaluated regarding the current character position and orientation and the best path is selected and played [LK06].

2.2.3 Conclusion

More generally, example based methods make it possible to compute a natural looking motion which minimizes the deformations necessary to perform a motion task in a given environment. However, the space of solutions that can be found remains strongly restricted to the neighbourhood of the reference motion database. The advantage is that the generated solutions are always natural in the sense of this database, which make this approach popular and largely used in open environments. The major drawback is that in highly constrained environments, these methods do not provide guarantees to find a solution when it exists.

The main issue is that even when they are deformed, the motions played always assume a predetermined sequence of contacts to achieve the tasks, which is not a reasonable assumption in our case (Figure 2.10). Combining this approach with motion planning allows to extend the possibilities by searching a combination of motions from the motion database which can address the problem. But being able to address all the possible combinations requires very large motion libraries, which, as of today, are tedious and expensive to obtain. A last issue is that motion captures only apply to humanoids so far, and cannot be easily applied to imaginary creatures, so common in video games. Those *data driven* methods can be opposed to *model based* methods, which tend to be more flexible.

2.3 Model based approaches for natural motions

Model based approaches have been also considered for synthesising natural motions. For the majority of the models proposed, a motion is considered natural if it satisfies simplified laws of dynamics, such as balance and effort minimization. Consequently, we refer to them as dynamic models. We show that existing dynamic models are often limited to cyclic motions and are not sufficient to describe a natural motion in constrained environment.

We then review other criteria proposed for a natural motion, and consider their application to motion synthesis in constrained environments, especially regarding the issue of contact generation.

2.3.1 Dynamic models

Several computer graphics contributions use simplifications of such models along with optimization frameworks or controllers. They approximate force actuators using spring models, and model the motion according to the associated dynamics, and the minimization of various criteria.

Jerk minimization and dynamic balance are commonly used in trajectory optimization frameworks [LCH03, QEMR11, MTP12]. To do so, each criterion is formulated as a cost function that an optimization solver will try to minimize along the trajectory. However, in section 2.1.3.2 we emphasize that these optimization frameworks are not adapted to constrained environments.

Several publications focus on gaited locomotion, combining dynamic optimization with manually designed gait controllers to perform motion. Although the applications of such methods remain limited to cyclic motions, comparatively to motion capture approaches they apply to arbitrary creatures and are support to dynamic perturbations such as external pushes. A drawback of these approaches is that the proposed dynamic controllers often have to be carefully tuned, which is hard because they depend on a large number of parameters [CKJ+11]. The most recent contributions work around this issue by automatically optimizing the parameters of the controllers [WPP13, WPP14]. However, as of today no such solution has been successfully applied to constrained environments

Furthermore, dynamic balance and jerk minimization do not seem to guarantee a natural motion in constrained environments. For instance, there are many ways to stand up from a chair without losing balance, but not all of them are natural. Presented frameworks cannot discriminate between them, because they assume simple actuation models. The notion of effort and the analysis of force actuation capabilities in a given configuration have been considered by biomechanics contributions and recently adapted to motion synthesis frameworks.

2.3.2 Biomechanical models

The biomechanics community has proposed various models representing the way real world creatures move based on their capabilities. For instance, complex musculo-

skeletal models aim at simulating more precisely the internal interactions that occur when muscles are actuated during motion. Recent works such as [WHD+12, MWTK13] have successfully integrated biologically inspired actuation mechanisms into motion synthesis frameworks. As of today however, these attempts have succeeded only for a restricted set of motions, such as walking or running in open environments.

Other contributions identify motion invariants in human motions and propose models to implement these invariants in robotic simulations [SSL12], or character animation [01h01]. These contributions are essential to synthesize human-like motions, and can be used as criteria for evaluating their naturalness. However most of their applications remain limited to grasping motions. In [Yos85], Yoshikawa proposes one such criterion, more generic, called the manipulability index. It is a geometric measure quantifying the ability of robotic mechanisms in positioning and orienting end-effectors. The compatibility of the manipulability index for human beings has been demonstrated by the biomechanics community in [JBGR12]. Several manipulability based methods have been proposed to optimize contact configurations for manipulation or locomotion tasks. In [NL06], Naksuk et al. propose a cost function based on the manipulability index and use it for trajectory optimization. In [GKNK06], given a manipulation trajectory, an inverse kinematics solver uses manipulability as a secondary objective to optimize limb configuration along the trajectory. In [LH03], the sampling of a PRM motion planner is biased to generate configurations presenting a high manipulability index. Based on this work, Chiu proposes the force transmission ratio in [Chi87], another index for optimizing a manipulator pose relatively to a specific task, expressed as the need to exert a force in a given configuration. Manipulability based heuristics are appealing because they are generic means to evaluate configurations, independently from the morphology of a creature or the environment. In [KHB10] Kallmann et al. use motion invariants to plan the appropriate number of steps allowing the hand of a humanoid robot to reach a target in a plausible way.

However, the existing uses of motion invariants for planning in constrained environments are limited. The primary use today is motion optimization of manipulation tasks. One noticeable exception is the inspiring work of Bretl et al. in [BRL+04], where contact configurations are chosen based on a measure similar to the manipulability. Drawbacks of their approach include manual specification of the possible contact locations of the environment, and the restriction of the application to climbing tasks. Proposing automatic manipulability based planning could allow to generate more natural looking motions in constrained environments.

2.3.3 Conclusion

Because they do not make assumptions on the environment, model based methods seem adapted to motion synthesis in constrained environments. However, existing dynamic approaches focus on cyclic motions in open spaces and have not addressed the issue yet. Other models, inspired or validated by biomechanics, are more and more considered to mimic the behavior of real life creatures by evaluating "how natural" a given configuration might look. Several methods and heuristics for task efficiency have

been proposed to produce more human-like procedural motions, although as of today none lives up to data driven methods such as motion capture. Most of the available methods focus on respecting the laws of physics and minimizing energy consumption. As recent work demonstrates, including other models based on biomechanical definition is a promising mean to improve the results of procedural motion synthesis.

2.4 General conclusion on the related work

The key observation of this review is that naturalness and control are antagonist notions in motion synthesis [vWvBE+10]. A tradeoff must be found between ad hoc, natural looking methods and generic planning solutions for constrained environments. To perform automatic motion synthesis in constrained environments, an ideal planner should be able to:

1. Find solutions in unknown constrained environments, with numerous obstacles and narrow passages;
2. Generate natural looking motions, resulting from the creation of suitable contacts;
3. Compute acyclic, non deterministic sequences of contacts.

Table 2.1 summarizes the capabilities of state of the art methods regarding those three requirements. Additional relevant features such as dynamic balance and performance of the method have also been included.

Motion planning methods in robotics provide the completeness necessary to address the first requirement, thanks to a solid mathematical formalism: the configuration space. Conversely, example based approaches struggle to find solutions in constrained environments, their ability to explore the search space being limited by their reference motions.

Hybrid methods have been developed in an attempt to combine the benefits of both procedural and example based approaches, obtaining interesting results. However, our third requirement remains out of reach for example based approaches: because they replay a recorded motion, they necessarily assume a determined contact sequence to achieve a given motion. This is not applicable to an unknown environment.

Nevertheless, combining motion planning techniques with methods allowing to generate more natural motions remains an interesting lead to follow up on. Because example based motions do not apply in our context, it is important to provide a definition for a "natural motion", and propose new criteria based on it. We observe that while dynamic balance is one common criterion for naturalness, it does not appear to be sufficient. Other criteria for natural motions have been proposed, but few directly address motion planning in constrained environments. Therefore there is a need to explore such criteria for this specific context. Specifically, in constrained environment the surrounding obstacles provide many opportunities to synthesize new motions through the creation of contacts. There is a need to propose heuristics for **task efficient** generating contact configurations, characterized by their ability to exert the efforts required by the task.

	Unknown constrained environments (1)	Naturalness / Task efficiency of contacts (2)	Acyclic contact sequence (3)	Dynamic balance	Computational Speed
Probabilistic planners [KM04, AW96]	YES				+++
Locomotion planners [PLS03, CLS03]		YES		YES	+++
Contact before motion [BELK09]	YES		YES	YES	+
Contact before motion [BRL$^+$04]	limited	YES	YES	YES	++
CIO [MTP12]			YES	YES	++
Motion editing [AAKC13]	YES	YES		YES	real time
Example based planning [LK06]		YES		YES	real time
Dynamic controllers [WPP14]		YES		YES	real time
Musculo skeletal models [WHD$^+$12, MWTK13]		YES		YES	++

Table 2.1: Feature comparison of state of the art methods.

Chapter 3

Overview of our framework

Contents

In this chapter, we give an overview of the contributions proposed in this thesis. First, we recall our problem statement in terms of technical challenges. Then we present the approach we have chosen to address this issue.

3.1 Technical problem statement

We consider the issue of automatically synthesizing natural motions for virtual characters in constrained environments.

First, we limit the scope of this thesis to a variety of motions characterized by the following properties: additionally to being performed in constrained environments,

they require important force exertion, and are performed through an acyclic sequence of contacts. Examples of such tasks are climbing, standing up from a chair, pushing or pulling a cupboard...

The first obstacle raised by this formulation is the definition of a "natural motion", for which there is no consensus in the literature, especially in constrained environments. As explained in Chapter 2, we choose to define a natural motion as a sequence of configurations which are optimal for the task being achieved, according to a heuristic for "task efficiency". Since the tasks we address require an important force exertion, we choose to evaluate task efficiency as the capacity a character has to exert a force in a given direction. Specifically, we are interested in contact configurations, since force actuation results from them.

This leads us to the formulation of two questions:

- How to evaluate the "task efficiency" of a configuration?
- How to generate a task efficient configuration in a constrained environment?

Answering these two questions provides a partial solution to our initial research question, which we reformulate as: given two configurations in a constrained environment, how to compute a task efficient motion between them?

3.2 EFORT, a heuristic and a method for task efficient contact generation

Chapter 5 presents our answer to the two questions; we briefly describe this solution here.

3.2.1 How to evaluate the "task efficiency" of a configuration?

We answer this question with the Extended FORce Transmission ratio (EFORT), a heuristic for task efficiency. The EFORT heuristic works at the limb level. Given:

- A limb configuration, the end effector of which is in contact with a surface;
- A directional force exertion task, expressed as a vector $\mathbf{v_t} \in \mathbb{R}^3$;

EFORT returns a numerical value used as an indicator of the efficiency of the configuration regarding a force exertion task. EFORT performs a kinematic analysis of the joint articulations, and determines the joint angle variation necessary to exert a force in the desired direction. The more important the variation, the less efficient the configuration is, and the lower will be the value returned by EFORT.

3.2.2 How to generate a task efficient configuration in a constrained environment?

We address this issue by proposing a real time generator for task efficient contact configurations. The generator is implemented as a sampling based method, where the EFORT heuristic is integrated. Given:

- The current global configuration of the virtual character;
- The directional force exertion task $\mathbf{v_t} \in \mathbb{R}^3$;

The generator returns, from a set of automatically computed candidates, the contact configuration for which the EFORT heuristic returns the highest value.

3.3 A multi stage framework for task efficient planning in constrained environments

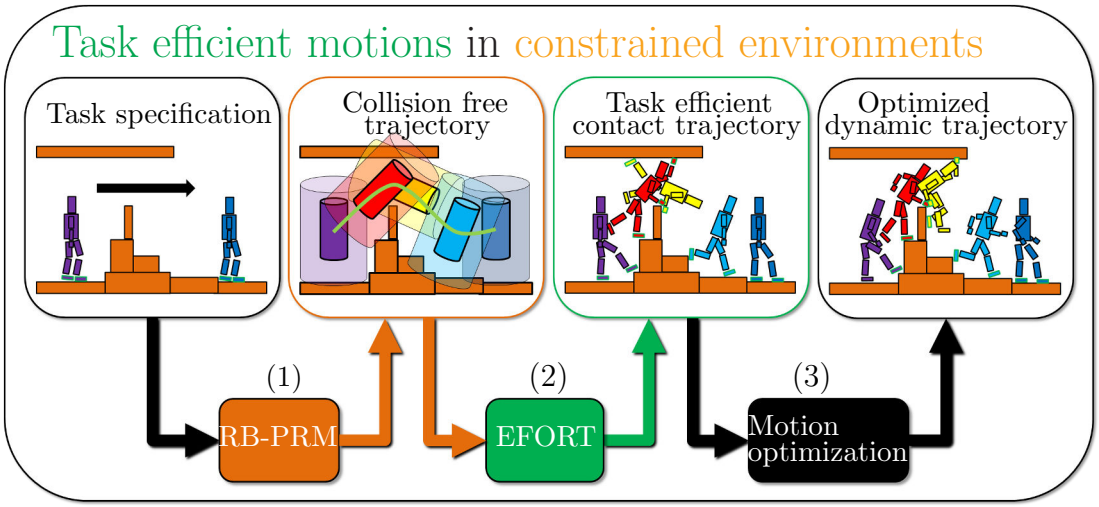

Figure 3.1: A multi step motion planner for the automatic computation of contact trajectories in constrained environments.

We can use the task efficient contact generator as a basis for a global motion planning method for constrained environments. Given a start and goal configuration for a character in a constrained environment, our motion planner generates a task efficient feasible contact motion connecting the two configurations.

In a first stage, using a probabilistic approach, a collision free path joining the start and goal is rapidly computed (Figure 3.1 - (1)). Thanks to this stage a solution path can be found in arbitrary environments. This path is purely geometric : constraints such as gravity and contact creation are not directly addressed. To ensure that the configurations along the path will allow to generate relevant contact configurations, we introduce a new heuristic called the **reachability condition**:
We first compute the Range Of Motion (ROM) of each limb of the virtual character. We then transform their convex hull into a 3D object, attached to the root of the character (Figure 3.2). If such an object collides with the environment, there is a good chance that a contact might be created between the concerned limb and obstacle. Also, we consider a second object englobing the trunk of the character, that must fulfill a

collision free constraint (Figure 3.3). The reachability condition we just described is verified by all the configurations composing the path computed at stage 1.

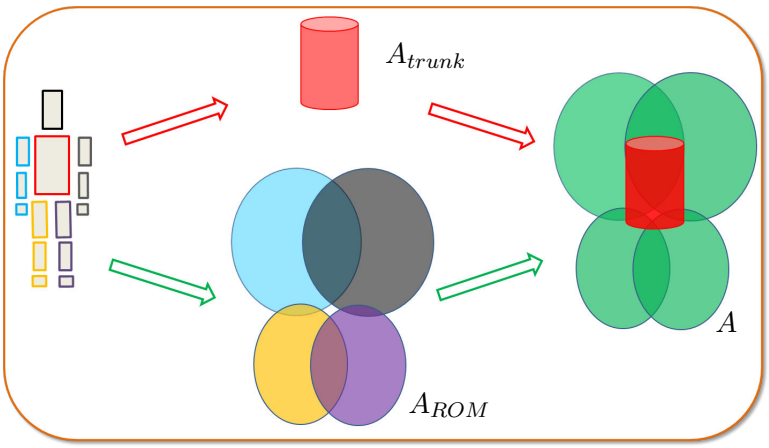

Figure 3.2: From a virtual humanoid R to its abstraction $A = A_{ROM} \cup A_{trunk}$. The red cylinder denotes A_{trunk} and must remain collision free. The green spheres are the objects composing A_{ROM}.

In a second stage, we consider the full kinematic and geometric definition of the character. As shown in Figure 3.1 - (2), the collision free path is transformed into a sequence of contact configurations, relevant for force exertion tasks. To do so, we transform a real time local contact generator proposed in Chapter 5 into a global method for computing task efficient contact sequences. Given a current configuration and the direction of motion, the contact generator computes a set of possible contact candidates and returns the best one according to our heuristic for task efficiency, the Extended FORce Transmission ratio (EFORT). By applying EFORT iteratively along the path, we obtain a trajectory expressed as a sequence of relevant contacts.

In a third and final stage, the contact trajectory is used as a guide for a high dimensional optimization method. As shown in Figure 3.1 - (3), the trajectory is refined to be dynamically balanced, minimize the jerk and respect joint velocities constraints.

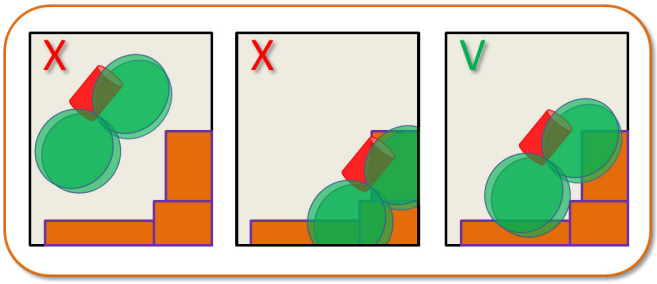

Figure 3.3: Illustration of the reachability condition. In the three examples shown, only the rightmost configuration satisfies the condition. It is the only one for which the red cylinder (A_{trunk}) is collision free while the green spheres (A_{ROM}) collide with the environment.

Chapter 4

Notation conventions and character representation

Contents

In this chapter we propose several definitions and conventions used throughout the thesis. First we define how we denote the mathematical objects used, then we give several definitions used in the algorithms described in this manuscript with a focus on the representation of the virtual character, its range of motion and its environment. The methods presented in this thesis rely on the generic definitions proposed here, making them adapted to arbitrary creatures and environments.

4.1 Notation conventions and mathematical tools

4.1.1 Notation conventions

A vector is represented by a bold lower case letter:

$$\mathbf{x} = [x_1, x_2, x_3]^T$$

A matrix is represented by a bold upper case letter:

$$\mathbf{A}_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

Variables and functions are represented lower case letters:

$$f(r) = r + 1$$

A set is represented with a upper case italic letter:

$$I_{even} = \{i \in I : \exists k \in I, i = 2k\}$$

4.1.2 Polytope and residual radius

In this thesis we use an elaborated model to represent the Range Of Motion of virtual characters, which can be based on experimental data (section 4.3.3). To accurately define this Range of Motion, we use a polytope formulation. Polytopes are also used in the formulation of a criterion for dynamic balance used in this thesis, and detailed in chapter 7.1. In this section we give basic definitions relative to polytope computation.

A polytope P is a sub space of the Euclidian space $\mathbb{R}^n$ defined by a set of hyperplanes:

$$P = \{\mathbf{s} \in \mathbb{R}^n : \mathbf{A}\mathbf{s} \leq \mathbf{b}\}$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$. In this work only convex polytopes are considered.

The residual ball radius $r(\mathbf{s}), \mathbf{s} \in P$ is the radius the largest hypersphere of center $\mathbf{s}$ that can be included in P. The computation method for $r(\mathbf{s})$ is given by in Algorithm 1.

Algorithm 1 $r(s) = \text{ResidualRadius}(\mathbf{A}, \mathbf{b}, \mathbf{s})$

Inputs H-representation of a polytope $[\mathbf{A}, \mathbf{b}]$, point $\mathbf{s}$
Output The residual radius r_q
$m \leftarrow \mathbf{A}.rows, i \leftarrow 0$
$\mathbf{A_n} \leftarrow \mathbf{A}, \mathbf{b_n} \leftarrow \mathbf{b}$
while $i < m$ **do**
    $\mathbf{A_n}(i,:) \leftarrow \mathbf{A}(i,:)/norm(\mathbf{A}(i,:))$
    $\mathbf{b_n}(i) \leftarrow \mathbf{b}(i)/norm(\mathbf{A}(i,:))$
    $i \leftarrow i + 1$
end while
return $min(\mathbf{b_n} - \mathbf{A_n}\mathbf{s})$

We define $\mathbf{c} \in P$ as the Chebyshev center. It is the center of the largest hypersphere contained in the polytope P:

$$\min_{\mathbf{c}, r(\mathbf{c})} \{r(\mathbf{c}) : \|\mathbf{c} - \mathbf{s}\|^2, \forall \mathbf{s} \in P\}$$

Finding **c** is a classical linear optimization problem. In this work we compute **c** thanks to the Matlab mpt toolbox[1]. We use the same toolbox for all the computations presented in this section.

4.2 Environment definition

An obstacle $O \in W$ is a set of 3D triangles. A triangle belonging to O is denoted t_O. W is the workspace, the set of all obstacles in the environment. An example of workspace is shown in Figure 4.1.

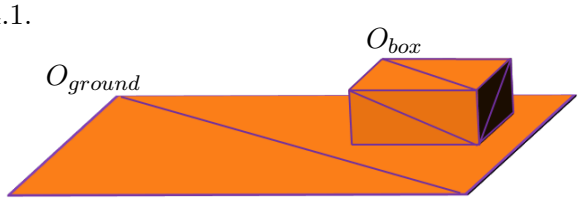

Figure 4.1: A example of workspace W, composed of two obstacles: the ground and a box, each one described by a set of triangles.

4.3 Virtual character definitions and representations

4.3.1 Skeleton definition

A virtual character is represented by an arbitrary kinematic chain R. R can be decomposed into a number l of subchains $R^k, 0 \leq k < l$, each one describing a limb of the creature. Every joint of a limb possesses at most one child. In Figure 4.2 the six limbs of the virtual creature appear in light blue. Each limb has one end effector. $l = 4$ for a human.

4.3.2 Character configuration

A character possesses a number $n \geq 6$ of degrees of freedom, attached to the several joints that compose the kinematic chain R. 6 degrees of freedom are used to denote the position and orientation of the geometric root of the character in the workspace W. The orientation of each joint is described using Euler angles.

The following definitions are illustrated in Figure 4.3:

- A configuration **q** is a n dimensional vector describing the current position, orientation and joint values of a virtual character.
- $\mathbf{q}^r \in \mathbb{R}^6$ is the vector describing the position and orientation of the root of the virtual character R.
- $\mathbf{q}^{x,y,z} \in \mathbb{R}^3$ is the vector describing the position of the root of the virtual character R.

[1] http://people.ee.ethz.ch/ mpt/3/

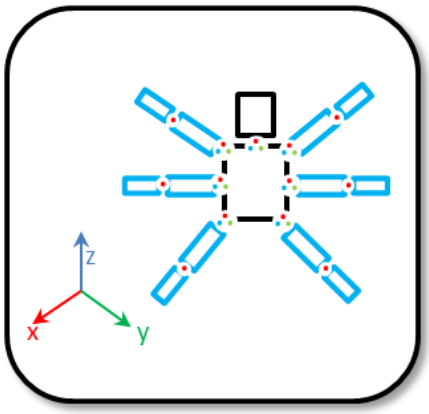

Figure 4.2: Reference posture of a virtual insect composed of 6 limbs and 33 degrees of freedom. Each limb is composed of 4 degrees of freedom. Each colored dot represents a degree of freedom around an axis, to which we add the position and orientation of the root of the creature in the world coordinates.

- $\mathbf{q}^k, 0 \leq k < l$ denotes a configuration of a limb R^k, the sub-vector of $\mathbf{q}$ that contains the joint angle values relative to the limb k.
- $\mathbf{q}^{\overline{k}}, 0 \leq k < l$ is the sub-vector of $\mathbf{q}$ that contains the joint angle values which are **not** relative to the limb R^k.
- For convenience, we define the operator $\oplus$:

$$\mathbf{q} = \mathbf{q}^k \oplus \mathbf{q}^{\overline{k}}$$

- $\mathbf{p}_{\mathbf{q}^k}$ denotes the world position of the end-effector of the limb R^k, given the configuration $\mathbf{q}$. Finally, we define
- $\mathbf{s^k}$ as the vector including the angle values corresponding to the root articulation of a given limb. In the case of a human for instance, s^k denotes the three angles that define the shoulder or ankle articulation.

Three additional vectors are defined for R:

- $\mathbf{a}_{max} = [a_0, ..., a_n]^T$ denotes the maximum acceleration allowed for each degree of freedom of R;
- $\mathbf{v}_{max} = [v_0, ..., v_n]^T$ denotes the maximum velocity allowed for each degree of freedom of R;
- $\mathbf{f}^k_{max}$ denotes the maximum force that can be applied by a limb R^k;

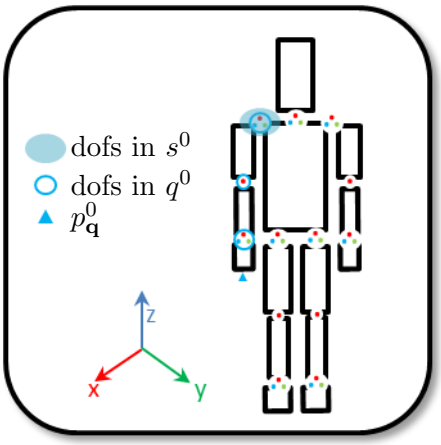

Figure 4.3: Virtual human in a rest configuration. The right arm is denoted as the limb R_0.

4.3.3 Character Range Of Motion (ROM)

Real world animals and robots have limited motion capabilities. For instance, the flexion of the human elbow can only be performed in one direction. Traditionally, to represent those limitations, the angle values that can be taken by a virtual character are bounded within a closed interval [Wel93]. Considering these constraints, it is easy to determine the Range of Motion, or ROM, of a given subchain of a virtual character. For instance, the possible values of the three degrees of freedom of the human shoulder are enclosed in a a parallelepiped rectangle (Figure 4.4 – bottom left).

However the actual ROM of the shoulder is more complex and restrictive, because of the dependencies existing between the degrees of freedom, as shown in [HUF04], [LLDM05] or [MKHB15]. Using fixed joint limits can result in unnatural configurations, or, if the limits are too restrictive, in the rejection of natural configurations. This is illustrated in Figure 4.4. For the case of a human, we address this issue by determining a more accurate ROM for the complex shoulder and hip articulations. To do so we use the protocol established by Haering et al. in [HRB14].

Using a motion capture system, we record hip and shoulder motions of maximal amplitude. At each frame of the recorded data, we determine the angular configurations of the studied articulations. Each configuration can be seen as a three-dimensional point, where one coordinate describes the value of one Euler angle (Figure 4.5).

We then compute the 3D non-convex hull K including all the recorded angular configurations. This is achieved using the method proposed in [Lun12]. K_{Hip} and $K_{Shoulder}$ represent the Range Of Motion (ROM) of the hip and shoulder. This means that any configuration included in K_{Hip} (respectively $K_{Shoulder}$) is a configuration of the hip (respectively the shoulder) that is valid for a human.

In this work, the protocol was applied on a single male subject, therefore the computed K_{Hip} and $K_{Shoulder}$ are specific to him. Haering et al. define a normalized ROM

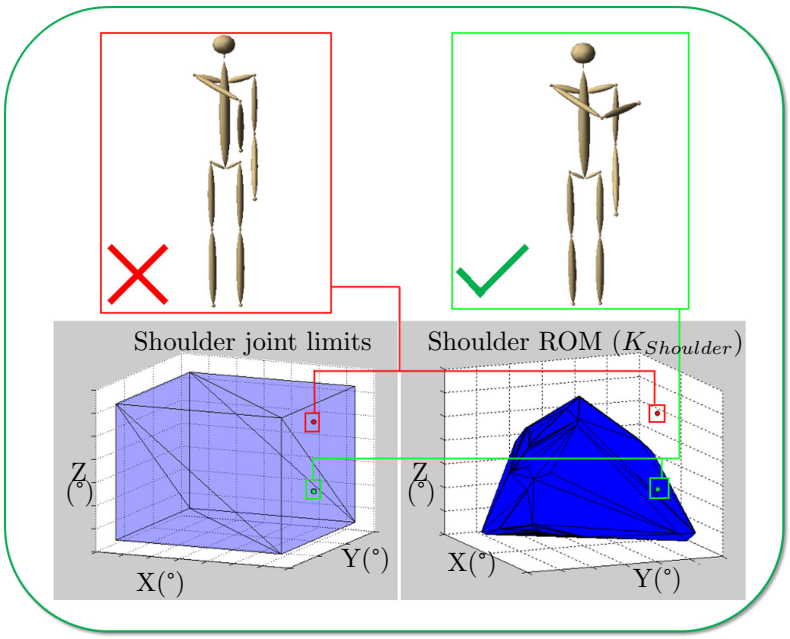

Figure 4.4: Generation of two sample configurations. Up: The two configurations lie within the joint limits of the shoulder, as shown in the bottom left plot. However the red configuration is rejected because it does not belong to $K_{Shoulder}$ (bottom right).

by including 3D poses common to a maximal number of participants into a hull of average volume. The user of the method has the choice between using an average ROM fitting any virtual human or using one specific to a given morphology.

In this thesis, in order to enforce the naturalness of a posture, rather than individually asserting that each joint value of a given configuration lies between its fixed limits, we consider simultaneously all the joints of a given articulation and verify that the point they describe is included in the non-convex hull K that represent the ROM of the articulation. To do so, we write K as a polytope and use the residual radius algorithm to assess that a configuration is valid. An additional benefit of this formulation is that is allows the computation of a normalized "distance" between a given configuration and a ROM violation.

However, the work proposed in this thesis concerns arbitrary creatures which are not human like, and for which the experimental data does not apply. When the data is not available, in the rest of this thesis, the Range Of Motion of the character is computed using the classical joint limits approach: the value of each euler angle associated to the articulation is bounded.

4.3.4 Abstraction of a virtual character

We consider a virtual character R, as described in section 4.3.2. In the first stage of our framework, our Reachability Based planner generates configurations based on the

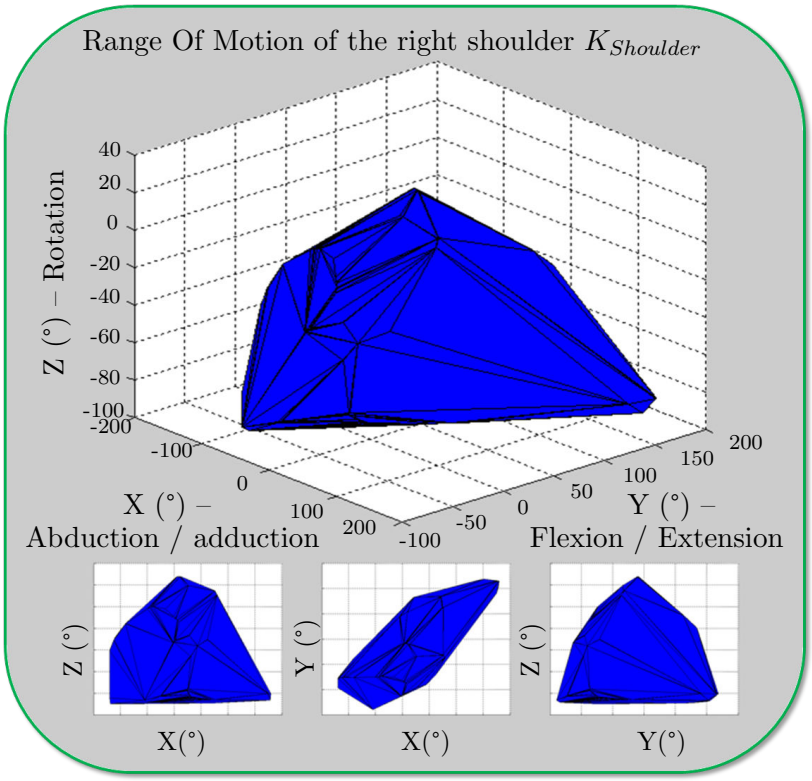

Figure 4.5: Representation of the Range Of Motion of the author's shoulder. The blue volume is the non convex hull $K_{Shoulder}$ including all the shoulder configurations that were recorded in a motion capture session, following the YXZ euler angle decomposition.

obstacles that lie within the Range Of Motion of the character. To do so, it uses a character abstraction defined here.

We define a character abstraction A as follows:

$$A = A_{trunk} \cup A_{ROM} \tag{4.1}$$

where A_{trunk} and A_{ROM} are two sets of 3D objects.

A_{trunk} is a 3D object englobing the trunk of the character (Figure 4.6 - Red cylinder). We compute A_{trunk} so that the following property stands: If A_{trunk} is collision free for a given position and orientation of the trunk $\mathbf{q}^r$, then there exists at least one set of joint angle values $\mathbf{q}^{\overline{r}}$, so that $\mathbf{q} = \mathbf{q}^r \oplus \mathbf{q}^{\overline{r}} \in C_{free}$. This means that if A_{trunk} is free of collision for a given position and orientation, then we can find a collision free configuration for R at these coordinates.

A_{ROM} represents the Range Of Motion of each limb of the virtual character. To compute A_{ROM}, we proceed as follows, for each limb $R^k, 0 \leq k \leq l$:

- We randomly sample a large number (> 10000) of valid configurations for the limb, for which we consider the positions of the end effector, expressed in R^k coordinates;

- We compute the minimum convex hull encompassing all these points, as described in section 4.3.3. We obtain a polytope ROM_k, which describes the Range Of Motion of R^k.

Then A_{ROM} is defined as the union of all the ROM_k:

$$A_{ROM} = \bigcup_{k=0}^{l} ROM_k \tag{4.2}$$

A_{ROM} is represented by the four green ellipsoids in Figure 4.6.

A is entirely described by a 6 dimensional vector $\mathbf{q}^r = [x_\mathbf{q}, y_\mathbf{q}, z_\mathbf{q}, \alpha_\mathbf{q}, \beta_\mathbf{q}, \gamma_\mathbf{q}]$, which describes the position and orientation of the geometric center of A_{trunk}.

We define $A^\mathbf{q} = A^\mathbf{q}_{trunk} \cup A^\mathbf{q}_{ROM}$ as the position and orientation of A when assigned a configuration $\mathbf{q}$.

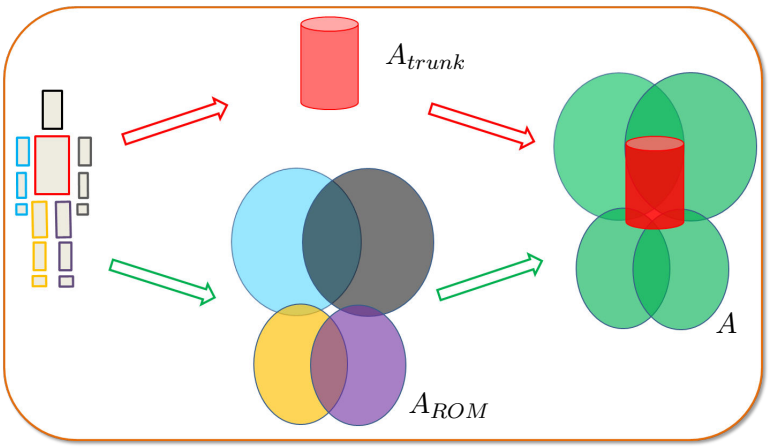

Figure 4.6: From a virtual humanoid R to its abstraction $A = A_{ROM} \cup A_{trunk}$. The red cylinder denotes A_{trunk} and must remain collision free. The green spheres are the objects composing A_{ROM}.

4.4 Additional useful definitions

4.4.1 The configuration space

C is the configuration space, the set of all possible values of $\mathbf{q}$. C^k is the configuration space of the limb R^k, the set of all possible values of $\mathbf{q}^k$. We define some relevant subspaces of C and C^k, illustrated in Figure 4.7:

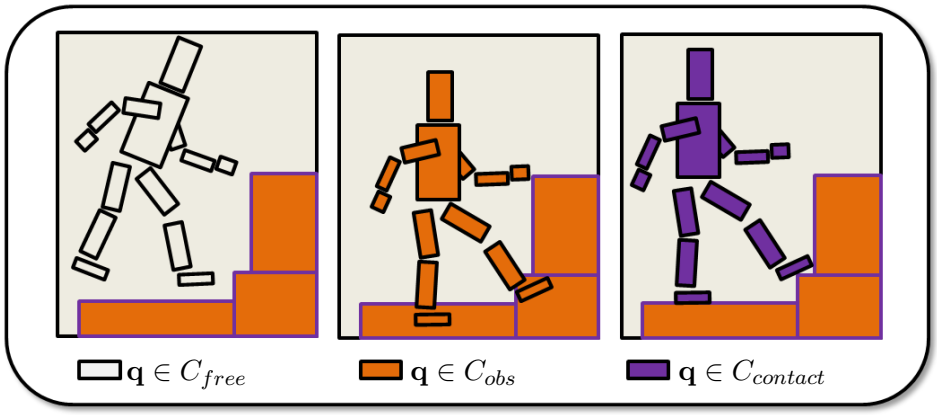

Figure 4.7: Examples of configurations in C_{free}, C_{obs} and $C_{contact}$

- C_{free} is the space of all configurations where there is no inter penetration between R and the environment W.

- $C_{obs} = \overline{C_{free}}$ is the space of all configurations presenting interpenetrations between R and the environment W.

- $C^k_{contact} \subset C_{free}$ is the space of all configurations where the end-effector of limb R^k is in contact with the environment. It is defined by

$$C^k_{contact} = \{\mathbf{q} \in C_{free} : \exists O \in W, d(\mathbf{p}_{\mathbf{q}^k}, O) < \epsilon\} \tag{4.3}$$

  where d is the function computing the minimal Euclidian distance between to objects, and $\epsilon \in \mathbb{R}$ a small user-defined number.

- $C_{NATURAL} \subset C_{free}$ is the space of all configurations that lie within the Range Of Motion (ROM) of a given character. $C^k_{NATURAL}$ is the space of all the configurations of the limb R^k that lie within the ROM of the limb.

4.4.2 Path, path interpolation, and trajectory

Path. A path $\mathbf{P}$ is a $n \times m$ matrix $[\mathbf{q}_{start}; \mathbf{q}_1; ...; \mathbf{q}_{m-2}, \mathbf{q}_{goal}]$ of configurations.
We define t_{max} as the total distance traveled along $\mathbf{P}$ from $\mathbf{q_{start}}$ to $\mathbf{q_{goal}}$;

Path interpolation. We define the path interpolation function $P(t), 0 \leq t \leq t_{max}$, defined over $m-1$ intervals:

$$\begin{array}{rccl} P: & [t_i, t_{i+1}] & \rightarrow & \mathbb{R}^n \\ & t & \mapsto & interpolate(P[\_, i], P[\_, i+1], \frac{t-t_i}{t_{i+1}-t_i}) \end{array}$$

where $t_i, 0 \leq i < m-1$ is the total distance traveled from $\mathbf{q}_{start}$ to $\mathbf{q}_i$ and *interpolate* performs the interpolation between two configurations.

Trajectory. We transform a path $\mathbf{P}$ into a continuous trajectory $T(t)$ by considering the maximum joint velocities constraints. We define the time between two consecutive configurations as the minimum time necessary to reach the second configuration without violating the velocity limits:

$$\Delta t_i = \max_{j=0,\ldots,n} \frac{(\mathbf{P}[j,i]^2 - \mathbf{P}[j,i-1]^2)^{\frac{1}{2}}}{\mathbf{v}_{max}[j]}$$

With $0 < i \leq m$, and $\Delta t_0 = 0$.

$T(t)$ is then obtained by computing the path interpolation function of P by replacing the indices t_i with the indices Δt_i.

Chapter 5

A heuristic for task efficient contact configurations: the Extended FORce Transmission ratio (EFORT)

Contents

In this chapter we consider our objective examples (Figure 5.1), and the following question: Given a force exertion motion task (pushing, pulling, standing up, climbing...) for a virtual character in a constrained environment, how should the next contact be created in order to efficiently perform the task? Addressing this local issue is essential to address the global motion planning problem, which requires the generation of plausible contacts configurations along a trajectory.

To address this issue, we first define what "efficiently perform the task" stands for, and provide a mean to evaluate this "task efficiency": the Extended FORce Transmission ratio (EFORT). Then we propose a real time contact generator to automatically

generate candidate configurations and select the best candidate configuration according to EFORT.

This chapter is organized as follows: In section 5.1 we give additional definitions, specific to this chapter. In section 5.2 we present EFORT, a kinematic heuristic to evaluate the task efficiency of a given limb configuration. In section 5.3 we present a real time contact posture generator based on EFORT. Finally we conclude this chapter with a discussion on the benefits and drawbacks of EFORT.

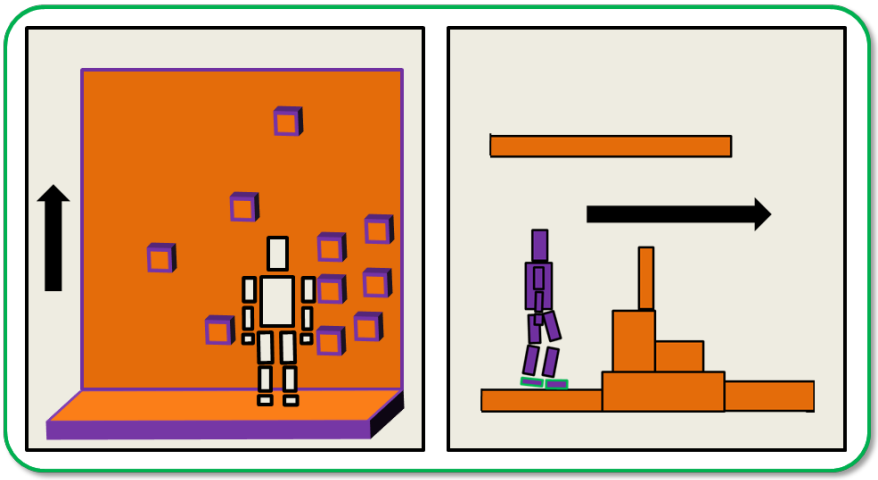

Figure 5.1: Where and how to create the next contact for a given limb?

5.1 Additional definitions

In this section we introduce the jacobian matrix and the velocity and force ellipsoids. We also present the inverse kinematics problem as well as some useful additional definitions.

As a kinematic heuristic, EFORT deals with a classical issue: studying how variations of the joint angle values of a kinematic chain affect the position and orientation of its end effector. It is therefore based on the jacobian matrix, a classical tool used to map joint velocities to effector velocities.

Using the jacobian matrix of a given configuration, we can determine how much variation of the joint angles is necessary to move the end effector along a given direction. This allows us to evaluate how efficient the configuration is to perform the desired motion: the more variation is necessary, the less efficient is the configuration. The velocity ellipsoid presented in this section (and similarly, the force ellipsoid) captures this efficiency in all directions of motion. EFORT extends the force ellipsoid as a heuristic for task efficiency.

In the following subsections we give the mathematical definitions associated with those concepts, before concluding with new definitions specific to our work.

5.1.1 The jacobian matrix

We consider a kinematic chain R, described by a configuration vector $\mathbf{q} = [q_0, ..., q_{n-1}]^T$. The position and orientation of the l end effectors $\mathbf{p}^k, 0 \leq k < l$, when expressed in R coordinates, only depend on $\mathbf{q}$. We can express it as follows:

$$\mathbf{p} = f(\mathbf{q}) \tag{5.1}$$

where $\mathbf{p} = [\mathbf{p}^0, ..., \mathbf{p}^{l-1}]^T$.

Differentiating with respect to time, we obtain

$$\dot{\mathbf{p}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{5.2}$$

$\mathbf{J}(\mathbf{q})$ is the $l \times n$ jacobian matrix. Its elements are given by:

$$\mathbf{J}_{i,j} = \frac{\partial \mathbf{p}^i}{\partial \mathbf{q}_j}$$

As a linear approximation of f at $\mathbf{q}$, $\mathbf{J}(\mathbf{q})$ describes how small variations from the configuration $\mathbf{q}$ affect the position vector $\mathbf{p}$. It can be seen as the transformation mapping a velocity in the configuration space into a velocity in the euclidian space.

Because the formulation is used several times throughout the chapter, we define the jacobian product $\mathbf{J}_p$:

$$\mathbf{J}_p(\mathbf{q}) = \mathbf{J}(\mathbf{q})\mathbf{J}(\mathbf{q})^T$$

For simplicity in the rest of the section $\mathbf{J}(\mathbf{q})$ is simply noted $\mathbf{J}$.

5.1.2 The velocity ellipsoid

We consider the unit ball in the configuration space C defined by the set of joint velocities for which the norm is inferior or equal to 1:

$$||\dot{\mathbf{q}}||^2 \leq 1 \tag{5.3}$$

From Equation 5.2 we can obtain the following equality:

$$\dot{\mathbf{p}}^T(\mathbf{J}\mathbf{J}^T)^{-1}\dot{\mathbf{p}} = \dot{\mathbf{q}}^T\dot{\mathbf{q}} \tag{5.4}$$

We can use Equation 5.4 to map the ball into an ellipsoid in the euclidian space $\mathbb{R}^m$:

$$\dot{\mathbf{p}}^T(\mathbf{J}\mathbf{J}^T)^{-1}\dot{\mathbf{p}} \leq 1 \tag{5.5}$$

This ellipsoid is called the manipulability ellipsoid, or velocity ellipsoid, introduced by Yoshikawa in [Yos85]. It describes the set of velocities that can be reached under the constraints of Equation 5.3 for the current configuration. The longer an axis of the ellipsoid is, the faster the configuration can move along the direction of the axis. Figure 5.2 - left shows the velocity ellipsoid for different configurations of a manipulator with two degrees of freedom.

Additionally, Yoshikawa defines the manipulability index w:

$$w(\mathbf{q}) = \sqrt{\mathbf{J}(\mathbf{q})\mathbf{J}(\mathbf{q})^T} \tag{5.6}$$

w measures the "distance" between a given configuration and a singular configuration. When w is equal to 0, the configuration is in a singular state; the greater w is, the further away the configuration is from singularity.

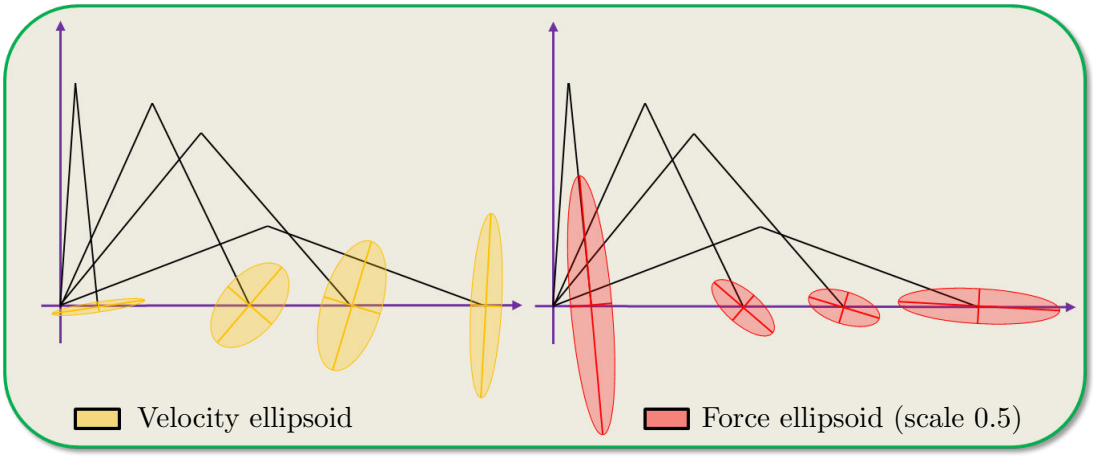

Figure 5.2: Examples of velocity and force ellipsoids for a manipulator composed of 2 dofs and 2 segments. Only the horizontal and vertical speeds are shown (not the rotation speeds), since it would require being able to draw in four dimensions.

5.1.3 The force ellipsoid

Similarly to the velocity ellipsoid, the force ellipsoid can also be defined. We consider a force vector $\mathbf{f}$ expressed in the task space $\mathbb{R}^m$; we also consider the equivalent joint torque vector τ. We can define the mechanical work in both spaces:

$$\dot{\mathbf{q}}^T\tau = \dot{\mathbf{p}}^T\mathbf{f} \tag{5.7}$$

From Equation 5.2 and 5.7 we obtain:

$$\dot{\mathbf{q}}^T\tau = \dot{\mathbf{q}}^T\mathbf{J}^T\mathbf{f} \Leftrightarrow \tau = \mathbf{J}^T\mathbf{f}$$

which leads to:

$$\tau^T\tau = \mathbf{f}^T(\mathbf{J}\mathbf{J}^T)\mathbf{f} \tag{5.8}$$

Therefore, the set of achievable force in $\mathbb{R}^m$ subject to the constraint:

$$||\tau||^2 \leq 1 \tag{5.9}$$

is the force ellipsoid defined by:

$$\mathbf{f}^T(\mathbf{J}\mathbf{J}^T)\mathbf{f} \leq 1 \tag{5.10}$$

The longer an axis of the force ellipsoid is, the more appropriate the configuration is to apply a force in the direction of the axis. Figure 5.2 - right shows the force ellipsoid for different configurations of a manipulator with two degrees of freedom.

5.1.4 Sample and sample container

We define a sample $s(\mathbf{q}^k)$ as the triplet $< \mathbf{p}_{\mathbf{q}^k}, \mathbf{q}^k, \mathbf{J}_p(\mathbf{q}^k) >$, where:

- $\mathbf{q}^k$ is a configuration of the limb R^k;
- $\mathbf{p}_{\mathbf{q}^k}$ is the position of the end effector of R^k in configuration $\mathbf{q}^k$;
- $\mathbf{J}_p(\mathbf{q}^k)$ is the jacobian product for the configuration $\mathbf{q}^k$.

We also define a sample container S^k as a set of samples. In our implementation, S^k is an octree data structure, where the end effector positions $\mathbf{p}$ serve as spatial indexes. This implementation allows to perform extremely efficient requests when it comes to find configurations in contact with the environment, as explained in section 5.3.2.

Finally, we define $Q^k \in C_{NATURAL}$ as the set of configurations in S^k:

$$\mathbf{q} \in Q^k \Leftrightarrow s(\mathbf{q}) \in S^k$$

5.2 EFORT: a new heuristic for task efficiency

In this section we describe EFORT, a mathematic mean to evaluate the "task efficiency" of a contact configuration. Given the environment and a set of candidate contact configurations, EFORT returns the most relevant configuration to achieve a given task. Because the tasks addressed in this thesis require important force actuation (Figure 5.1), we define the efficiency of a configuration as the ability a limb has to exert a force in a given direction.

We therefore consider the force ellipsoid defined in section 5.1.3 as a basis for our heuristic. We use the force transmission ratio proposed by Chiu in [Chi87] to measure the length of the force ellipsoid in the desired direction. The force transmission ratio is often used for optimizing a configuration against a single surface for a given task [GKNK06].

However in our specific problem, we do not know in advance the surfaces with which a contact must be created (Figure 5.1). Any reachable surface is a potential candidate, but some surfaces are more appropriated than others. To discriminate between them, we extend the force transmission ratio to include the quality of the contact surface.

5.2.1 The force transmission ratio

We consider a limb R_k of a virtual character R. R is given a force exertion task denoted by the translation vector $\mathbf{v_t}$. We also consider the current position of R, and a set of possible contact configurations $Q^k \in C^k$, in contact with several reachable surfaces. We want a heuristic to answer the following question: What is the most appropriate contact configuration $\mathbf{q}^k \in Q^k$ regarding $\mathbf{v_t}$? We make the hypothesis that for a subset of the possible motions, $\mathbf{v_t}$ will be satisfied more easily if the end-effector can exert a high force in the direction opposite to $\mathbf{v_t}$. This makes sense for the tasks we are

addressing, such as pushing a cupboard, or standing up from a chair, which might require an important effort.

Under this hypothesis, Chiu defines in [Chi87] the force transmission ratio f_{T}. It can be used to evaluate the length from the origin to the boundary of the force ellipsoid in the direction $\mathbf{v_t}$:

$$f_{\mathsf{T}}(\mathbf{q}^k, \mathbf{v_t}) = f_{\mathsf{T}}(\mathbf{q}^k, -\mathbf{v_t}) = [\mathbf{v_t}^T (J(\mathbf{q}^k) J(\mathbf{q}^k)^T) \mathbf{v_t}]^{-\frac{1}{2}} \tag{5.11}$$

Therefore, the higher f_{T} is, the more suited the configuration is for the force exertion task $\mathbf{v_t}$.

Given a force exertion task, it appears that the force transmission ratio is a good tool to evaluate a configuration. However, a drawback of f_{T} is that it does not account for the obstacle on which the contact is created.

5.2.2 EFORT: the Extended FORce Transmission ratio

We extend the force transmission ratio to include the quality of the contact surface. We consider that $\mathbf{q}^k$ is in contact with an obstacle O_i. We weigh f_{T} with the dot product between the task $\mathbf{v_t}$ and the normal $\mathbf{n_{Oi}}$ of O_i.

$$\alpha_{EFORT}(\mathbf{q}^k, \mathbf{v_t}) = f_{\mathsf{T}}(\mathbf{q}^k, \mathbf{v_t}) \mathbf{v_t} \cdot \mathbf{n_{Oi}} \tag{5.12}$$

If we maximize α_{EFORT}, obstacles with normals collinear to the motion task will be advantaged for contact creation. Therefore α_{EFORT} ensures that force exertion is actually applied against the obstacle (Figure 5.3).

This small modification largely extends the utility of the force transmission ratio: α_{EFORT} allows to consider the relevance of the surface the contact is created with. This results in discarding physically inefficient configurations, as illustrated by Figure 5.3: we consider the motion task $\mathbf{v_t}$ indicated by the black arrow. According to the force transmission ratio only, the top configuration is better to achieve $\mathbf{v_t}$. However, considering the contact surface normals, it appears that this configuration is inefficient. Indeed, applying a vertical force in this configuration would only result in moving the effector downward, because the surface reaction would not oppose the force. EFORT therefore selects the bottom configuration, despite its poor force transmission ratio.

Furthermore, it is interesting to observe that when the force transmission ratio is a symmetric and positive function, α_{EFORT} is asymmetrical and returns opposite values for opposite tasks. This makes it possible to distinguish between pulling and pushing configurations. Important negative values of α_{EFORT} imply that the surface normal goes in a direction opposite to $\mathbf{v_t}$. This is preferred for pulling tasks.

Therefore we can simply define a heuristic for choosing pulling configurations:

$$\alpha_{pull}(\mathbf{q}^k, \mathbf{v_t}) = -\alpha_{EFORT}(\mathbf{q}^k, \mathbf{v_t}) \tag{5.13}$$

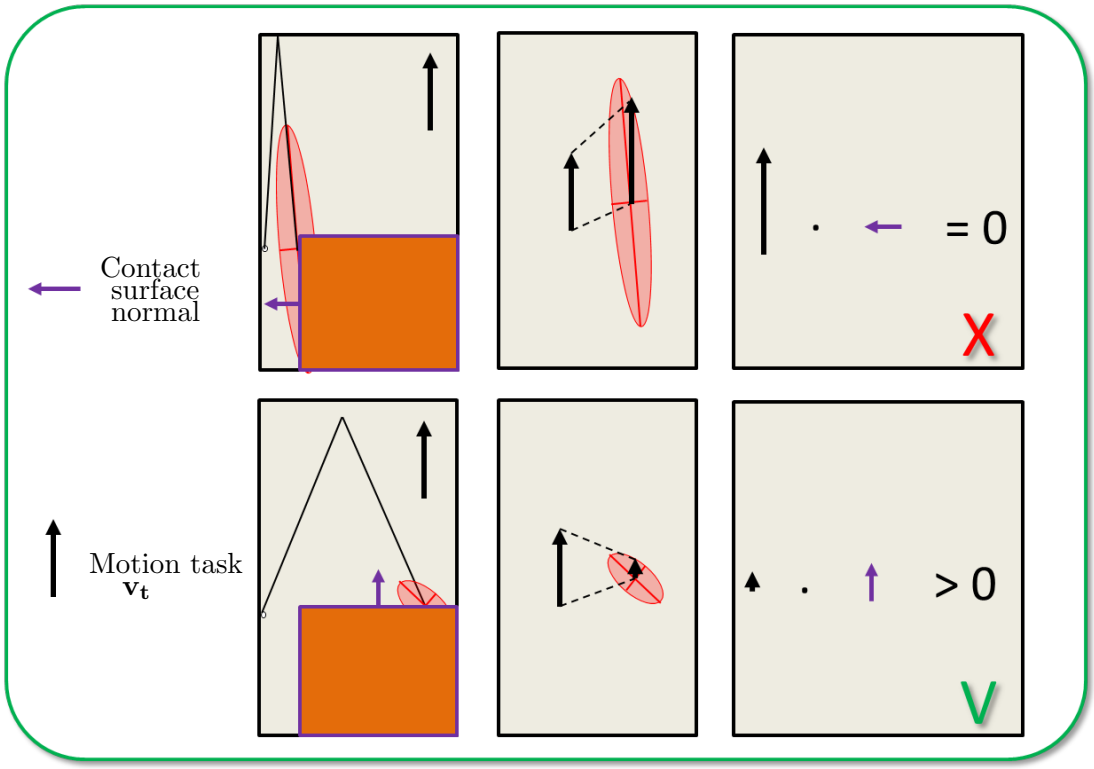

Figure 5.3: EFORT integrates the contact surface normals to the evaluation of contact configurations and favors surface normals aligned with the task.

5.3 Real time generation of contact configurations

We now describe a method to generate task efficient contact configurations. It computes a set of candidate contact configurations and uses EFORT to return the most effective for the considered task. For efficiency reasons, the algorithm is decomposed in two steps:

Offline sampling step. The first step is independent from the environment. A large set of arbitrary configurations Q^k is randomly generated for each limb R^k (Figure 5.4). Several precomputations are made for each configuration to accelerate runtime performance.

Online request step. The second step consists in a request performed on the configuration set Q^k. The configurations currently in contact with the environment W will be selected as potential solutions. Among those we select the configuration for which EFORT gives the highest score (Figure 5.5) in order to produce the final animation.

5.3.1 Offline generation of random limb configurations

This step is independent from the environment, and thus only has to be run once for each limb R^k composing our creature R. Figure 5.4 illustrates the sample configuration generation process. As inputs, we take a number of samples N and a limb R^k. We fill a sample container S^k with the sample configurations of R^k by repeating four steps N times:

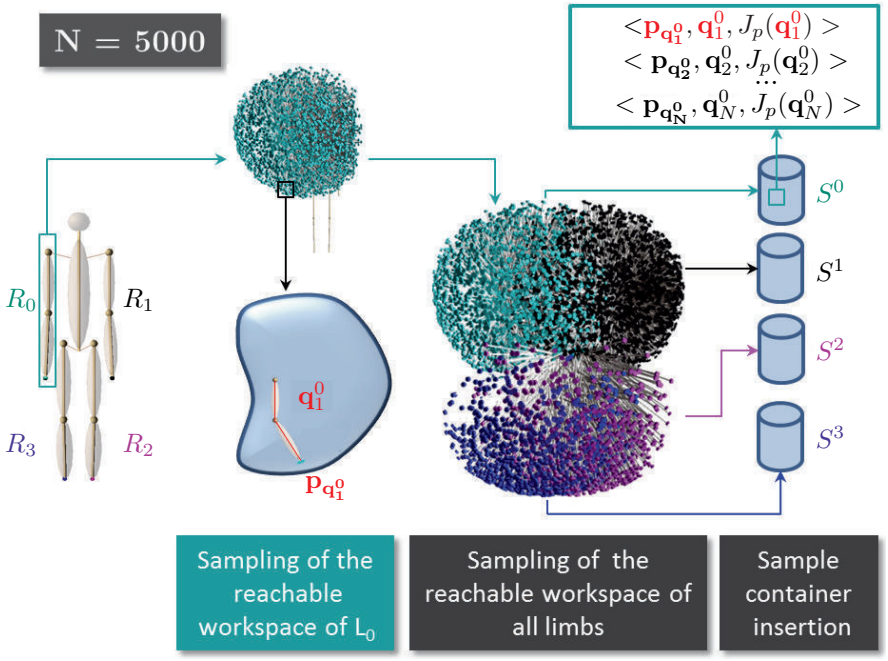

Figure 5.4: Illustration of the environment-independent offline sampling for $N = 5000$, for the right arm first, then for all the limbs. A sample container is created for each limb. An entry contains a configuration $\mathbf{q}^k$, and its jacobian product $\mathbf{J}_p$. Entries are indexed by the end-effector position $\mathbf{p_{qk}}$. For clarity the samples are shown in a wireframe form.

Random generation of a configuration $\mathbf{q}^k$ (Figure 5.4 – 1). This is done by generating a random angle value for each joint of R^k. The value of the angle is restricted by a range of motion (ROM) to avoid obtaining unnatural poses.

Computation of the jacobian product $\mathbf{J}_p(\mathbf{q}^k)$. The jacobian matrix $\mathbf{J}(\mathbf{q}^k)$ is computed and multiplied by its transpose $\mathbf{J}(\mathbf{q}^k)^T$ to obtain the jacobian product $\mathbf{J}_p(\mathbf{q}^k)$, needed for the runtime computation of the extended force transmission ratio α_{EFORT}. It can be computed directly from $\mathbf{q}^k$ but its computation is expensive. Storing it results in the additional storage of N matrices of size $3 * 3$ in memory. However it improves the online performance since it avoids computing it multiple times and reduces the computation of α_{EFORT} to two simple matrix products.

Computation of $\mathbf{p}_{\mathbf{q}^k}$ (Figure 5.4 – 1). The end-effector position $\mathbf{p}_{\mathbf{q}^k}$ is expressed in the limb coordinates R^k. Storing $\mathbf{p}_{\mathbf{q}^k}$ allows the implementation of the sample container S^k as a data structure efficient regarding proximity requests that will be performed at runtime.

Insertion of the resulting sample in S^k (Figure 5.4 – 2). We create the sample denoted by the triplet $< \mathbf{p}_{\mathbf{q}^k}, \mathbf{q}^k, \mathbf{J}_p(\mathbf{q}^k) >$, and store it into the sample container S^k.

As stated earlier, the generation of sample configurations has to be performed for every limb composing our creature R. For instance, for a virtual human, we would end up with four sample containers (one for each arm, and one for each leg). We cannot use a single tree for both arms because the joint limits differ symmetrically in our model. In this particular case, the symmetric container can be obtained by mirroring the value of the original one.

5.3.2 Online computation of task efficient contact configurations

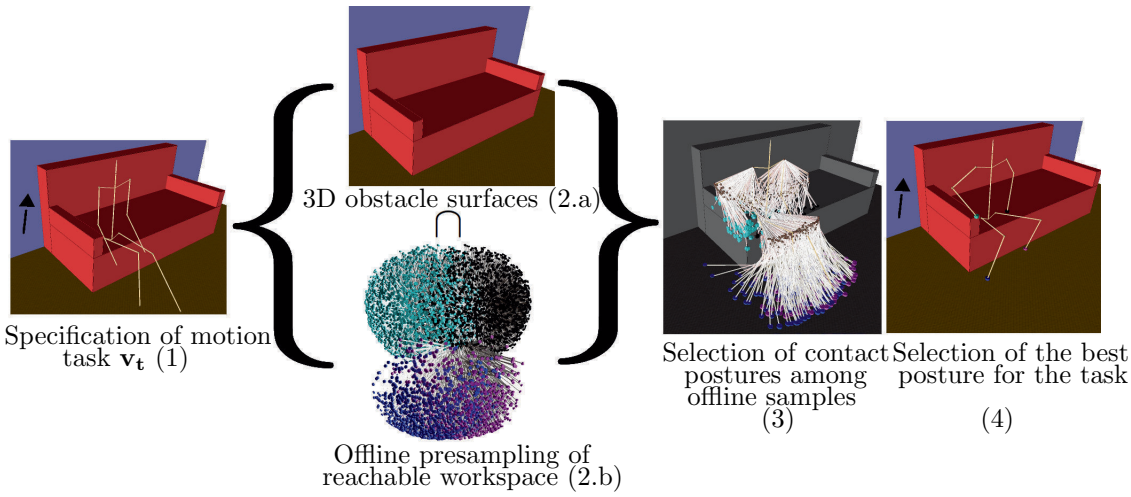

Figure 5.5: Online step request. Given the task of getting up (1), We transpose the samples from our database into the local environment (2), and select the configurations in contact with the environment (3). Among these candidates, we select the collision-free configurations that maximize the heuristic α_{EFORT} (4). For clarity the creature and samples are shown in a wireframe form.

We consider the motion task $\mathbf{v_t}$ (*Figure 5.5 – 1: the black arrow indicates the task of getting up*). To find a contact configuration for a limb R^k that is efficient for $\mathbf{v_t}$, we proceed in four steps:

Identification of the reachable obstacles. We retrieve the obstacle set $E \subset W$ of obstacles potentially reachable by the limb R^k (*Figure 5.5 – 2.a: the sofa, the ground*

and the wall). E is the result of a collision detection query between the environment and the non convex hull A^k_{ROM}, encompassing the range of motion of the limb centered at the root of R^k.

Selection of the samples in contact. We request S^k for all the samples that are in contact with an obstacle of E (*Figure 5.5 – 2.b)*. We recall that a configuration $\mathbf{q}$ is in contact if $\exists O \in E, d(\mathbf{p}_{\mathbf{q}^k}, O) < \epsilon$. The result of the query is a list of limb configurations $Q^k_{contact} \subset (Q^k \cap C^k_{contact})$ (*Figure 5.5 – 3: Selection of configurations in contact with the sofa and the ground*).

Ordering of the candidate samples. We sort the samples of $Q^k_{contact}$ using our heuristic $\alpha_{EFORT}(\mathbf{q}^k, \mathbf{v_t})$. This means that the first sample of $Q^k_{contact}$, that we call $\mathbf{q}^k_{max}$, verifies:

$$\forall \mathbf{q}^k \in Q^k_{contact}, \alpha_{EFORT}(\mathbf{q}^k, \mathbf{v_t}) <= \alpha_{EFORT}(\mathbf{q}^k_{max}, \mathbf{v_t}).$$

This is the configuration that is the most appropriate for the task regarding the extended force transmission ratio.

Selection of the best collision-free sample. As an output, the contact generator returns, if it exists, a collision free contact configuration maximizing α_{EFORT} (*Figure 5.5 – 4: our method places the right hand on the armchair, both feet on the ground, close to the root, and the left hand on the sofa*). Our algorithm does not provide a guarantee to find such configuration. The selection process is described in Algorithm 2.

Alignment of effector normal with surface normal using inverse kinematics In order to create a perfect contact, the end effector and obstacle surface normals are aligned using an inverse kinematics solver applied on the considered limb. The dot product of the normal vectors is used as a secondary objective function to be minimized by the solver, so as to align the effector and obstacle surfaces.

Algorithm 2 Selection of the best candidate configuration $\mathbf{q}^k_{max} \in Q^k_{contact}$

```
1: function SelectConfiguration(Q^k_contact)
2:     for all q^k in Q^k_contact do
3:         if CollisionFree(q^k) then
4:             return q^k
5:     return NULL
```

5.4 Discussion

In this chapter we presented a real time generator for task efficient contact configurations: given a force exertion task and the current configuration of the virtual character

in the environment, a contact configuration is automatically generated, based on its compatibility with the task. This compatibility is evaluated using the heuristic EFORT, also presented in this chapter.

In this section we discuss our method from three different perspectives: Advantages and limitations of the method, relevance of the EFORT heuristic, and finally applications and evolutions.

5.4.1 Advantages and limitations of EFORT.

To our knowledge, our contact posture generator is the first automatic real time solution for task efficient contact configurations in constrained environments. EFORT does not need prior knowledge of the contact surface, making it entirely automatic, and applicable for any environment, when previous manipulability based approaches required manual specification of the contact location.

Our method is simple to implement, and efficient regarding performance. It is generic because it works for arbitrary creatures and environments.

A limitation of our method is that our generator only creates contacts for the end effectors of a virtual creature. In constrained environments, people also use their knees, elbows, or any part of their body that might be useful. While the principles of EFORT could easily be applied to consider creating contacts on discrete locations (elbow, knee...), considering the creation of contacts for any parts of the creature is much more challenging with our approach. The issue has been considered by other contributions such as [KE08], but implementing them implies losing the real time property of our method.

A technical issue of EFORT is that configurations which maximize the force transmission ratio in a given direction tend to be close to singular configurations. This is not due to a numerical instability and reflects the proximity to joint limits. We overcome this issue by considering the manipulability index w, presented in section 5.1.2: the higher w is, the further away the configuration is from singularity. Configurations for which the manipulability index is not greater than a user defined threshold value are simply discarded during the sampling process. However, by doing this we also reject potential plausible solutions: a trade-off has to be found between an exhaustive approach and numerical stability.

5.4.2 Relevance of EFORT as a heuristic.

EFORT is a purely kinematic heuristic. A simple way to summarize it is to say that it evaluates, for a given configuration how much variation at the joint level is necessary to produce a variation at the task level. It has been shown that the manipulability measure is relevant for human beings [JBGR12], but that it is subject to some variations related to the muscular properties of the human limbs, as well as the joint limitations, not considered by the heuristic. A more accurate heuristic should integrate those physical limitations.

Because of this kinematic nature, EFORT does not consider the physics forces applied to the environment and the virtual creature. It must be used along with other heuristics which consider these criteria. The integration of dynamic balance within our contact posture generator is considered in our framework, and explained in Chapter 7.

EFORT maximizes the force transmission ratio in the direction opposite to that of the motion. This is relevant for the tasks we consider, but many other possibilities could be considered, using the force or the velocity ellipsoid. Small values of our heuristic are associated with a good control in the desired directions, which could be interesting for specific tasks such as writing. It would be interesting to consider these tasks and propose new heuristics for them.

5.4.3 Applications and future improvements

This thesis presents the integration of EFORT within a complete motion planning framework, to produce more natural motions in constrained environments. We believe that the contact posture generator proposed in this chapter can also be used as a "stand alone" method for real time animation, in a way similar to the locomotion system proposed in [Joh09]. Contrary to the locomotion system, EFORT is not limited to cyclic walking and running motions, and can be used to procedurally generate animations in constrained environments, as illustrated in Chapter 8. Another application that we are considering for the contact generator is its integration at the design level, within an animation software. By clicking on a button, an animator could be provided a set of suggestions for contact configurations for a creature in an environment. This could save a lot of time in the manual design of complex animations. A last considered application is the retargetting of motion capture data consisting in objects being manipulated. EFORT could be used to recompute new contact points relevant for new objects.

Among the possible improvements that could be brought to EFORT, it would be interesting to consider compatibility of the whole body posture rather than the limb configuration. Currently we do not integrate the fact that actuating several limbs at the same time could result in undesired torques on the body in the case of a dynamic simulation. Therefore we want to combine the method with a complementary global posture optimization technique [LMEE12]. Doing this would allow us to optimize the whole body according to the computed limb configurations, and produce more natural results.

Chapter 6

Stage 1: A Reachability Based Probabilistic Road Map (RB-PRM)

Contents

This chapter presents the first part of our second contribution: the Reachability Based Probabilistic Road Map (RB-PRM). This high level motion planner serves as an entry point to our motion planning framework (Figure 6.1 - 1).

Classical motion planners (PRMs or RRTs) generate contact configurations and connect them into a collision-free path in the workspace, independently from the followed trajectory. To be able to generate task efficient configurations however, the information brought by the trajectory is required, as detailed in Chapter 5. RB-PRM results

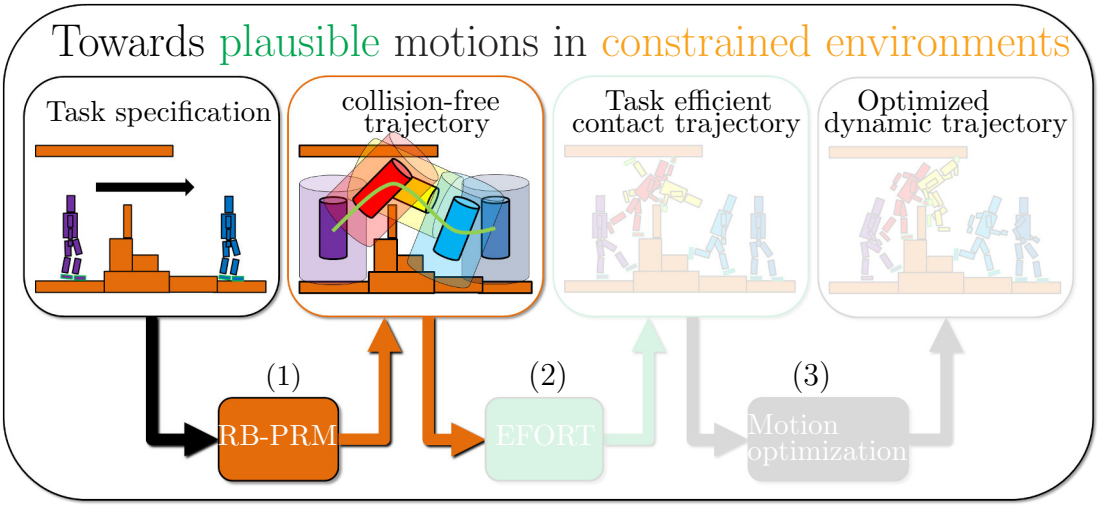

Figure 6.1: In this chapter we present RB-PRM(1), a global motion planner for collision-free trajectories.

from this observation and does not try to generate contact configurations. Rather, it is designed to return collision-free paths that stay close to obstacles. This will allow further contact creation in the second stage of our framework (Figure 6.1 - 2).

RB-PRM considers the following inputs: a virtual character, the environment, a motion task expressed as start and goal configurations. It outputs a continuous and smooth trajectory $q(t)$ going from the start to the goal configurations. $q(t)$ is constrained to remain close enough from obstacles while avoiding collisions.

RB-PRM belongs to the Probabilistic Road Map Family (PRM). As such, it operates in two steps:

- In the first phase we try to capture the topology of the environment into a graph (section 6.2); a set of collision-free configurations, randomly sampled, are connected to each other if a collision-free path exists between them. At this stage there is no information on the nature of the future queries (and thus, no information on the tasks), therefore the roadmap is not composed of contact configurations. Instead, RB-PRM selects configurations which are "close enough" from obstacles to allow ulterior contact creation, hence the term "Reachability based";

- In the query phase, a search is performed among the graph to find a path connecting a start configuration to a goal configuration (section 6.3). Due to the random nature of planners, the returned path often looks unnatural and requires refinement. We transform it into a smooth, refined and collision-free spline trajectory using a "shortcut" algorithm.

This chapter introduces the reachability condition, as well as an algorithm to efficiently generate configurations that verify it. Thanks to this formulation, we can abstract an arbitrary complex virtual creature into a model entirely described with

little more than 6 degrees of freedom. This is done without losing genericity. More importantly, in our context of constrained environments, this formulation provides a heuristic for selecting configurations that will allow ulterior contact creation. The trajectory outputted by RB-PRM is indeed used as an input for the second level of our framework (Figure 6.1 - 2). Using a real time posture generator, this second level transforms the said trajectory into a contact trajectory.

The remainder of this chapter is structured as follows: section 6.1 introduces additional definitions, specific to RB-PRM; section 6.2 describes the offline phase of RB-PRM; section 6.3 presents the online request phase of RB-PRM; section 6.4 concludes with a discussion on the benefits and drawbacks of using RB-PRM.

6.1 Additional definitions

6.1.1 Configurations of RB-PRM

In this chapter, we work with the character abstraction A of the virtual character R. This simplified model is entirely described by the 6 degrees of freedom indicating the position and orientation of the root of the character $\mathbf{q}^r$. The definitions relative to the character abstraction can be found in Chapter 4. In the present chapter every configuration mentioned refers to $\mathbf{q}^r$. For this reason and to simplify the equations, we use the notation $\mathbf{q}$ instead of $\mathbf{q}^r$ to describe a configuration.

6.1.2 The reachability condition

We recall that A is composed of two sets of objects, so that $A = A_{trunk} \cup A_{ROM}$. A_{ROM} is represents the Range Of Motion of each limb of the virtual character. A_{trunk} is a shape presenting the following property: For a position and orientation of the root $\mathbf{q}^r$, if A_{trunk} is free of collision, at least one collision-free exists for the virtual character R at this position. Using the character abstraction A allows us to define "interesting" configurations (Figure 6.2): If A_{trunk} is free of collision while the objects of A_{ROM} are in collision with the environment, there is a good chance that a contact configuration can be found for the virtual character. We call this the **reachability condition**. We can then define $C_{reachability}$ as the set of configurations verifying the reachability condition:

$$C_{reachability} = \{\mathbf{q} : A^{\mathbf{q}}_{ROM} \cap W \neq \emptyset \wedge A^{\mathbf{q}}_{trunk} \cap W = \emptyset\} \tag{6.1}$$

The reachability condition can be seen as an extension of the work of Pignon et al. in [PHL91] and [PHL92], restricted to motion planning for 2D cell decomposed environments, to the three dimensional unrestricted more general case.

6.2 Generating RB-PRM

In this section we present the algorithm used in the offline step of RB-PRM. It consists in generating a graph of configurations that captures, at least partially, the topology of

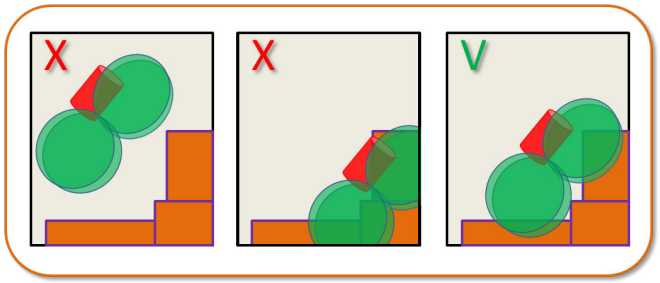

Figure 6.2: Illustration of the reachability condition. In the three examples shown, only the rightmost configuration is accepted. It is the only one for which the red cylinder (A_{trunk}) is collision free while the green spheres (A_{ROM}) collide with the environment.

the environment. The configurations (or nodes) are connected if a collision-free path exists between them. The inputs of this step are: a virtual character abstraction A; the environment W. During this step, the main difference between RB-PRM and classical probabilistic planners lies in the way that candidate configurations are generated. The configurations are required not only to be collision-free, but also to be close enough to the environment to allow contact creation in an ulterior step. First we describe how such configurations are generated. Then we describe the graph generation algorithm, which is based on the Visibility-PRM introduced by Siméon et al. in [NSL99]. This section is concluded with the description of the local methods used by Visibility-PRM to connect configurations in the graph.

6.2.1 Sampling the configuration space

The objective of RB-PRM is to generate a graph of configurations which belong to $C_{reachability}$. Uniform sampling is not relevant because the probability of randomly generating such configurations is low. To generate efficiently configurations in $C_{reachability}$, we use a variant of the OB-PRM generation method introduced by Yamato et al. in [AW96], presented in Chapter 2.1.2. The generation of one candidate configuration (Figure 6.3) is made thanks to Algorithm 3. The idea is to select a triangle in the workspace W, to translate the character abstraction A to its position, before translating and rotating randomly A by small increments until the reachability condition is met.

Two user defined numeric values, $LIMIT1$ and $LIMIT2$, ensure that the generation will not loop infinitely around areas of the workspace where the reachability condition cannot be met.

To select a triangle t_{obs} to sample, we do not proceed in a purely random manner. In [ABL+98] Yamato et al. describe two ways of selecting a triangle:

- Randomly select a triangle. This will bias the generation of configurations near areas of the workspace composed of a lot of triangles;
- Select a triangle with a probability proportional to its area. This will bias the generation of configurations near large areas (usually, open areas).

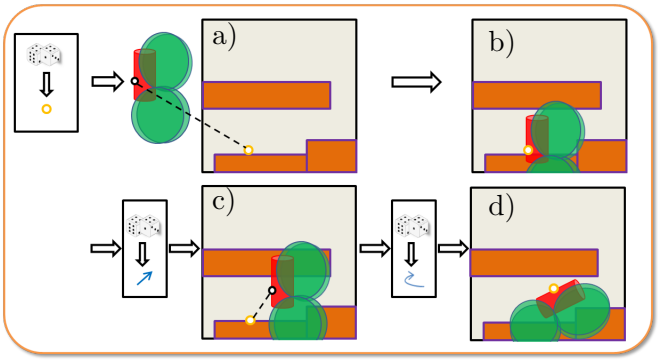

Figure 6.3: RB-PRM configuration generation process. A point in one of the obstacles surface is selected (a). The root of the character is translated to this point (b). It is then translated in a random direction (c), before being randomly rotated until the reachability condition is met (d).

Algorithm 3 Generation of a configuration $\mathbf{q} \in C_{reachability}$

function GENERATECONFIGURATION
  $\mathbf{q} \leftarrow [0, ..., 0]^T$ /*Set all DOFs to 0*/
  $ntry1 \leftarrow 0; ntry2 \leftarrow 0;$
  while $ntry1 < LIMIT1$ **do**
    Select a triangle $t_{obs} \in$ workspace W
    while $ntry2 < LIMIT2$ **do**
      Translate $\mathbf{q}$ to random point of t_{obs}
      $\mathbf{d} \leftarrow$ Random unit direction
      while $A^{\mathbf{q}}_{ROM} \cap t_{obs} \neq \emptyset$ **do**
        $\epsilon \leftarrow$ random number, $0 < \epsilon \leq 1$
        Translate $\mathbf{q}$ by $\mathbf{u} = \epsilon\mathbf{d}$
        if $\mathbf{q} \in C_{reachability}$ **then**
          return $\mathbf{q}$
        else
          Rotate $\mathbf{q}$ randomly
          $ntry2 \leftarrow ntry2 + 1$
    $ntry1 \leftarrow ntry1 + 1$
  return failure

Experience showed us that a combination of these two generation methods turns out to be more appropriate to efficiently cover the workspace. Therefore, each time we want to select a triangle, we choose a random selection method with a probability s, or the weighted generation method with a probability $1 - s$.

6.2.2 Graph construction

RB-PRM is a declination of the Probabilistic Roadmap Algorithm, more precisely of the Visibility PRM, introduced by Siméon et al. in [NSL99]. For the reader's convenience we provide and explain the algorithm in this section, although it is not our contribution.

The term "Visibility" in Visibility-PRM refers to the connectivity of a configuration, or node, in the graph. We consider a *local planner* l expressed as a predicate: $l(\mathbf{q_0}, \mathbf{q_1})$ returns true if a path exists between $\mathbf{q_0}$ and $\mathbf{q_1}$.

This allows us to define the visibility domain $Vis_l(\mathbf{q_g})$ of a configuration $\mathbf{q_g}$ as

$$Vis_l(\mathbf{q_g}) = \{\mathbf{q} \in C_{free}, l(\mathbf{q_g}, \mathbf{q})\} \tag{6.2}$$

Figure 6.4 illustrates this notion for a two-dimensional configuration space. Considering a virtual creature composed of a single non oriented circle, a configuration can be represented as a 2D point in the plane. This point refers to the center of the circle. Now we define a local planner l so that $l(\mathbf{q_0}, \mathbf{q_1})$ returns true if a straight line can be drawn between $\mathbf{q_0}$ and $\mathbf{q_1}$ without colliding with an obstacle. In the first 3 frames of Figure 6.4, a configuration is represented by a black dot and the colored area denotes its visibility domain. The graph generation process is described in Algorithm 4. When a new configuration $\mathbf{q}$ is generated, it is only added to the graph if one of the following conditions is met:

- $\mathbf{q}$ does not belong to the visibility domain of any configuration in the graph (this happens when the graph is empty for instance). This means that $\mathbf{q}$ is interesting because its location belongs to the unexplored area of the workspace W. The configuration is called a *guard* node (black dots in Figure 6.4);

- $\mathbf{q}$ belongs to the visibility domains of two configurations for which no path currently exists in the graph. This means that $\mathbf{q}$ allows to connect two previously unconnected components of the graph. The configuration is called a *connecting* node (white dots in Figure 6.4).

Any configuration that does not comply to either one of these conditions is considered unnecessary and will be rejected. An example of graph generation is shown in Figure 6.4. From frame g), it appears that Visibility-PRM generates a graph that can provide sub optimal paths in terms of distance traveled. This is discussed in section 6.4.

The generic Algorithm 4 relies on two methods to be functional: GenerateConfiguration and SimplePath. GenerateConfiguration has been introduced in section 6.2.1. We now need to describe how SimplePath is implemented in RB-PRM.

Algorithm 4 Graph construction using the Visibility PRM Algorithm

```
1: function GENERATE-PRM(m: number of tries)
2:     Guard ← ∅; ntry ← 0
3:     G ← ∅ /*Empty graph*/
4:     while ntry < m do
5:         q ← GENERATECONFIGURATION()
6:         q_vis ← NULL; G_vis ← ∅
7:         for all components G_i of Guard do
8:             found ← FALSE
9:             for all nodes q_g of G_i do
10:                if SIMPLEPATH(q, q_g) then
11:                    found ← TRUE
12:                    if q_vis ≠ NULL then
13:                        q_vis ← q_g; G_vis ← G_i
14:                    else
15:                        /*Add two new connections to the graph G*/
16:                        G ← G ∪ {(q, q_g), (q, q_vis)}
17:                        Guard ← Guard \ G_vis
18:                        G_i ← G_vis ∪ G_i
               until found
19:            if q_vis == NULL then
20:                Add {q} to Guard; ntry ← 0
21:            else
22:                ntry ← ntry + 1
23:    return G
```

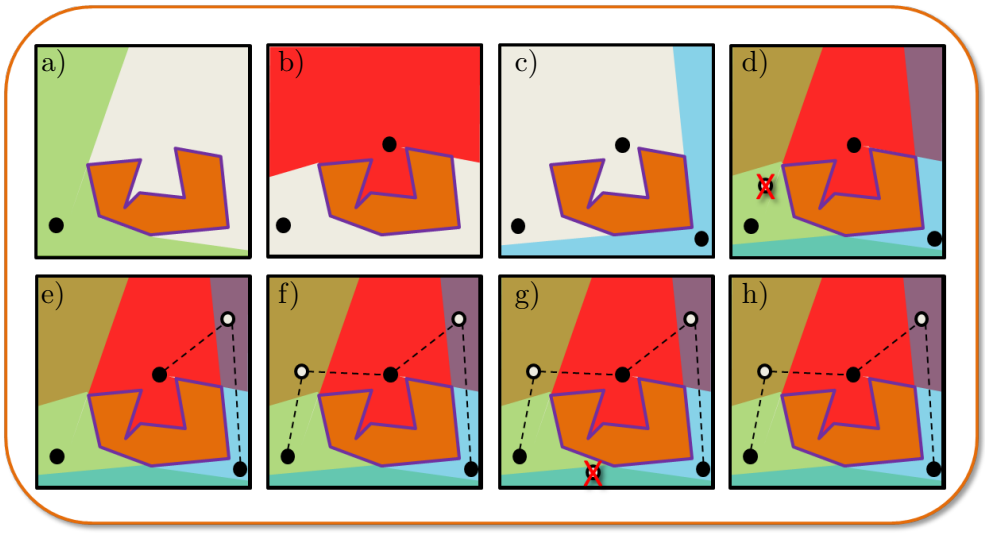

Figure 6.4: Generation process of the Visibility PRM in a simple 2D case. a)b)c) Randomly generated configurations are added to the graph, because they cannot be connected to any existing nodes: they are *guard* nodes. d) A new configuration is rejected because it belongs to the green visibility domain, but does not allow to improve the graph connectivity. e)f) Two *connecting* nodes are added to the graph because they allow to connect independent components of the graph. g) A new configuration is rejected because it fails to improve the graph connectivity. h) The final graph.

6.2.3 Connecting nodes

Local planners, or local steering methods, are used in PRMs to determine whether two configurations can be connected in the graph. They are called "Local" because they use really fast and simple methods to try to connect two configurations. Those methods usually only success if the configurations are close enough, or if the environment separating them is simple.

The most basic implementation of SimplePath consists in drawing a straight line between the two configurations, and using linear interpolation at a fixed step to determine if a collision occurs along the line.

Another simple implementation is the *Rotate-At-s* method. Rather than progressively rotating the configuration using linear interpolation, the configuration is suddenly rotated from its original orientation into the final one at a determined step s.

Those two methods are illustrated in Figure 6.5. Depending on the case, as once again demonstrated by Yamato et al. in [ABL+98], one can find a path when the other cannot. Using both approaches can therefore be beneficial. Our local planner first checks for a connection using a straight line planner and, if that fails, uses a *rotate-at-s* approach, where s is randomly determined.

We add another condition to our local planner. Not only do we ensure that the intermediate configurations are collision-free, we also make sure that all of them verify

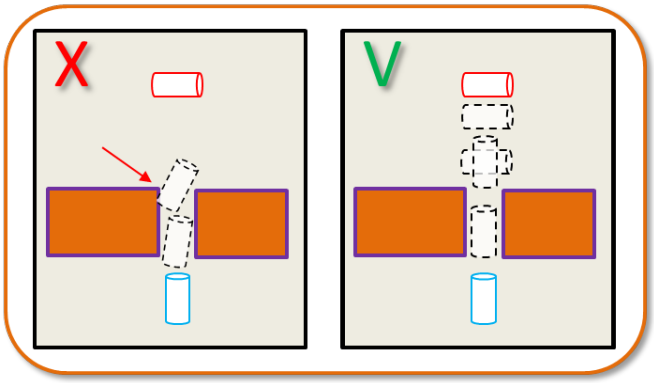

Figure 6.5: Two examples of local planner: straight line planner with linear interpolation (left); rotate-at-s planner (right). In this particular case the straight line planner fails to find a local path where the rotate-at-s succeeds.

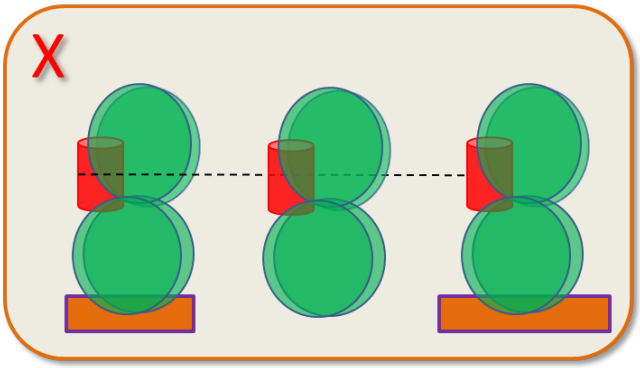

Figure 6.6: If the reachability condition is not verified by the local planner unrealistic paths might be found.

the reachability condition, to avoid situations such as the example shown in Figure 6.6. A consequence of this restriction is the inability to our planner to plan highly dynamic motions such as jumping. However, considering our initial strong hypothesis of highly constrained environments this drawback does not appear to be a strong issue.

6.2.4 Conclusion

In this section we describe the different steps leading to the generation of the RB-PRM graph. This graph is composed of configurations sampled in $C_{reachability}$. A configuration is connected to another one in the graph if two conditions are met:

- A simple collision-free path exists between them;
- All the configurations interpolated along this path also belong to $C_{reachability}$.

The graph we obtain contains a relatively small number of nodes thanks to its design based on the Visibility-PRM.

6.3 Online request and trajectory generation

With an instance of RB-PRM generated for a given workspace W and a virtual character R, we can request the graph for a path between two configurations. As inputs, the request step takes: the graph generated by RB-PRM; a motion task, expressed as start and goal configurations. It outputs a smooth continuous trajectory composed of collision-free configurations from the start to the goal configuration. In this section we explain how the path is requested and how it is transformed into a smooth spline trajectory. Finally we detail the algorithms used to simplify the trajectory.

6.3.1 Path request using the A* algorithm

To perform a path request on a RB-PRM graph G, we use the famous $A*$ algorithm. Its implementation is given in Algorithm 5.

For the algorithm to be complete, we need to provide two methods: ESTIMATE and DISTANCE. ESTIMATE is a heuristic used to estimate the distance between two unconnected nodes, when DISTANCE computes the actual distance between two connected nodes. In this work we use the same method in both case. Therefore in the rest of this chapter we only refer to the DISTANCE method. Once again, we rely on the report from Yamato et al. [ABL+98] to choose an implementation for DISTANCE. We choose to implement DISTANCE as the scaled euclidian distance between two configurations:

$$\text{DISTANCE}(\mathbf{q}_1, \mathbf{q}_2) = (s \sum_{k=x,y,z} |k_{\mathbf{q}_1} - k_{\mathbf{q}_2}| + (1-s) \sum_{k=\alpha,\beta,\gamma} |k_{\mathbf{q}_1} - k_{\mathbf{q}_2}|)^{\frac{1}{2}} \quad (6.3)$$

Where $s \in \mathbb{R}, 0 \leq s \leq 1$ is a user defined parameter.

6.3.2 Path refinement and simplification

The random nature of PRMs often result in suboptimal path, in terms of length and jerkiness of the motion. Several methods exist to simplify a path returned by a PRM like planner. Given a path as input, as well as the maximum velocity and acceleration of the character, the path refinement step returns a smoother spline trajectory.

To achieve this goal, two methods are successively employed: the simple "pruning" method, and the "spline shortcut method".

6.3.2.1 Path pruning

The path pruning algorithm is a simple method to try to remove redundant nodes from a returned path. Given a local planner l, and an ordered path $\mathbf{P} = [\mathbf{q_0}, ..., \mathbf{q_m}]$, a node $\mathbf{q_i}$ is redundant if

$$\exists j, k, 0 \leq j < i < k \leq n, l(\mathbf{q_j}, \mathbf{q_k})$$

Algorithm 5 The A* algorithm

1: **function** A*($\mathbf{q}_{start}, \mathbf{q}_{goal}$)
2: $ClosedSet \leftarrow \emptyset$ //The set of nodes already evaluated.
3: $OpenSet \leftarrow \{\mathbf{q}_{start}\}$ //The set of tentative nodes to be evaluated
4: $came\_from \leftarrow []$ //The map of navigated nodes.
5: $g\_score \leftarrow []$ //Cost from start to a node along best known path.
6: $g\_score[\mathbf{q}_{start}] \leftarrow 0$
7: $f\_score \leftarrow []$ //Estimated total cost from start to goal through a node.
8: $f\_score[\mathbf{q}_{start}] \leftarrow g\_score[\mathbf{q}_{start}] + \text{ESTIMATE}(\mathbf{q}_{start}, \mathbf{q}_{goal})$
9:
10: **while** $OpenSet \neq \emptyset$ **do**
11: //Select element of OpenSet for which f_score is minimal
12: $\mathbf{q}_{current} \leftarrow min(OpenSet, f\_score)$
13: **if** $\mathbf{q}_{current} == \mathbf{q}_{goal}$ **then**
14: **return** RECONSTRUCTPATH($came\_from, \mathbf{q}_{goal}$)

15: $OpenSet \leftarrow OpenSet - \{\mathbf{q}_{current}\}$
16: $ClosedSet \leftarrow ClosedSet \cup \{\mathbf{q}_{current}\}$
17: **if** $\mathbf{q}_{neighbor} \in ClosedSet$ **then**
18: **continue**

19: **for all** nodes $\mathbf{q}_{neighbor}$ in NEIGHBORNODES($\mathbf{q}_{current}$) **do**
20: $try \leftarrow g\_score[\mathbf{q}_{current}] + \text{DISTANCE}(\mathbf{q}_{current}, \mathbf{q}_{neighbor})$
21: **if** $\mathbf{q}_{neighbor} \notin OpenSet \vee try < g\_score[\mathbf{q}_{neighbor}]$ **then**
22: $came\_from[\mathbf{q}_{neighbor}] \leftarrow \mathbf{q}_{current}$
23: $g\_score[\mathbf{q}_{neighbor}] \leftarrow try$
24: $f\_score[\mathbf{q}_{neighbor}] \leftarrow g\_score[\mathbf{q}_{neighbor}] + \text{ESTIMATE}(\mathbf{q}_{neighbor}, \mathbf{q}_{goal})$
25: **if** $\mathbf{q}_{neighbor} \notin OpenSet$ **then**
26: $OpenSet \leftarrow OpenSet \cup \{\mathbf{q}_{neighbor}\}$

27: **return** failure

28: **function** RECONSTRUCTPATH($came\_from, \mathbf{q}_{current}$)
29: $path \leftarrow emptyList$
30: $path.push\_front(\mathbf{q}_{current})$
31: **while** $came\_from[\mathbf{q}_{current}] \neq NULL$ **do**
32: $\mathbf{q}_{current} \leftarrow came\_from[\mathbf{q}_{current}]$
33: $path.push\_front(\mathbf{q}_{current})$
34: **return** $path$

Algorithm 6 defines a pruning algorithm. Figure 6.7 illustrates a benefit of the pruning algorithm.

Algorithm 6 Pruning Algorithm

```
1: function PRUNE(P, l)
2:     for i = 0 to P.length() do
3:         for j = P.length() to i + 2 do
4:             if l(P[i], P[j]) then
5:                 Remove all nodes between i and j
6:                 break
```

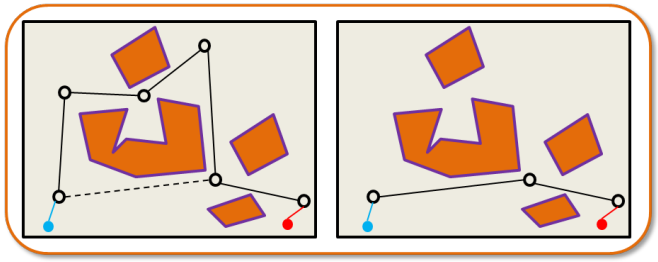

Figure 6.7: Pruning algorithm illustration. Blue circles correspond to the start configuration, and red circles to the target configuration.

6.3.2.2 From a piecewise linear path to a shortcut spline trajectory

A popular way to transform a PRM trajectory into a smoother one is to approximate the trajectory using spline curves. In this step we transform a sequence of configurations **P** with no notion of time into a spline trajectory $q(t)$, respecting the character boundaries in terms of velocity and acceleration. Pan et al. proposed to use the properties of B-splines to further refine a spline trajectory in constrained environment in [PZM12]. This section will give a simple overview of their method which is implemented in this thesis. The interested reader is invited to refer to the paper to obtain more complete details.

Transforming the original path into a B-spline trajectory. Given a path **P**, we consider the interpolation function $P(t), 0 \leq t \leq t_{max}$, defined in Chapter 4. We recall that t_{max} denotes the maximum distance traveled along **P**, and that $P(t)$ returns the interpolated configurations corresponding to a distance t traveled along **P**.

We approximate **P** into a smooth spline $q(t)$.

Figure 6.8 illustrates the process. The idea is to randomly sample $m + 1$ knots $t_i, t_0 = 0 \leq t_1 \leq ... \leq t_m = t_{max}$ and use them to create a B-spline $q(t)$ approximating $P(t)$. If all the configurations of $q(t)$ verify the reachability condition, then $q(t)$ is a valid approximation. Otherwise we increase the number of samples and try again. This

algorithm always converges since, as the number of samples increases, the $q(t)$ becomes more and more similar to $P(t)$, which already verifies the reachability condition.

In the process, the path is transformed into a trajectory. The knots t_i, which refer to the traveled distance along $q(t)$, are reparametrized as time indexes $u_i, u_0 = 0 \leq u_1 \leq ... \leq u_m$ using the maximum velocity of the virtual character's degrees of freedom:

$$u_i - u_{i-1} = \max_{k=x,y,z,\alpha,\beta,\gamma} \frac{\int_{t_{i-1}}^{t_i} \mathbf{q}^k(t)}{\mathbf{v}^k_{max}}$$

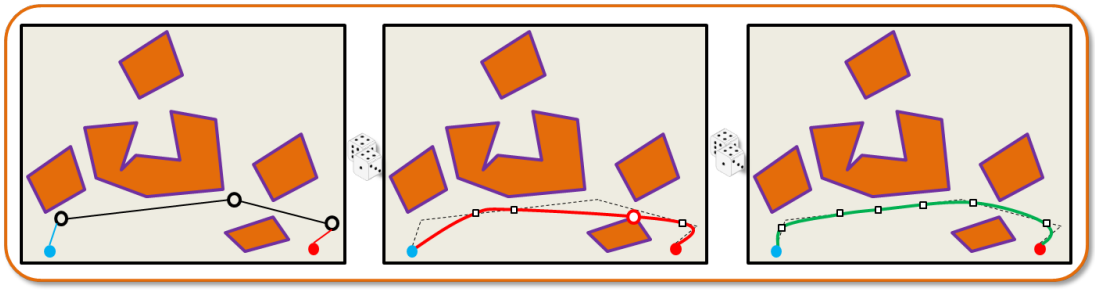

Figure 6.8: Path approximation as a spline trajectory. A random number of knots are sampled along the original trajectory and a spline is created with them. While the spline does not verify the reachability condition (middle), we sample more knots on the trajectory (right).

The spline shortcut algorithm The spline shortcut algorithm proposed by Pan et al. is a generalization of the pruning algorithm to B-splines [PZM12]. It exploits the locality property of B-splines to deform small portions of the curve without altering the whole trajectory. We apply N times the following steps, where N is a user defined variable:

- Two knots u_a and $u_b, 0 \leq u_a < u_b \leq u_m$ are randomly sampled (Figure 6.9 - b);
- using the same process used to transform the original path into a B-spline (with $q(t)$ instead of $P(t)$), a new spline is generated between u_a and u_b;
- if the new spline verifies the reachability collision, it is merged with $q(t)$ (Figure 6.9 - c);
- when the reachability condition is not verified, additional knots are sampled (Figure 6.9 - d,e);

6.3.3 Conclusion

In this section we describe the different steps leading to the computation of a smooth, collision-free trajectory connecting two configurations.

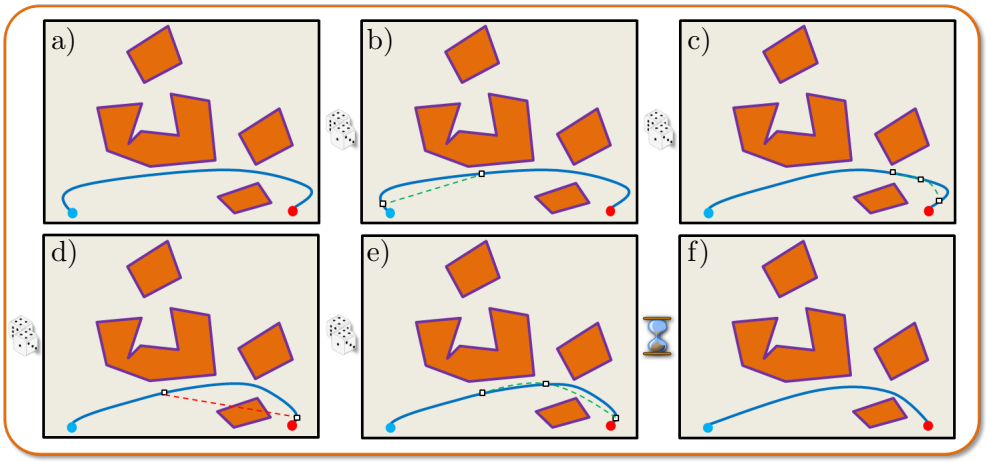

Figure 6.9: Illustration of the spline shortcut algorithm.

First, the RB-PRM graph is requested. The start and goal configurations are temporary added to the graph, and a pathfinding algorithm is run. If a path is found it is returned, otherwise the algorithm fails.

The path is then transformed into a trajectory, including the notion of time. This is done by considering the total distance traveled along the path and the maximum speed the character can travel.

In a last step, the trajectory is approximated and, if possible, simplified, using a spline shortcut algorithm, introduced by Pan et al. in [PZM12].

6.4 Discussion

In this chapter we present the Reachability Based Probabilistic Road Map. RB-PRM is designed for the fast generation of a collision-free trajectory for a character abstraction. The configurations along this trajectory verify the "reachability condition", ensuring their proximity to obstacles. The purpose of RB-PRM is to delay the generation of contact configurations to an ulterior step while computing a promising trajectory for a virtual character. This will allow the creation of contacts configurations suitable for the trajectory. The trajectory produced by RB-PRM is smooth and refined thanks to the implementation of a spline shortcut algorithm.

Other applications could be considered for RB-PRM. For instance it could be used to plan trajectories for flying drones subject to ground proximity constraints. In this section we discuss the implications resulting from the use of RB-PRM.

6.4.1 Interest of RB-PRM over other probabilistic planners

Considering the number of already existing probabilistic planners, it seems appropriate to motivate the need for RB-PRM. This thesis is based on the claim that new heuristics must be proposed to generate plausible contact configurations, and that these heuristics

must consider the task being performed. In this chapter we explain that the PRM approach cannot address this issue correctly because the task is not known when the contacts are generated.

However, one could argue that when using RRT planners, specifically designed for single query planning, the task is known prior to the generation of configurations. However the issue is similar: even if the start and goal configurations are known and give a rough idea about the direction of motion, in a constrained environment the local task to perform remains unknown. A configuration can be connected to several others in the RRT graph, and we have no way of knowing which one will be part of the final trajectory. Therefore we cannot compute task efficient contact configurations at the graph generation stage.

This being said, adapting RB-PRM into a RRT like planner would be trivial.

The choice of basing RB-PRM on the visibility PRM rather than a classical PRM can also be discussed. The visibility PRM has the obvious advantage of usually containing a smaller set of nodes than classical PRM planners. The associated drawback is that the paths returned by the planner might not be the shortest ones. However from our experience, in constrained environments this is not too problematic because there are so many obstacles that the computed trajectory usually goes close to the shortest possible path (Figure 6.10). Furthermore, we also rely on the spline shortcut algorithms to remove unnecessary detours in the returned trajectory.

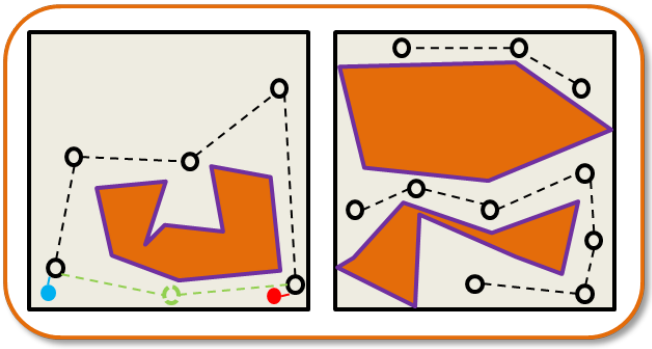

Figure 6.10: Left: While the graph covers the whole workspace W, to go from the blue configuration to the red one, the green path would be much shorter that the detour proposed by the graph. Right: In constrained environments the options are more limited and the problem is less likely to occur.

6.4.2 Genericity and relevance of RB-PRM

One strength of RB-PRM resides in the simplicity of the abstract virtual character used for the graph generation, associated with the reachability condition. Contrary to example based approaches such as motion capture, RB-PRM restricts the research space without suffering from a loss of genericity. In theory any valid solution for a contact based motion planning problem can be found from our planner.

However, the reachability condition is only a heuristic: a configuration verifying the condition is not necessarily a valid contact configuration, mostly from a balance point of view, because the contact creation is delayed. To handle this issue, two non exclusive options can be considered to complete the reachability condition:

- Before adding a new configuration to the graph, assert that a statically balanced configuration can be generated. This can be achieved in many ways, such as [BELK09]. This balanced configuration can then be discarded since it will not be efficient for achieving the task, or stored in the graph as a "backup solution" if balance is critical in the application.
- A second option is to specialize the reachability condition. For instance, it is really easy to ensure that several limbs verify the reachability condition. We could require standing configurations (vertical axis of the configuration loosely aligned with the work vertical axis) to verify the condition for both legs, and bent positions to verify it for all limbs. This increases the chances of generating balanced configurations, although still not guaranteeing it. This is the approach we chose to implement in some of our test scenarios. Additional constraints can be also considered regarding the obstacles: minimum size, normal orientation...

In both cases, the genericity of RB-PRM does not hold because of the additional constraints. The user of RB-PRM must consider these options and weigh the pros and cons associated to them. In any case, RB-PRM remains a kinematic planner, and does not provide guarantees that the returned path will ultimately be balanced all along the trajectory. State of the art methods encounter the same issues as RB-PRM [BELK09, MTP12]. Our advantage over them is expressed in terms of performance and completeness of the approach.

Chapter 7

Stage 2 and 3: Generation of a task efficient contact trajectory

Contents

This chapter addresses the stages 2 and 3 of our framework: we transform a collision-free trajectory for a 6 dofs character abstraction into a contact trajectory for a virtual character. In stage 2 (Figure 7.1 - (2)), we take as inputs: the collision-free trajectory, resulting from stage 1 (Figure 7.1 - (1)); the user defined start and goal configuration of the character. We output a sequence of discrete task efficient contact configurations along the trajectory (Figure 7.1 - (2)). Each configuration is separated by a user defined time step. This stage is the most important step of our motion planner: again, generating task efficient contact configurations is critical to obtain a feasible motion. To ensure this property, the method uses the contact generator introduced in Chapter 5.

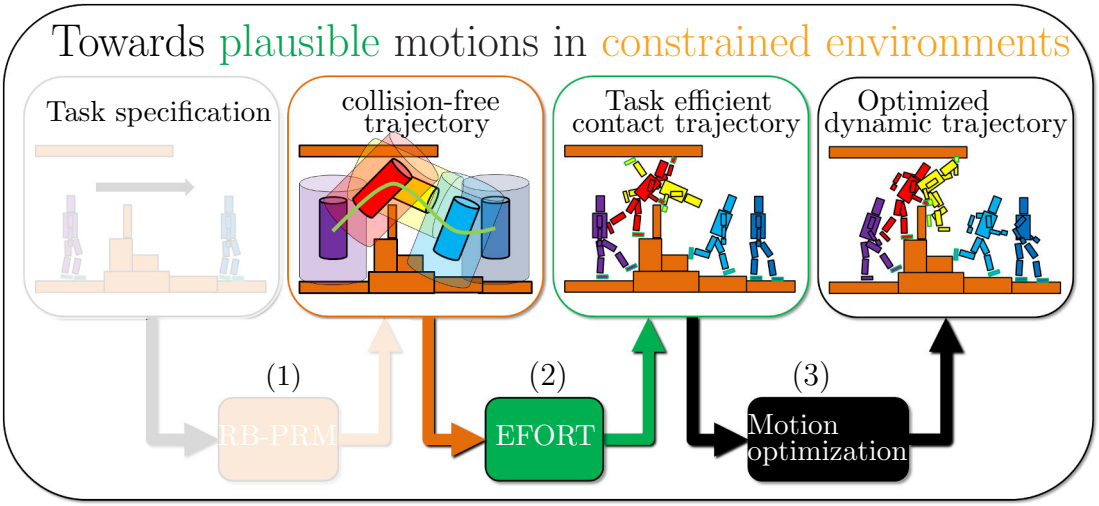

Figure 7.1: In this chapter we cover the last two stages of our framework, the transformation of a collision-free trajectory into a contact trajectory (2 and 3).

At this stage, the complete trajectory of the root of the virtual character is known. This allows additional criteria for contact creation to be considered. Additionally to maximizing the EFORT heuristic, we propose a heuristic for contact duration: it evaluates whether a contact configuration has a reasonable chance to be maintained in the next few frames, and favors them. A third criterion is also proposed: considering the existing contacts, as well as the root velocity and acceleration, it tries to generate contacts which maintain dynamic balance, based on a criterion proposed in [QEMR11].

The contact sequence obtained after stage 2 provides an approximation of the solution motion along the trajectory computed in stage 1. The time information associated with this trajectory must be updated to respect the internal joint velocities limits of the character. This is achieved by considering the joint variation between successive contact configurations. This results in a continuous kinematic trajectory. We are interested in improving it to maintain balance along the motion, and make it smoother.

One way to improve this trajectory is to optimize it using a framework from the literature (Figure 7.1 - (3)). This last stage enforces relevant properties, namely dynamic balance and jerk minimization. The output of stage 3 is the final trajectory used to animate the virtual character.

This chapter is organized as follows: section 7.1 presents two new criteria, integrated within our generator for task efficient contact generation; section 7.2 describes the generation of the contact sequence performed in the second stage; section 7.3 discusses the trajectory optimization performed in the third stage of our framework; section 7.4 concludes with a discussion on these last two stages of our framework.

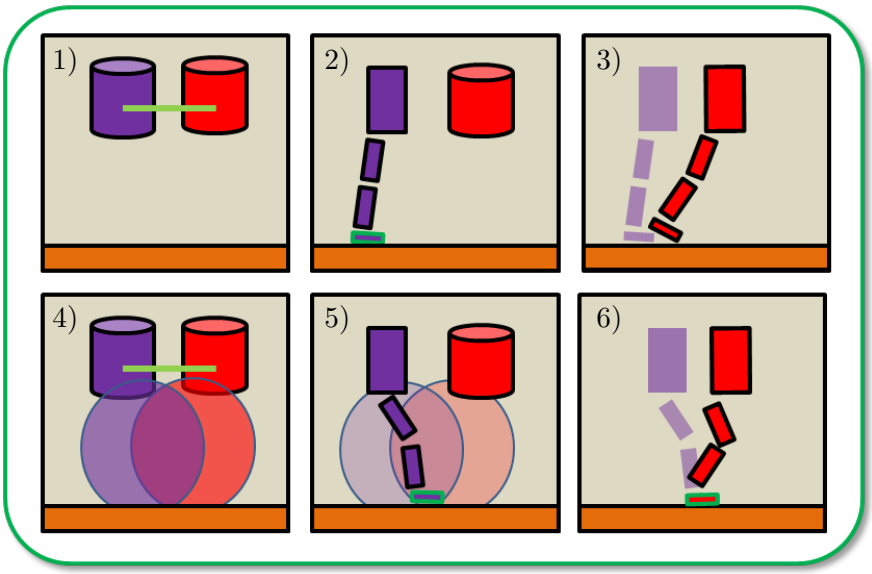

Figure 7.2: From a input trajectory (1), we create a contact for the purple configuration (2). However, this contact does not hold for the next configuration (3). To ensure that the contacts hold for a given period of time, we bias the contact generation towards location included in the Range Of Motion of consecutive configurations (5) (6).

7.1 Two criteria for contact duration and dynamic balance

In Chapter 5, we present the EFORT heuristic, and a real time generator for task efficient contact configurations. This generator is a local method: the selected contact configuration is only chosen based on the current configuration and the local task.

If the trajectory followed by the character is known, it is possible to improve the contact generator with a simple predictive heuristic for contact duration. Considering the velocity and acceleration of the character's root, we can also use a simple heuristic for dynamic balance. We present these two heuristics in this section. For both heuristics, an implicit parameter is the collision-free trajectory $q(t)$, resulting from stage 1 of our planner.

7.1.1 A simple heuristic for contact duration

We define a simple criterion for contact duration, based on the Range Of Motion (ROM) of the virtual character. It asserts that a contact location belongs to the Range Of Motion of the following configurations in the trajectory: this does not guarantee that the location will be reachable, because of the obstacles of the environment, but it discards configurations for which we are sure that they cannot be maintained. This heuristic is proposed to avoid obtaining too many contact changes along the trajectory. The ROM of any limb of a virtual character can be abstracted into a 3D object $\in A_{ROM}$,

as defined in Chapter 4.3.2. We consider two successive configurations $\mathbf{q_1}$ and $\mathbf{q_2}$ for the root of the virtual character, and a given limb R^k. We want to generate a contact configuration ${\mathbf{q_1}}^k \in C^k_{contact}$. We would like it to be maintained when the root of the character is moved towards $\mathbf{q_2}$ (Figure 7.2).

In the absence of obstacles, the set of positions reachable by the end effector of R^k in both configurations is the shape $P^k_{1,2}$, resulting from the intersection of each configuration's associated Range Of Motion (Figure 7.2 - 4):

$$P^k_{1,2} = A^{\mathbf{q_1}}_{ROM_k} \cap A^{\mathbf{q_2}}_{ROM_k} \tag{7.1}$$

We propose a heuristic which gives a higher score to contact positions $\mathbf{p}_{{\mathbf{q_1}}^k}$ contained by $P^k_{1,2}$.

$$\alpha_{duration}({\mathbf{q_1}}^k, \mathbf{q_2}) = \begin{cases} 1 & \text{if } \mathbf{p}_{{\mathbf{q_1}}^k} \in P^k_{1,2} \\ 0 & \text{otherwise} \end{cases} \tag{7.2}$$

This heuristic can be generalized to evaluate the contact duration over a period of time resulting in w successive configurations $\mathbf{q_1}, ..., \mathbf{q_w}$. To do so we consider the geometric series $\alpha^i_d, 0 < i \leq w$:

$$\begin{array}{ll} \alpha^1_d({\mathbf{q_1}}^k) & = 1 \\ \alpha^i_d({\mathbf{q_1}}^k) & = \alpha^{i-1}_d({\mathbf{q_1}}^k)\alpha_{duration}({\mathbf{q_1}}^k, \mathbf{q_i}) \end{array} \tag{7.3}$$

We then define the generalized contact duration heuristic:

$$\alpha^w_{duration}({\mathbf{q_1}}^k) = \frac{\sum_{i=2}^{w} \alpha^i_d({\mathbf{q_1}}^k)}{w-1} \tag{7.4}$$

$\alpha^w_{duration}({\mathbf{q_1}}^k)$ is a normalized function which considers a possible contact location for w configurations: the upper term of equation 7.4 is equal to the number of successive configurations, starting from $\mathbf{q_2}$, for which the end effector position of ${\mathbf{q_1}}^k$ is reachable.

- If $\alpha^w_{duration}({\mathbf{q_1}}^k)$ returns 0, it means that the contact location cannot be reached by any configuration other than $\mathbf{q_1}$.
- If $\alpha^w_{duration}({\mathbf{q_1}}^k)$ returns 1, it means that the contact location is potentially reachable by all of the w configurations.

To implement this heuristic, the Range Of Motion A_{ROM_k} is approximated into a sphere to simplify collision checking. The chosen sphere is the Chebyshev sphere. As defined in Chapter 4.1.2, it is the largest sphere contained by A_{ROM_k}.

7.1.2 A heuristic for dynamic balance

In order to bias the contact generation towards balanced configurations, we implement a criterion proposed by Qiu et al. in [QEMR11]. We present its principles.

To maintain balance, given a set of exterior forces, the considered constraints are the following:

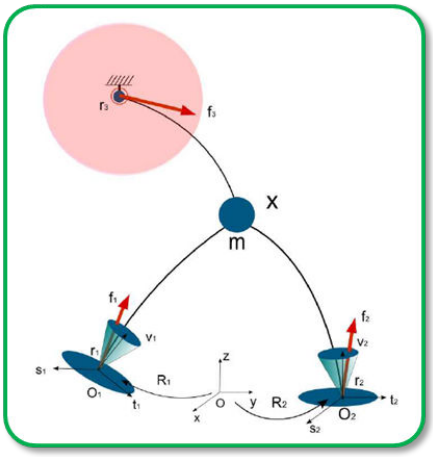

Figure 7.3: An example of a simplified model for the balance criteria: a point mass with two non-coplanar contacts and one grasp. This figure is reproduced from [QEMR11]

- Contact forces must respect the condition of no slipping, given by the friction cone from Coulomb's law (Figure 7.3 – $\mathbf{f_1}$ and $\mathbf{f_2}$);
- Grasping forces are bounded by a user defined maximal force applicable by a given limb (Figure 7.3 – $\mathbf{f_3}$).

Assuming a simplified model, where the mass m of a virtual character is concentrated in one point $\mathbf{x}$ (Figure 7.3), the dynamic perturbation $\mathbf{w}$ is given by Newton's second equation:

$$\mathbf{w} = m\ddot{\mathbf{x}} - m\mathbf{g} = \sum_{i=0}^{n_t} \mathbf{f}_i \tag{7.5}$$

where $\mathbf{g}$ is the gravity and n_t is the number of active exterior forces. The authors prove that the set of dynamic perturbations satisfying the constraints can be written as a polytope:

$$\varpi = \{\mathbf{w} \in \mathbb{R}^3, \mathbf{H}(\mathbf{x})\mathbf{w} \leq \mathbf{h}\}$$

If for a given $\mathbf{x}$, we have $\mathbf{H}(\mathbf{x})\mathbf{w} \leq \mathbf{h}$, then the configuration is stable.
We then define the balance heuristic:

$$\alpha_{balance}(\mathbf{q}) = \begin{cases} 1 & \text{if } \mathbf{w} \in \varpi \\ 0 & \text{otherwise} \end{cases} \tag{7.6}$$

Where $\mathbf{w}$ is given by the trajectory function $q(t)$.

7.2 Stage 2: Extension a collision-free trajectory into a task efficient contact sequence

In stage 2, as an input, we consider the collision-free trajectory computed in stage 1. The output is a discrete sequence of contact configurations along this trajectory.

This extension is achieved by the algorithm based on the contact generator described in chapter 5. Given the position and orientation of the root, a limb, and a local force exertion task, this generator returns a contact configuration which maximizes the EFORT heuristic.

Additional work is necessary to automatically provide this set of inputs. The position and orientation of the root is given by the trajectory computed in the first stage of our planner (Figure 7.1 - 1). However, we need to define a strategy to determine when to create a contact for a given limb and when to stop maintaining the contact. To achieve this, we propose an iterative algorithm based on simple principles: we create contacts when the current configuration is not balanced; and we stop maintaining them (or break them) based on kinematic heuristics. Furthermore, we also need to generate configurations for the limbs which are not in contact, to make sure they are collision-free.

Finally, we modify the contact generator to include the new heuristics proposed in section 7.1.

7.2.1 Extension algorithm.

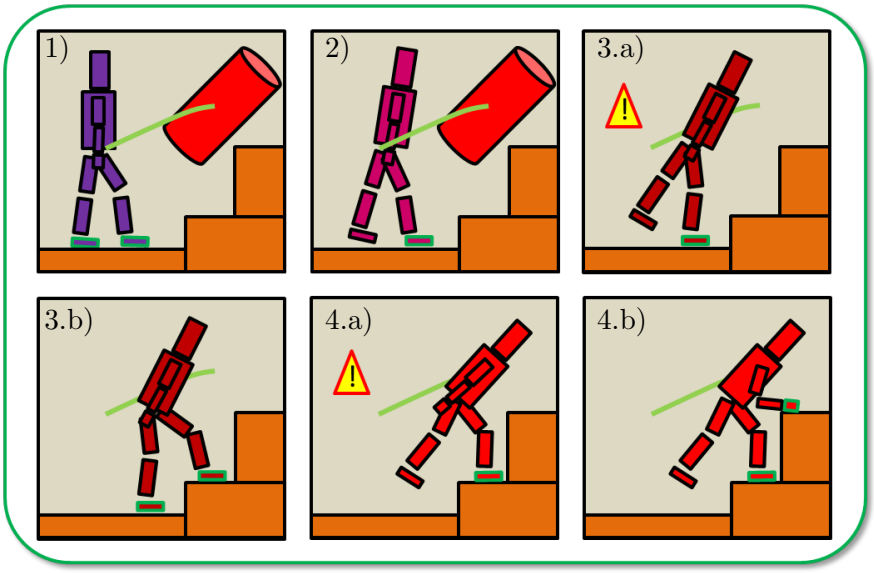

Figure 7.4: Generation of a contact sequence. We consider the previously computed configuration (1), and try to maintain the contacts for the new configuration (2). If the configuration is balanced, we move on to the next configuration. If it is not (3.a, 4.a), we generate additional contacts (3.b, 4.b).

Algorithm 7 describes the method used to extend a collision-free trajectory into a contact sequence (Figure 7.4).

Algorithm 7 Extension of a collision-free trajectory into a contact sequence

function GENERATECONTACTTRAJECTORY($\mathbf{q}_{start}$, $\mathbf{q}_{goal}$, $q(t)$, Δt)
  $n_{states} \leftarrow 1/\Delta t$ /* number of states in trajectory */
  $\mathbf{T}$: *Contact sequence: matrix of size* $n \times n_{states}$
  $\mathbf{T}[\_, 0] \leftarrow \mathbf{q}_{start}$
  $\mathbf{T}[\_, n_{states} - 1] \leftarrow \mathbf{q}_{goal}$
  $i \leftarrow \Delta t$
  while $i < n_{states}$ **do**
    $\mathbf{q}_{old} = \mathbf{T}[\_, i - 1]$
    $\mathbf{q}^r \leftarrow q(i\Delta t)$
    /* compute translation and orientation variation between configurations */
    $\Delta\mathbf{p} \leftarrow \mathbf{q}^r - \mathbf{q}^r_{old}$
    $\mathbf{q}_{new} \leftarrow$ GENERATEFULLBODYPOSTURE($\mathbf{q}_{old}, \Delta\mathbf{p}$)
    $\mathbf{T}[\_, i] \leftarrow \mathbf{q}_{new}$
    $i \leftarrow i + 1$
  return T

function GENERATEFULLBODYPOSTURE($\mathbf{q}_{old}$, $\Delta\mathbf{p}$)
  /* *create new configuration from previous state* */
  $\mathbf{q}_{new} \leftarrow \mathbf{q}_{old}$
  $\mathbf{q}^r_{new} \leftarrow \mathbf{q}^r_{new} + \Delta\mathbf{p}$
  $\mathbf{v_t} \leftarrow \Delta\mathbf{p}^{x,y,z}$ /*Compute translation task*/
  $\mathbf{v_t} \leftarrow \mathbf{v_t}/\|\mathbf{v_t}\|$ /*Normalize the task vector*/
  for $k = 0, ..., l$ do **do**
    /* try to maintain contacts from previous states*/
    if $\mathbf{q}^k_{old} \in C^k_{Contact}$ **then** MAINTAINCONTACT($\mathbf{q}_{new}, \mathbf{p}_{\mathbf{q}^k_{old}}, k, \Delta\mathbf{p}$)
  /* If the new configuration is not stable create new contacts */
  if !DYNAMICALLYSTABLE($\mathbf{q}_{new}$) **then**
    for $k = 0, ..., l$ do **do**
      if !($\mathbf{q}^k_{new} \in C^k_{Contact}$) **then**
        GENERATECONTACTCONFIGURATION($\mathbf{q}_{new}, k, \Delta\mathbf{p}$)
  /* Find a suitable collision-free collision for limbs not in contact */
  for $k = 0, ..., l$ do **do**
    if ($\mathbf{q}^k_{new} \in C^k_{Obs}$) **then**
      GENERATECOLLISIONFREECONFIGURATION($\mathbf{q}_{new}, k, \mathbf{dt}$)
  return $\mathbf{q}_{new}$

Before going further, we recall that:

- The complete configuration of a virtual character R is given by the vector $\mathbf{q} \in \mathbb{R}^n$;
- The position and orientation of the root of the character is given by the vector $\mathbf{q}^r \in \mathbb{R}^6$;
- The position of the root of the character is given by the vector $\mathbf{q}^{x,y,z} \in \mathbb{R}^3$;
- The configuration of a limb k is given by the vector $\mathbf{q}^k, 0 \leq k \leq l$.

These definitions are detailed and illustrated in Chapter 4. The algorithm proceeds in an iterative fashion. First, we choose a time step Δt. Considering the normalized trajectory $q(t)$ connecting the start and goal configurations ($q(0) = \mathbf{q}^r_{start}$ and $q(1) = \mathbf{q}^r_{goal}$), we choose $n_{states} = 1/\Delta t$ configurations, representative of the trajectory. Then the following steps are repeated:

1. At step i, we set $\mathbf{q}^r = q(i\Delta t)$. $\mathbf{q}^r$ describes the position and orientation of the new configuration. We consider the set of contacts existing at the previous configuration $\mathbf{q}((i-1)\Delta t)$, and we try to maintain them; In Figure 7.4 - 1) and -2), we can see that a contact is maintained for one foot, and broken for the other. This is achieved by the method MaintainContact. It tries to perform inverse kinematics on the considered limb to move the end effector towards its previous position. If the target position does not belong in the range of motion of the limb, or if the inverse kinematics fails to find a collision-free solution, the contact is broken;
2. Once this is done, we call the method DynamicallyStable on the global configuration. If the method returns true, it means the configuration is balanced. In this case, the configuration is valid (Figure 7.4 - 2). If not, we try to generate new contacts by calling the method GenerateContactConfiguration (Figure 7.4 - 3 and 4);
3. In the last stage of the algorithm, we consider all the limbs which are not in contact and might be colliding with the environment. The method GenerateCollisionFreeConfiguration looks for the collision-free configuration that is the closest to the current limb configuration and assigns it to the limb.

The process is repeated until the goal configuration is reached.

As an output of Algorithm 7.4, we obtain a sequence of discrete contact configurations between the start and goal configurations. The time frame of this sequence is given by the collision-free trajectory resulting from the first stage of our framework(Figure 7.1 - (1)). It does not consider the time constraints associated with the internal joint velocities. To update the time frame separating two successive configurations, we reapply the spline shortcut algorithm described in Chapter 6.3.2.1. This operation considers the joint angle variation between two configurations, and computes the minimum time necessary to achieve the interpolation without violating the joint velocities limits. If

this value is greater than the current time frame, the time parameter of the sequence in updated. As a result the time separating two consecutive configurations is greater than the minimum time necessary to achieve their interpolation, while respecting the joint velocities limits imposed on the model. This step also allows us to define continuous spline trajectories interpolating the motion between two configurations. Using an inverse kinematics solver then allows us to obtain a continuous kinematic trajectory.

7.2.2 A modified contact generator, including new heuristics for task efficiency

The method GenerateContactConfiguration of Algorithm 7 performs a call to the contact generator presented in Chapter 5. The generator considers a set of candidate contact configurations, and returns the one maximizing the EFORT heuristic. We bring a small modification to the generator: we integrate the heuristics for contact duration and balance introduced in section 7.1:

$$\alpha(\mathbf{q}^k, \mathbf{v_t}) = c_0\alpha_{EFORT}(\mathbf{q}^k, \mathbf{v_t}) + c_1\alpha^w_{duration}(\mathbf{q}^k) + c_2\alpha_{balance}(\mathbf{q}^k) \tag{7.7}$$

where the c_X are weighing variable.

7.3 Stage 3: Computing the final trajectory

As an output of the second stage of our algorithm, we obtain a contact trajectory which describes a sequence of contact configurations achieving the motion from a start and a goal configurations.

This trajectory is destined to be used as a guide input for a trajectory optimization framework. In the context of an ongoing collaboration with the Gamma Lab at University of North Carolina (UNC), we chose the ITOMP optimization framework proposed by Park et al. in [PPM12].

In this section we first present the ITOMP framework, then we describe the integration of our method within this framework.

7.3.1 Presentation of ITOMP and motivation.

ITOMP is a trajectory optimization framework for characters comprising a high number of degrees of freedom in dynamic environments. Given a start configuration $\mathbf{q}_{start}$ and a goal configuration $\mathbf{q}_{goal}$, ITOMP outputs a smooth and physically accurate trajectory.

To achieve this, first an initial trajectory $\mathbf{T}_{init}$ is generated by discretizing the configurations along a curve connecting $\mathbf{q}_{start}$ and $\mathbf{q}_{goal}$ into $n+1$ segments equally spaced in time: $\mathbf{T}_{init} = [\mathbf{q}_{start}, \dots \mathbf{q}_n, \mathbf{q}_{goal}]$.

ITOMP optimizes these internal waypoints according to the cost function:

$$\min_{\mathbf{c}_1,\dots,\mathbf{c}_n,\mathbf{q}_1,\dots,\mathbf{q}_n} \sum_{i=1}^{n} (C_{obs}(\mathbf{q}_i) + C_{spec}(\mathbf{q}_i, \mathbf{c}_i)) + \frac{1}{2}\|AQ\|^2$$

where

- $\|AQ\|^2$ is a cost function evaluating the smoothness of the overall motion. It is computed as the sum of squared accelerations along the trajectory;
- C_{obs} is a cost function penalizing collisions;
- C_{Spec} denotes a set of context specific cost functions.
- $\mathbf{c}_i = [c_i^0, \ldots, c_i^l]$ denote a set of contact related variables with the following semantics: if c_i^k is large, then the end effector of the limb R^k must be in contact for configuration $\mathbf{q}_i$; otherwise whether there is a contact or not is not relevant.

In the initial implementation of ITOMP, C_{Spec} addresses two features:

- Physics violation ($C_{Physics}$): As in section 7.1.2, a set of constraints for balance is computed, based on maximum force exertion and Coulomb's law; To determine the violation cost, an inverse dynamics resolution is performed using a quadratic programming method; The $\mathbf{c}_i$ contact variables are included in the cost function in such a way that the larger their value is, the lower $C_{Physics}$ is. This makes sense because it means that the effectors are in contact, which allows the creation of the forces necessary to preserve balance.
- Contact Invariant Optimization (C_{CIO}, based on [MTP12]): the cost function is expressed as the distance separating the end effectors and the closest point in the obstacles surrounding them, weigthed by the contact variables $\mathbf{c}_i$. C_{CIO} equals 0 if all the contact variables equal 0. Otherwise, the closest the end effectors are to an obstacle (and from contact), the lower C_{CIO} is.

We can observe that C_{CIO} and $C_{Physics}$ have opposed objectives: C_{CIO} aims at minimizing the value of the contact variables towards 0, while $C_{Physics}$ aims at increasing them to ensure physical accuracy. The obtained motion results from a compromise between these two cost functions, as a trajectory which minimizes the number of contacts while remaining physically accurate.

As for many optimization frameworks, ITOMP is not directly adapted to motion planning in constrained environments. Because of the complexity and the number of obstacles, the optimization can easily fall into local optima and fail to produce a solution. The objective of our collaboration is to provide ITOMP with relevant guide trajectories to adapt it to constrained environments.

7.3.2 Adaptation of ITOMP into our framework

To fully integrate ITOMP as the third stage of our framework, two modifications are required: first, the initial trajectory $\mathbf{T}_{init}$ is replaced by the contact trajectory resulting from stage 2. This step is immediate and does not require strong modifications of the framework.

Furthermore, an additional cost function is added to the set C_{Spec}: In order to maintain the task efficiency of the computed contact configurations, the EFORT heuristic is integrated.

We define C^k_{EFORT}, the cost function associated to a specific limb R^k:

$$C^k_{EFORT}(\mathbf{q}_i, \mathbf{c}_i, \mathbf{v_t}) = c^k_i / \alpha_{EFORT}(\mathbf{q^k_i}, \mathbf{v_t}) \tag{7.8}$$

Integrating c^k_i in the equation allows to ensure that EFORT is maximized when the limb R^k is in contact, and limits its influence when it is not.

We then define the global cost function:

$$C_{EFORT}(\mathbf{q}_i, \mathbf{c}_i) = \sum_{k=0}^{l} C^k_{EFORT}(\mathbf{q}^k_i, \mathbf{q}^{\{x,y,z\}}_{i+1} - \mathbf{q}^{\{x,y,z\}}_i) \tag{7.9}$$

7.4 Discussion

In this chapter, we first presented an algorithm for transforming a continuous collision-free trajectory into a sequence of discrete task contact configurations. The input trajectory results from the stage 1 of our motion planning solution. It uses a combination of three heuristics to generate contact configurations along the trajectory: the first heuristic maximizes the ability to apply an important force allowing to perform the motion; the second heuristic tries to ensure that the created contacts can be maintained for a user defined time horizon; the last heuristic tries to ensure that the created contacts maintain dynamic balance.

Then, we present the ITOMP optimization framework, and describe how it is integrated as the third stage of our motion planner: the contact sequence is used as an input trajectory to guide the optimization process and avoid local minima. Additionally, the EFORT heuristic is integrated as a cost function within the optimization process to task efficiency. ITOMP transforms the contact sequence into a smooth, physically accurate trajectory. At the time of writing this thesis, the integration with ITOMP is not fully functional.

The combination of the stages 1 and 2 allows us to address the objectives of this thesis. Given a constrained environment, we are able to compute a trajectory, discretized as a sequence of task efficient contact configurations.

The main strength of the approach lies in the fact that the trajectory is known before the contacts are created, which makes it possible to use new heuristics for task efficiency in the contact creation phase.

The heuristics proposed in this chapter are useful in this regard, however they could be improved in several ways:

- The heuristic for contact duration is useful to prevent generating contacts which would be broken immediately after being created. However, the heuristic does not consider the environment in the evaluation of the validity of a configuration, and could be improved in this regard;

- The heuristic for balance does not provide guarantees that the resulting configuration will actually be balanced.

Similarly, the contact generation algorithm relies on rather simple rules to decide when to create or break a contact.

These drawbacks are not inherent to the approach, and future research will try to improve these aspects of the method. We believe that, as for EFORT, proposing new heuristics for contact duration, as well as proposing relevant strategies for deciding when to create and break contacts, is a promising and complex research subject, which will be addressed in future work.

This being said, the method remains one of the few to address our objectives examples. Furthermore, the resulting trajectory can be used as a relevant input for an optimization framework. In this final phase, physics accuracy and smoothness of the trajectory can be enforced. Additionally, we propose a new cost function to optimize the contact configurations and locations according to the EFORT heuristic.

Chapter 8

Results

This chapter reviews the results obtained throughout this thesis. It is divided in three parts: in section 8.1, we detail the results obtained with a "stand alone" use of our contact generator in real time scenarios, as presented in Chapter 5; in section 8.2, we depict the outcome of the first two stages of our motion planning framework, presented in Chapters 6 and 7. These first two sections present implementation details, screenshots of the obtained motions, as well as a performance analysis section. A discussion on the results obtained follows in section 8.3. Finally, in section 8.4, we discuss the ongoing integration of our method with the ITOMP optimization framework.

The difference of style between the images shown in the first two sections is explained by a technical reimplementation of the framework.

8.1 Stand alone use of our task efficient contact generator

In this section we consider one application of our contact generator for local planning in real time applications. The task to be accomplished is either provided along with the scenario, or inputed by a user with a joystick. It is represented in the figures with a black arrow, indicating the direction of motion.

In this section we first provide some details about the implementation of the test framework. Then, we present the results obtained in different scenarios, before comparing them with simple heuristics. The section is concluded with the presentation and discussion of the performances obtained.

8.1.1 Implementation details

The test application was developed using the C++ language.

Environments are described in the obj format, while virtual creatures and scenarios are described using custom xml files. Rendering is achieved using the OpenGl API. No other third-party libraries were used. The runs were performed on a laptop with an Intel Core i7-2760QM 2.40GHz processor and 4 GB of memory. The application is not multi-threaded.

8.1.2 Test scenarios

We consider a virtual creature in a constrained environment. Six coordinates describe the position and orientation of its root. By default, the initial limb configuration is the reference posture of the creature (as shown for instance in Figure 8.1). We consider a directional task, and one or several limbs of the creature. We then use our method to compute a task efficient contact configuration.

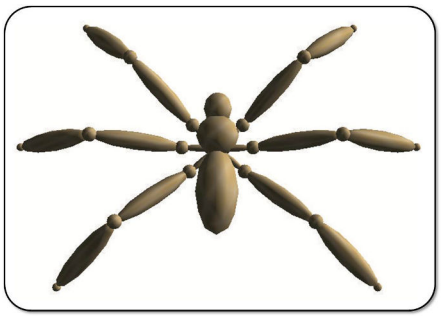

Figure 8.1: Reference posture of a virtual insect composed of 6 limbs. Each limb has 5 degrees of freedom.

Pushing and pulling objects (Figure 8.2 and Figure 8.3). The creature is a virtual human. Two environments are used: In the pushing scenario, the environment consists in a cupboard and the ground; in the pulling scenario it consists in a cupboard, the ground, as well as a rope attached to the cupboard and a small wall. The task consists in pulling (pushing) the cupboard. We formulate the task as a horizontal vector, and compute task efficient configurations for the arms and the left leg of the human. The right foot is already in contact.

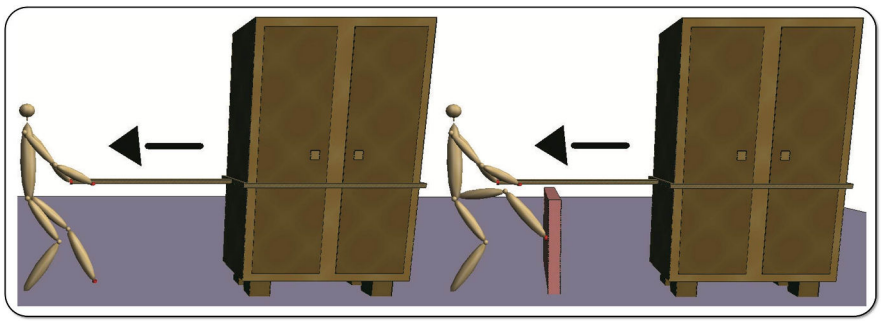

Figure 8.2: Configurations found for a pulling task. In the right figure, our creature uses the pink wall as a better support for the foot. The asymmetry between the arm configurations is induced by the sampling phase.

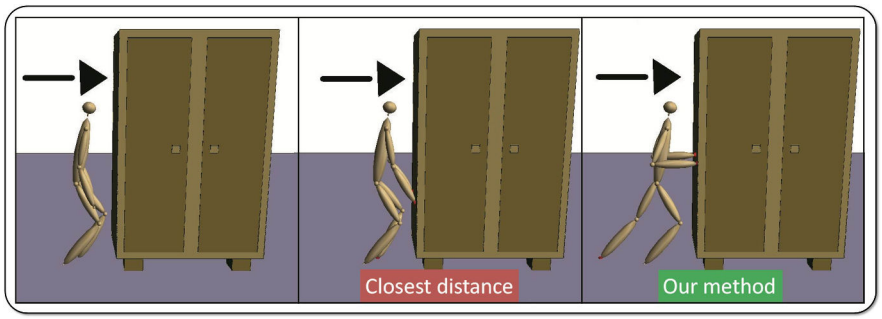

Figure 8.3: In this example of pushing a cupboard, our method (right) is compared with the closest distance heuristic (middle). The closest distance heuristic places the hands and left feet at locations close to their original positions (left) while our method places the end-effectors in configurations relevant for the pushing task.

Multi-limb creatures in constrained environments (Figure 8.4). The environment is composed of a challenging set of books placed on a bookshelf. The creature is an insect with six limbs (Figure 8.1). The input is a forward directional task. This example shows that our method is generic and can be applied to arbitrary creatures, as opposed to example-based approaches. In this example the trajectory followed by the insect is a manual input and does not result from a motion planning method. Similarly, the change of contact stances is determined by an *ad-hoc* controller.

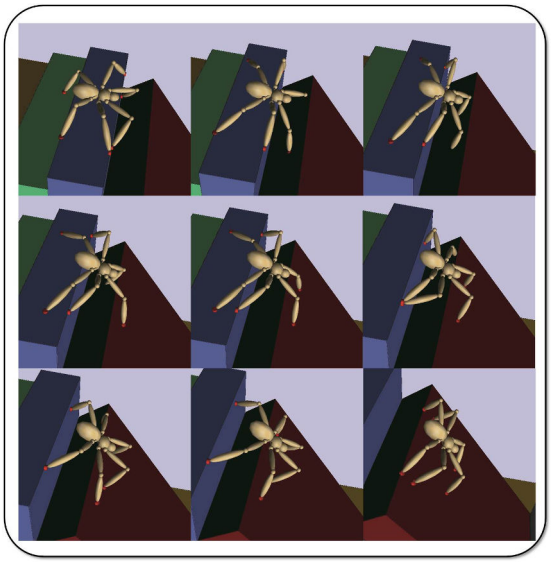

Figure 8.4: Configuration sequence for an insect with 6 limbs crossing a bookshelf. Task efficient contact configurations are found along the trajectory.

Getting up (Figures 8.5, 8.6). The environment is composed of a sofa in a first example, a chair and a table in a second one. In the initial configuration, depending on the environment, the human is sitting either on the sofa or the chair. We formulate the task of getting up as a vertical vector. These examples show the adaptability of our method: the same task in different environments results in different configurations that take advantage of the reachable obstacles.

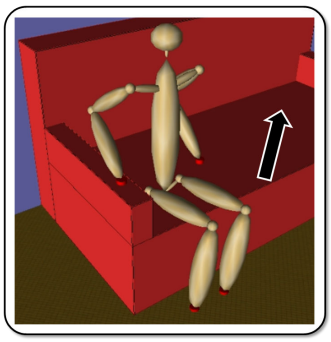

Figure 8.5: Computed configuration for the four limb of a standing up virtual human.

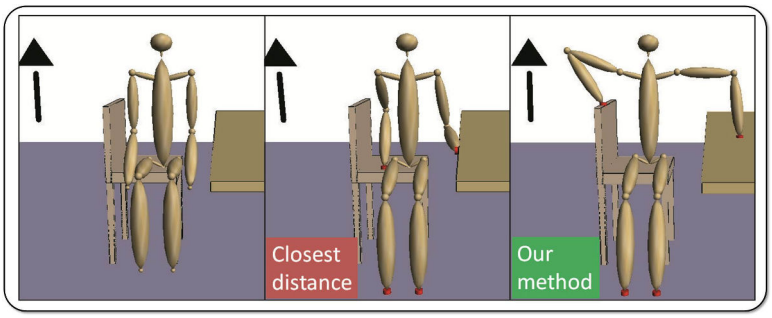

Figure 8.6: Our method (right) is compared with the closest distance heuristic (middle) in this example of getting up from a chair. In the latter case, the left hand position (on the side of the table) is not suitable to generate a vertical effort.

8.1.3 Comparison against the closest distance heuristic

Comparing the results obtained by our method is not trivial because few methods perform the real time automatic computation of contacts: Several previous contributions only address cyclic motions such as walking [Joh09, LP12]. Hauser et al. manually predefine the set of possible contacts [HBL05]. Bretl et al. use a form of manipulability integrated in a motion planner [BRL+04]. Mordatch et al. use a closest distance approach as part of an optimization loop that takes several minutes to compute a result [MTP12]. Therefore we choose to compare the results we obtained with this closest distance heuristic.

In Figure 8.6 the environment consists of a chair, the ground and a table. We compare our method with the closest distance heuristic. The configuration of the left arm in particular seems more appropriate to generate a vertical effort with our method.

In Figure 8.3 the environment consists of a cupboard and the ground. The task for a virtual human is to push the cupboard. In the middle we can see that the results provided by the closest distance heuristic are highly determined by the original location of the end-effectors. Our method, on the other hand, creates contact configurations relevant for the pushing task.

In Figure 8.7 the environment is a climbing wall. The creature is a virtual human. The initial configuration is the reference posture –Figure 8.7 (middle)–. The task consists in navigating along the wall in arbitrary directions. This example shows the advantage of our method over heuristics such as the closest distance, or even manipulability based heuristics such as [BRL+04], because the selected configurations vary according to the motion task.

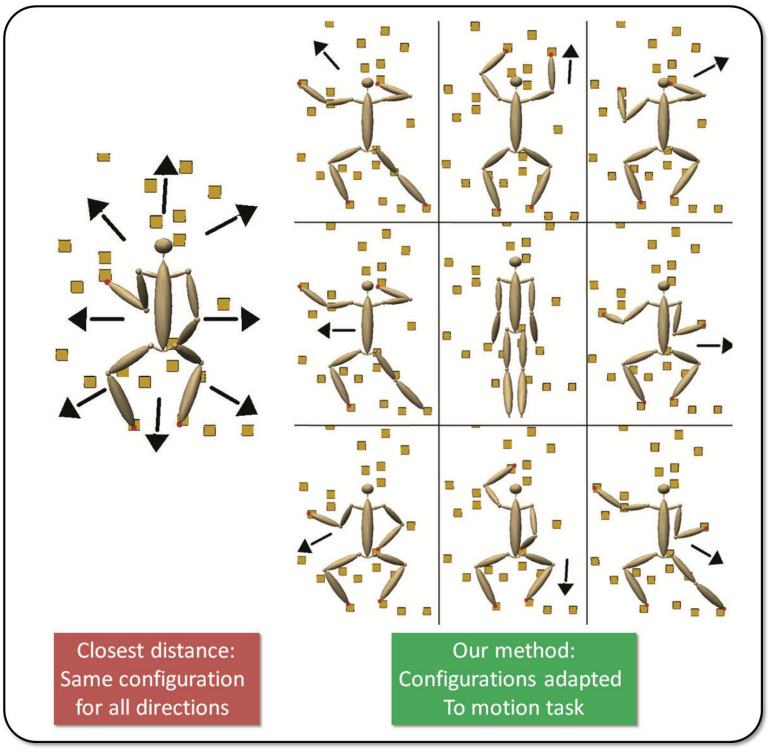

Figure 8.7: Configurations for a humanoid on a climbing wall. Left: the closest distance heuristic does not consider the motion task, therefore it always computes the same configuration. Right: From the same initial root location (position and orientation), different configurations are computed depending on the task (black arrow).

	N = 1 000	N = 10 000	N = 100 000
Human climbing	**1** (3)	**2** (6)	**3** (30)
Getting up	**4** (7)	**5** (60)	**154** (856)
Insect locomotion	**1** (4)	**8** (40)	**55** (370)
Pushing / pulling	**1** (1)	**5** (6)	**34** (70)

Table 8.1: **Average time** (worst time) (in ms) spent for the generation of one contact (in step 3 of our framework) relative to the scenario and the number of samples N.

	N = 1 000	N = 10 000	N = 100 000
Human climbing	0	1	26
Getting up	41	49	3442
Insect locomotion	20	312	2553
Pushing / pulling	13	142	1387

Table 8.2: Average number of contact configurations found (in step 3 of our framework) relative to the scenario and the number of samples N.

	N = 1 000	N = 10 000	N = 100 000
Generation time	256	1500	18000

Table 8.3: Average time (in ms) spent generating samples (in step 3 of our framework) relative to the number of samples N.

8.1.4 Performance analysis

Table 8.1, Table 8.2 and 8.3 provide the performances obtained using our contact generator in the presented scenarios. To analyze them, the reader should have in mind a few details about the method presented in Chapter 5, recalled here. Our method is based on a sampling approach. Independently from the environment, a number N of samples are generated in an offline step. In the online phase, the method compares these samples and select the most relevant one given the current configuration and task. N has a strong influence on the performance of the method: the higher its value, the more likely the method will find good configurations, but the slower it will be. Therefore, we are interested in finding a value for N that will be as low as possible while maintaining an acceptable quality in the results obtained.

We have observed that in our scenarios, the number of samples N has a limited influence on the average maximum value of the EFORT heuristic. Consequently we focus on the relationship between N and the number of candidate configurations found. Table 8.1 shows the time spent for one call to our method. Table 8.2 presents the average number of contact candidates returned by a spatial request. Table 8.3 shows the time spent during the offline step relative to the number of samples generated. It is interesting to note that even for $N = 100000$, the generation time is acceptable, since this step is only performed once. Parallelization and code optimization could probably allow to obtain better performances, but the interest of doing this is limited.

We observe that for $N \leq 10000$, the computation time is short and the average

number of candidates is satisfying. The human climbing scenario is an exception: a higher number of samples is necessary to find enough contact candidates. This is because the environment is composed of a small set of small obstacles.

Looking at the worst performances, we observe a correlation between the time spent in the method and the maximum number of hits obtained. This is explained by the growing number of requests that must be made. In each scenario, the number of triangles is about the same (a hundred); the performance variation is explained by their spatial distribution. For the getting up scenario for instance, setting $N = 1000$ is a reasonable choice, where a value of $N = 100000$ seems more appropriate in the human climbing scenario. In the case of the insect locomotion scenario, a smaller amount of samples suffices to correctly cover the Range Of Motion of the limb. The limited range of motion of each of the insect's limbs explains this fact.

Under the appropriate conditions on the number of samples, we observed that the framerate never went below 52 fps even in the worst case scenarios, including: control of the character, contact generation along the trajectory, and rendering.

Finally, we observe that the memory occupation grows linearly with the number of samples, and remains in reasonable ranges (from 2 MB for 10 000 samples to 166 MB for 1 000 000 samples).

8.2 Computation of the contact trajectory

We present in this section the guide trajectories produced by combining the first two steps of our method. In three different challenging environments, in an offline step, we first generate a navigation graph using the RB-PRM algorithm, presented in Chapter 6. At runtime, a request is performed to find a collision-free trajectory between two configurations given as input. The path returned by RB-PRM is then transformed into a sequence of discrete contacts, according to the method described in Chapter 7. For these configurations, we only provide the position and orientation of the root, and let the planner compute the appropriate contacts. Therefore all the contact configurations shown in the Figure of this section are computed by our motion planner.

8.2.1 Implementation details

The test application was developed using the C++ language. The motion capture data used to compute the Range Of Motions was processed using the Matlab framework. The polytopes describing the Range Of Motions are computed using the Matlab mpt toolbox [1]. They are exported to the obj format using a custom Matlab script. The dynamic balance criteria was implemented using the ccd polytope library [2].

The virtual creatures are described using the urdf file format [3]. It is used as a standard by the popular ROS platform, in which the optimization framework ITOMP

[1] http://people.ee.ethz.ch/m̃pt/3/

[2] http://www.inf.ethz.ch/personal/fukudak/cdd_home/cdd.html

[3] http://wiki.ros.org/urdf

is integrated. This choice was therefore made to unify the character description between our motion planner and ITOMP.

The scenarios where described using a custom text file format. Environments are described in the obj format. Rendering is achieved using Blender [4].

The runs were performed on a laptop with an Intel Core i7-2760QM 2.40GHz processor and 4 GB of memory. The application is not multi-threaded.

8.2.2 Test scenarios

All the scenarios presented in this section were tested with a humanoid virtual character. In the figures, the black arrow represents the local motion task considered for contact creation. The first and last figures represent the start and goal positions inputed to the planner.

Truck egress (Figure 8.8 and Figure 8.9). We consider the truck presented in Figure 8.8 – left. We work with a simplified geometry of this high resolution model (Figure 8.8 – right), and we consider a truck egress task. In the scenario we consider that the doors are blocked, so that the character has to crawl from the front window to leave the truck. Figure 8.9 presents the sequence of contacts obtained.

Figure 8.8: Left, middle: original high resolution truck model. Right: the simplified model used in our scenario.

Climbing (Figure 8.10). We consider a climbing wall with several grasps disposed along it. The virtual character is given the task of climbing along the wall. Figure 8.10 presents the contact sequence obtained for the task. The wall is not represented on the Figure for clarity reasons.

Crouching (Figure 8.11). We consider an abstract obstacle in the environment, floating in the air. A goal position on the other side of the obstacle is specified for the character, forcing him to crouch to cross the obstacle. Figure 8.11 presents the contact

[4] http://www.blender.org

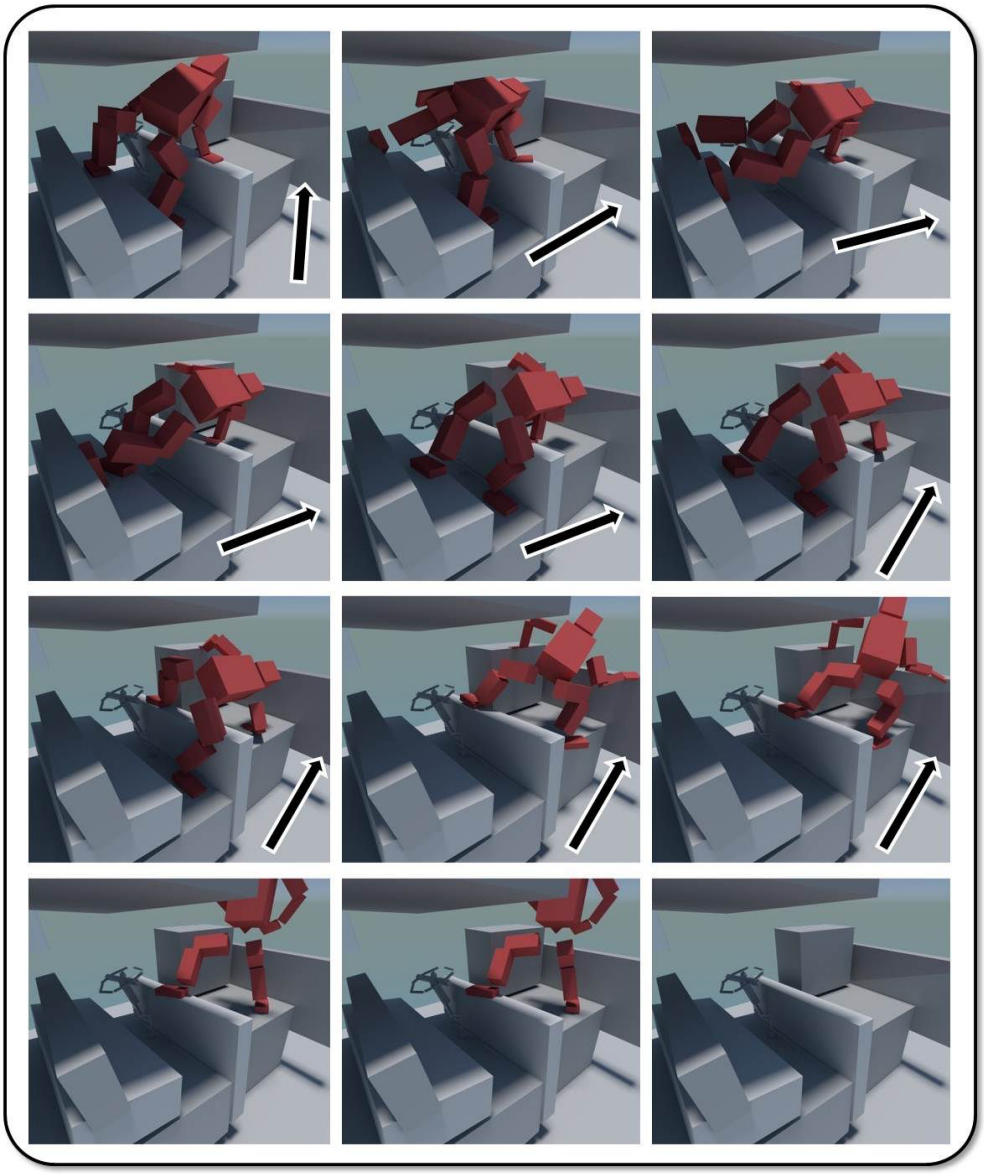

Figure 8.9: The computed contact sequence for the truck egress scenario.

sequence obtained for the task. Because the computed sequence of contacts is rather long, only selected stances are presented in the Figure. We can see that the character starts by taking a step back to go down, before crawling forward.

8.2.3 Performance analysis

The number of samples used for generating the contacts of each limb is 10000. Table 8.4 presents the average time (in seconds) spent in the different phases of the planner, for

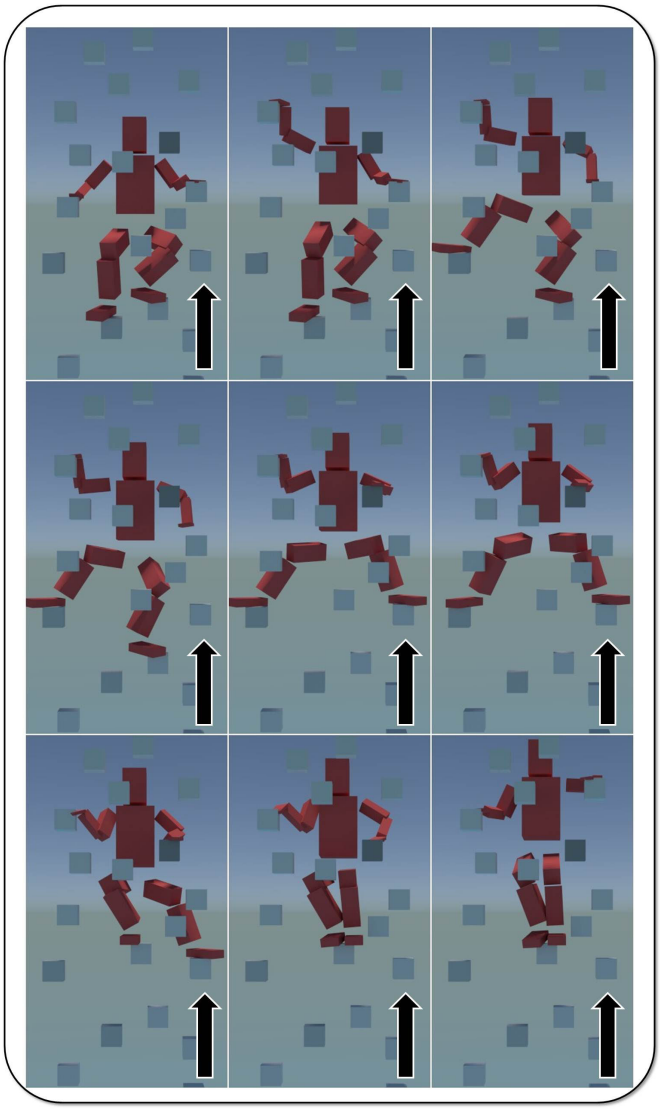

Figure 8.10: The computed sequence for the climbing scenario.

each phase and each scenario. For the online request phase, we distinguish: the time spent refining the path computed by RB-PRM, using the spline shortcut algorithm presented in Chapter 6; the time spent generating the contact sequence. The time spent for the actual path request of RB-PRM is not indicated; it is always inferior to 15 ms.

The time spent generating the navigation graph can be approximated to one minute. In the climbing and crouching scenarios, it is much shorter than in the truck egress scenario, because the topology of the environment is easier to capture.

In the request phase, the use of the spline shortcut algorithm to refine the trajectory is rather time consuming, especially, again, in the truck egress case. The reason for this

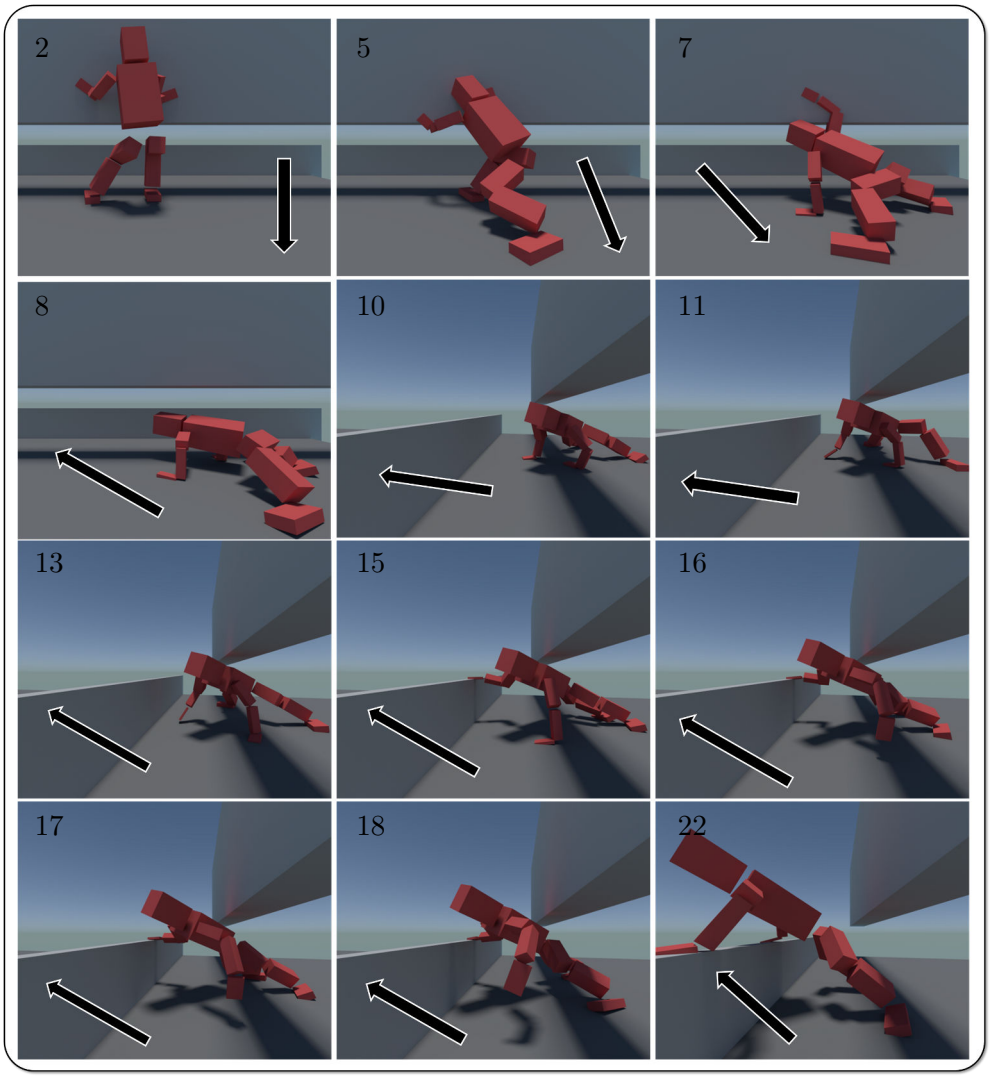

Figure 8.11: The computed sequence for the crouching scenario. The number displayed in each figure corresponds to the order of the configuration in the contact sequence.

is that many collision requests have to be performed to try to simplify the trajectory.

Finally, we observe that generating the contact sequence takes several seconds. This can be surprising, considering that this step is based on our real time contact generator. Our current implementation of the dynamic balance criteria explains the important time spent in this phase. The generation of the contact sequences takes the most time in the crouching scenario. This is simply explained by the fact that more states have to be created to achieve the motion.

	Generate RB-PRM (offline)	Path refinement	Generating the contact sequence	total online time
Truck egress	73	4	10	14
Climbing	2	< 0.1	1	3
Crouching	5	1	12	13

Table 8.4: **Average time** (in seconds) spent in RB-PRM generation, refinement of the trajectory, and generation of the contact sequence.

8.3 Discussion

In this chapter we present the results obtained in this thesis. First, we show examples of generated configurations in a real time, reactive context. Then we show examples produced by the second stage of our planner.

In both cases, the environment is constrained, and the objective was to generate task efficient contact configurations. In the first case, we generate only one contact configuration, and the motion task is manually given by the user. The constraint is to generate the configuration as fast as possible. In the second case, although it is considered, performance is less critical. However, this time the user only specifies a start and goal position, and we use our motion planner to compute a sequence of task efficient contact configurations.

Therefore we consider three ways of evaluating the results: the ability to compute a solution, the performance, and the quality of the motion.

Completeness of the method The contact generator we use prevents our methods from being probabilistically complete. Indeed, the method generates N samples offline used as inputs for the contact queries. This restricts the contact possibilities to the neighbourhood of these samples. It is possible to obtain probabilistic completeness if, instead of relying on a precomputed database of limb configurations, the sampling is performed online. However, from a practical point of view, performance seems a more interesting property than completeness for computer animation. Indeed, we show in this chapter that, when N is high, solutions can be found for a large variety of scenarios.

Performance The performance analysis presented in section 8.1.4 demonstrates that our contact generator is compatible with real time applications. In this regard, we can state that our objective is achieved.

Our motion planner, on the other hand, does not present this property. However compared to similar approaches for computing guide trajectories [BELK09], the planner is faster by several orders of magnitude, although it does not provide the same guarantees of dynamic balance.

Quality of the results In the absence of other elaborated heuristics for task efficiency, we compare our results with a simple heuristic which selects the closest contact available for contact creation. To evaluate the quality of the results produced by our method, it would be necessary to conduct a user perception study. Motion capture data would be compared against our method and other heuristics to determine the influence of the EFORT heuristic on the plausibility of the motion. This study will be conducted in future research.

One noticeable characteristic of our results is that in some cases, it appears that the number of contacts stances to perform is high. This can be seen for example in the crouching scenario (Figure 8.11). This illustrates a limitation of our approach: since we only consider contacts for the end effectors (even in situations where it would be more comfortable to use the knees or elbows), the created contacts can sometimes not be held for long. Future work will consider implementing our method for other possible contact locations. While applying our method to additional discrete points (such as knees or elbows) is straightforward, considering contact creation with arbitrary parts of the body is more challenging.

However, it appears that balance is the main limitation of our framework: it is not considered for real time applications, and despite a dedicated heuristic proposed with our motion planner, it cannot be ensured. Our method remains a kinematic motion planner.

This is the reason why in the third stage of our planner, the trajectories presented in section 8.2 must be optimized.

8.4 Discussion on ITOMP integration

We recall that the trajectories presented here are not meant to be directly used as a solution motion; rather, their purpose is to be used as a guide trajectory for the ITOMP optimization framework, presented in Chapter 7.

The motivation for providing a relevant guide trajectory is that optimization methods often fail to converge in constrained environments, because of their complexity. At the time of writing this thesis, we only have partial results with a not fully optimal set of optimization parameters for ITOMP.

ITOMP is based on the Contact Invariant Optimization formulation proposed by Mordatch et al. in [MTP12]. In this model, a cost function C_{CIO} guides the end effectors towards the nearest obstacles to create contacts. This is problematic in constrained environments, because the nearest obstacle is not necessarily the most relevant. In theory, this issue should be solved thanks to another cost function $C_{physics}$ which enforces the validation of the law of physics. However, the CIO approach simplifies the contacts formulation: for instance to generate a climbing motion, hands and feet are allowed to exert infinite forces. This results in the effectors being guided towards irrelevant contact configurations, and diverging from the guide trajectory.

To address this issue we first propose a more accurate formulation of the contact

location. Furthermore we integrate the EFORT heuristic as a cost function to enforce the task efficiency property of the selected contacts.

Another issue that we are facing is that in constrained environments the number of obstacles impacts the performance of the method, because more collision checks must be performed at each step to reduce the value of the collision cost function $C_{Collision}$. Additionally, the importance of $C_{Collision}$ is greater in this context and increasing its weights affects the importance of the other costs functions.

An optimization framework for virtual creatures comprising a high number of degrees of freedom is a complex system with many parameters. We must adapt them to find the good compromise between respecting the guide trajectory and producing balanced trajectories.

Chapter 9

Conclusion

This thesis was set out to explore new means of providing virtual characters and robots with the ability to automatically synthesize motions in constrained environments. Precisely, it focuses on tasks characterized by the important force exertion they require and the non stereotypical sequence of contacts needed to achieve them, such as climbing, standing up, pushing an object... Providing entities with an enhanced autonomy of motion in constrained environments has several applications, from the deployment of robots in search and rescue missions, to the improvement of the immersion experience of a video game player.

The study was driven by two potential applications: firstly, real time motion synthesis in interactive applications such as video games, where the motion must be computed within milliseconds; secondly, global trajectory planning for simulations, which focus on computing a feasible and natural looking solution, and in which case the performance constraints are alleviated.

The existing methods and models were found to be inconclusive on this topic: robotics motion planners are generic approaches which can compute trajectories in constrained environments, but produce unnatural motions. Conversely, example based approaches generate natural looking motions but lack genericity, and often fail in this context. The study has sought to determine whether new heuristics could be used along with motion planning methods to address this issue.

This thesis lies in the continuity of robotics approaches, which formulate the problem as the research of a trajectory decomposed in an ordered sequence of contacts configurations. Specifically, two research questions were addressed:

1. How to rapidly generate a contact configuration compatible with a force exertion task in an unknown environment?

2. How to compute relevant contact trajectories for a force exertion task in a constrained environment?

9.1 Findings and contributions

The two main contributions of this thesis are chapter specific and were summarized and discussed within the respective chapters:

- Our first contribution was presented in Chapter 5: A heuristic for task efficient contact configurations: the Extended FORce Transmission ratio (EFORT);

- Our second contribution was presented along Chapter 6: A Reachability Based Probabilistic Road Map: RB-PRM, and Chapter 7: Stage 2 and 3: generation of a task efficient trajectory.

In this section we synthesize the contributions to answer the two research questions.

9.1.1 How to generate rapidly a contact configuration compatible with a force exertion task in an unknown environment?

By proposing EFORT, a simple and generic measure for task efficiency. Addressing the first research question required defining what is a compatible configuration and providing a mean to evaluate the said compatibility. The Extended FORce Transmission ratio addresses this issue. It is a tool which evaluates how easily variations of the joint values of a kinematic chain are transformed into a force exerted against a surface in a given direction. We chose it consequently as a measure of the potential force that can be exerted in a direction.

Thanks to those properties, EFORT can be used to compare different candidate configurations for a given task. It can also be used as a secondary objective of an inverse kinematics solver to optimize a contact configuration.

Furthermore, EFORT is fast to compute and generic, since it can be applied to any kinematic chain.

By proposing an automatic real time generator for task efficient configurations. An environment independent sampling based approach, combined with the EFORT heuristic has proven a relevant way to generate contact configurations in constrained environments. When no assumption on possible contact locations can be made, sampling the reachable workspace of the considered limb allows to generate rapidly a large set of possible configurations in contact with the environment. The candidate which maximizes the EFORT heuristic is then returned as the most compatible configuration found.

The combination of the EFORT heuristic and our contact generator allowed us to propose the first automatic real time generator for task efficient contact configurations for arbitrary creatures and environments.

9.1.2 How to compute relevant contact trajectories for a force exertion task in a constrained environment?

By proposing the Reachability-Based PRM (RB-PRM), a new way to explore the high dimensional configuration space. The study has determined that generating relevant contact configurations requires having knowledge of the trajectory that must be followed. Ideally the trajectory and the contact configurations should be computed simultaneously, but this objective remains yet to be achieved in constrained environments. To address the delicate issue of computing a relevant trajectory without directly considering contact generation, the reachability condition was proposed. Thanks to this heuristic the trajectories computed by RB-PRM are close enough to obstacles to allow relevant contact creation in a ulterior step.

By computing guide trajectories for optimization frameworks. Combining RB-PRM and a task efficient contact generator makes it possible to compute a sequence of relevant contact configurations achieving a motion task in a constrained environments. The contact sequence is of a significant help for classical optimization frameworks, which face strong convergence issues in our context. The combination of these three bricks will result in the first automatic motion planning solution able to produce task efficient, dynamically balanced trajectories in constrained environments.

9.2 Findings implications

In this thesis we propose a motion planner which combines task efficiency, robustness to constrained environments and acyclic motion planning. This unprecedented combination of properties allow us to synthesize relevant motions for our objective examples, characterized by constrained environments, requirement for important force exertion and acyclic contact generation.

Addressing automatically these examples is a significant step towards the long term goal of providing virtual characters with a complete autonomy of motion.

In the meantime, interactive applications such as video games can already benefit from our contributions. We propose new procedural means to produce task efficient motions, not as an alternative, but as a complement to example based approaches such as motion capture. The additional motion capabilities brought to virtual characters alleviate the constraints on the design of virtual environments, making them less stereotypical. This results in an improved player experience.

The automatic generation of contact configurations could also find a useful application within a 3D modeling software: providing an animator with contact configurations instead of letting him manually create the postures could assist him in the animation process.

9.3 Limitations of the study

The contributions of this thesis are organized around the proposal of simple kinematic heuristics to reduce the complexity of searching feasible trajectories in high dimensional spaces. As a direct consequence of this approach, they face a number of limitations which need to be considered.

A first limitation concerns the global optimality of the computed contact configurations. Indeed, our motion planner provides a guarantee that a generated contact is the most appropriate *locally*, but does not propose any means to verify that in the long run the decision is the best. Similarly, the decision of creating or removing a contact is based on really simple heuristics, which may not lead to the most efficient solution. Formulating new criteria for optimality objectives (such as "minimizing the number of contacts created along the trajectory") and proposing a method to achieve them would allow to overcome this limitation and improve the quality of the results.

A related limitation of our reachability based approach lies in the fact that dynamic balance is only addressed once the trajectory has been computed. In its actual state, our motion planner cannot provide absolute guarantee that a balanced solution can be found along the trajectory. This is not an issue for computer graphics animations. But before being able to implement our solution with actual robots, our motion planner will have to be extended to provide a guarantee of stability.

9.4 Recommendation for future research

Indeed, we believe our contributions might have applications to the robotics field, although this thesis focuses primarily on computer graphics applications. Future research will consider implementing and testing our motion planning methods with humanoid robots in real world experiments. To achieve this and improve our contributions, several other research questions will be considered in the near future. They all revolve around the ideas developed in this thesis: to propose and validate simple heuristics to simplify the search for relevant motion trajectories without losing genericity.

Towards real time contact trajectory generation. The heuristics proposed in this thesis are really fast to compute, and allow the computation of a rough trajectory in a few seconds. However, to produce a dynamically balanced trajectory, in the last stage of our framework, we use a slow optimization method. In future work, we will try to propose simple heuristics to generate dynamically balanced configurations more efficiently. Some promising work on stability margin has been proposed by Qiu et al. in [QEMR11] and used in this thesis. We believe it is a good basis for prospective work on the possibility to simplify the dynamic optimisation step.

Towards real time contact decision. An exciting and challenging long term objective is the design of an "acyclic pattern generator" which would allow to generate relevant contacts with appropriate effectors given an arbitrary configuration. Addi-

tional work is required to understand the laws that determine when to create or break contacts.

Additional heuristics for task efficiency. This thesis addresses task efficiency as the ability to exert important forces in a given direction. While this formulation is adapted to a large set of tasks, there are many more for which it does not apply. In the future we will work on proposing new heuristics allowing to perform other categories of tasks, such as writing. We will also consider other heuristics which can be applied simultaneously to EFORT.

Validation of EFORT through perception studies. Finally, we would like to experimentally validate the hypothesis that using EFORT allows to produce more plausible motions. To do so we intend to conduct a perception study to determine how the motions synthesized with EFORT are perceived by human observers. This study also be completed by a biomechanical analysis based on real human data.

9.5 A final word

As of today, the reasons why we move the way we move in constrained environments cannot be explained simply. This thesis claims that, above the complexity of balance constraints, muscular models, collision avoidance, there are simple rules which dictate how we move in this situation. Knowing them would not remove the necessity for the complex optimization of trajectories, but it would be of precious help to guide such motion planning techniques towards optimal solutions. Our future research will be focused on trying to develop synergies between the fields of computer animation, robotics and biomechanics to understand what makes a motion natural.

Chapter 10

Resumé long de la thèse en francais

10.1 Introduction

Avec la complexité croissante des environnments virtuels apparaît le besoin de doter les personnages virtuels d'une plus grande autonomie de mouvement. En plus de marcher, courir ou de sauter, les dernières simulation requièrent des personnage qu'ils rampent, escaladent, poussent ou tirent des objets... Ainsi dans le jeu vidéo Assassin's creed TM, le personnage contrôlé par le joueur peut grimper aux murs d'une apparente variété de batiments, ou se déplacer dans les hauteurs d'une forêt aux arbres géants (Figure 10.1).

.

Figure 10.1: Exemples de mouvements d'escalades et de franchissement d'arbres dans le jeu vidéo Assassin's creed. © Ubisoft.

En parallèle, les robots humanoïdes quittent les laboratoires de recherche oú ils ont été développés pour être déployés dans des situations complexes, comme dans des cénarios d'intervention sur des lieux sinistrés (Figure 10.2). Par exemple pendant le "DARPA challenge", des robots sont en compétition pour la réalisation de tâches complexes comme de retirer des débris ou bien encore de se déplacer sur des terrains inégaux.

Figure 10.2: The robots bipèdes Chimp et Schaft durant le "DARPA challenge". © DARPA challenge.

Dans ces deux contextes il apparait nécessaire de doter les protagonistes d'une autonomie de mouvement dans ces environnements contraints. C'est la problématique de cette thèse, qui se concentre sur les applications graphiques.

Motivation et objectifs

Parce qu'elles avaient des objectifs différents, les champs de la robotique et de l'animation graphique ont traité le problème de la planification de mouvement différement.

Dans le contexte interactif des jeux vidéos, le principal objectif est la performance: il faut à tout prix éviter la latence qui pourrait fruster le joueur; la simulation doit être réactive à ses actions. De plus, la qualité des animations est également très importante. Une animation peu naturelle aurait un impact dramatique sur l'expérience utilisateur.

Pour ces raisons, la plupart des techniques employées dans le domaine graphique sont dites "basées exemple". Ces techniques s'appuient sur le travail d'animateurs qui produisent à la main des animations, ou bien encore des méthodes de capture de mouvement, puis permettent de reproduire à l'identique les mouvements d'acteurs réels (Figure 10.3). L'avantage des méthodes basées exemple est qu'elles donnent à l'animateur un contrôle total sur les animations jouées, et donc sur leur qualité.

Cependant ces méthodes souffrent d'un défaut majeur: elles permettent très peu de modifications de l'animation de référence, ce qui implique que l'environnement où l'animation est jouée doit être strictement le même que celui dans lequel elle a été créée. Celà conduit à des environnements très stéréotypés: Dans le jeu Assassin's creed (Figure 10.1), on peut voir que les mouvements d'escalade sont exactement les mêmes

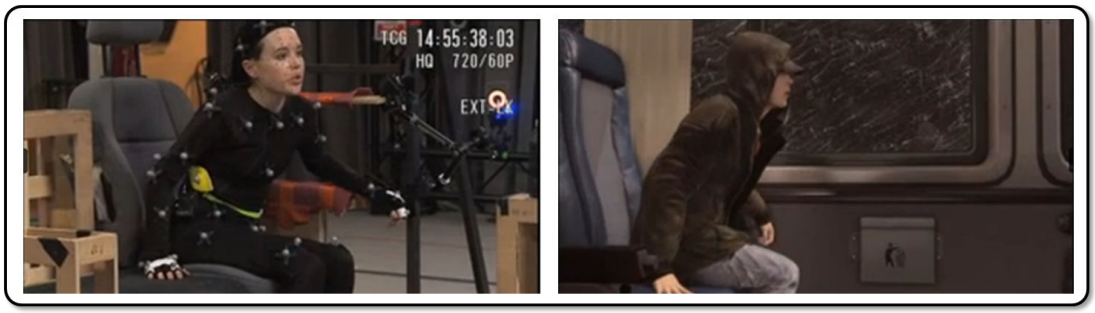

Figure 10.3: Les techniques de capture de mouvement permettent d'enregistrer et de rejouer les mouvements d'un acteur réel à l'identique. © Quantic dreams.

que l'on grimpe au mat d'un bateau ou sur le toit d'une maison, parce que les prises sont placées exactement à la même distance l'une de l'autre.

Motivation 1: Donner aux personnages virtuels une autonomie de mouvement accrue permettrait de donner plus de liberté dans la conception d'environnements virtuels, et améliorerait l'expérience de jeu du joueur de jeu vidéo. Une solution valide permettrait d'obtenir des résultats en temps réel, et de qualité suffisante.

A l'inverse, en robotique l'objectif principal est de permettre aux robots de se déplacer dans des environnements complexes et inconnus (on peut par exemple penser au robot Curiosity qui explore la surface de Mars). Il s'agit donc de leur donner des capacités de raisonnement et de planification de mouvement avant tout. Peu importe la qualité du mouvement obtenu pourvu que l'objectif soit atteint !

La plupart des contributions en robotique présentent donc des planificateurs de mouvement [KSLO96, LaV98, HBL05, BRL$^+$04, EKMG08]. Ces planificateurs sont robustes et peuvent trouver des solutions dans des environnements très complexes, cependant les mouvement résultant des trajectoires calculées apparaissent en général peu naturelles à un observateur humain.

Pour pouvoir utiliser ces méthodes en animation graphiques, certains travaux visent à générer des trajectoires moins chaotiques en utilisant des méthodes d'optimisation [MTP12]. Cependant, les problèmes de plainification de mouvement sont très complexes et d'une très grande dimensionalité; quand l'environnement devient trop contraint, les méthodes d'optimisation classiques échouent.

Motivation 2: Doter les planificateurs de mouvement robotiques d'heuristiques permettant de réduire la complexité du problème de recherche de chemin permettrait d'améliorer leur performance et la qualité des résultats qu'ils proposent.

Exemples - objectifs: Nous considérons deux types d'application que les techniques actuelles ne permettent pas de réaliser complètement. D'un côté nous considérons le

problème de générer en temps réels des contacts pertinents pour un personnage virtuel (Figure 10.4); d'un autre côté nous considérons le problème global de planification de mouvement en environnements contraints (Figure 10.5, Figure 10.6 – Left).

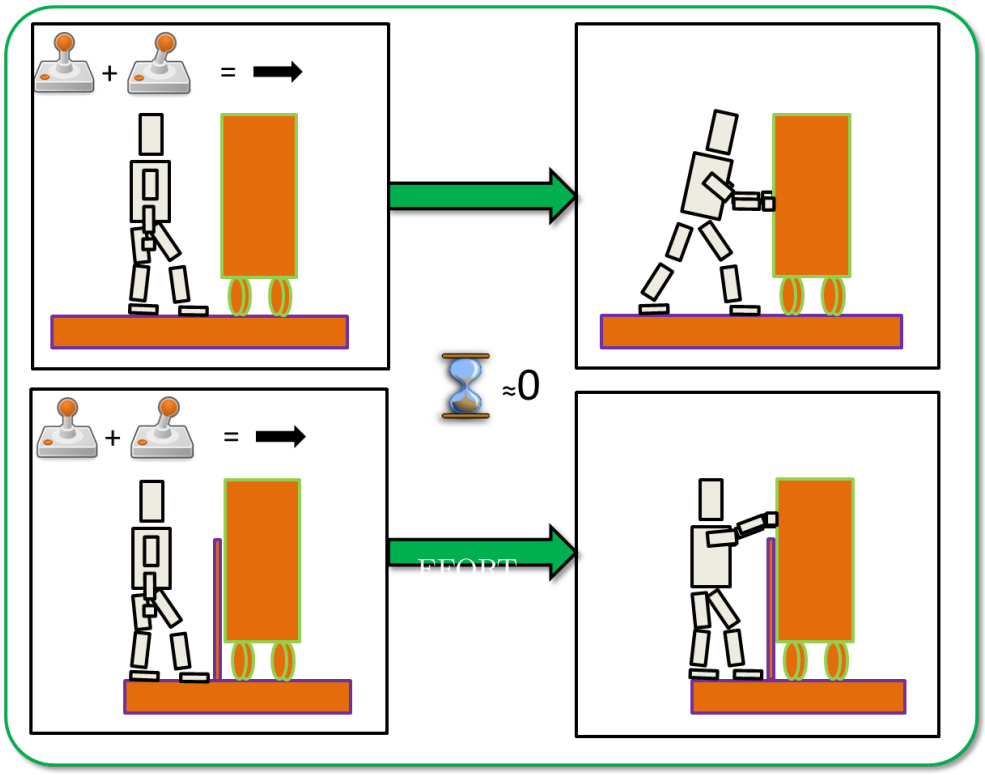

Figure 10.4: Notre première contribution consiste en un générateur de postures de contact pour des environnements contraints.

Les exemples illustrés par ces figures sont difficiles parce qu'on ne peut pas utiliser d'approches déterministes pour trouver une solution. Selon l'environnement, les contacts nécessaires pour réaliser le mouvement vont varier énormément.

Le problème commun à ces exemples est celui de la génération d'une séquence de contacts pertinente pour réaliser la tâche, et c'est celui auquel nous nous intéressons dans cette thèse.

Contribution

Nous proposons une méthode pour réaliser nos exemples cibles.

D'abord, nous proposons un générateur de configurations de contacts en temps réel pour répondre à notre motivation 1. Il est basé sur une heuristique appelé l'Extended FORce Transmission ratio, our EFORT.

Deuxièmement, nous proposons d'utiliser ce générateur au sein d'une architecture de planification de mouvement complète, en 3 étapes (Figure 10.6).

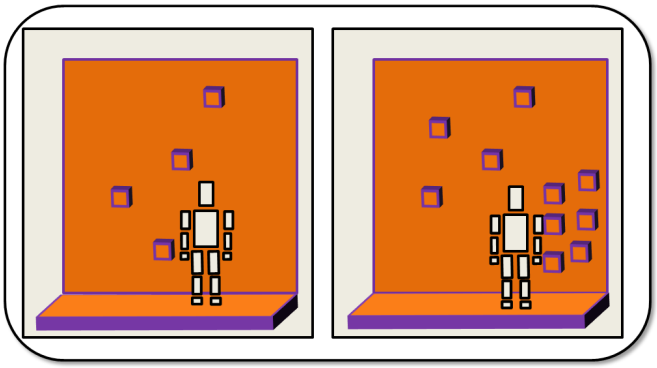

Figure 10.5: Ce scénario d'escalade est problématique pour les méthodes existantes.

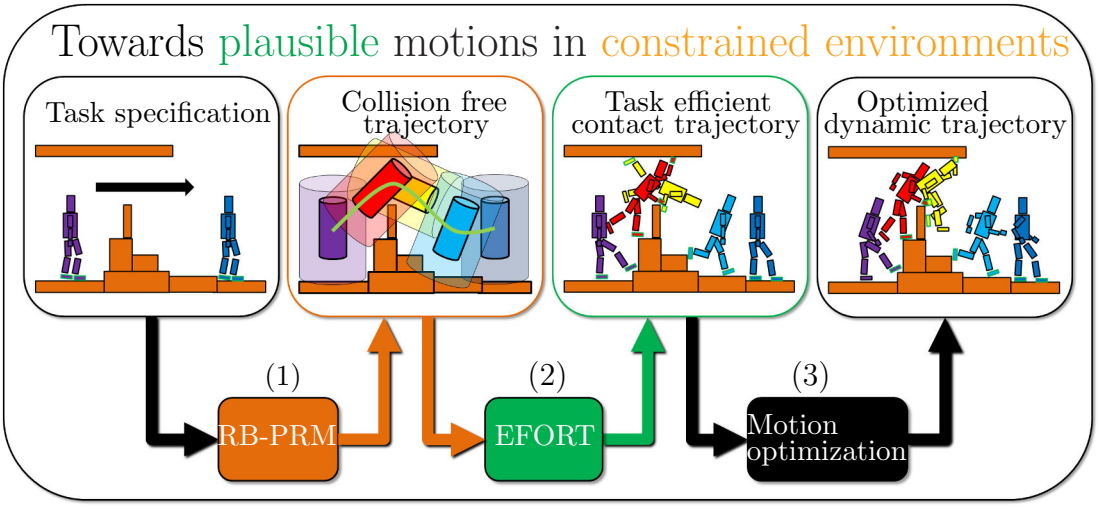

Figure 10.6: Notre solution s'articule en 2 étapes pour déterminer une trajectoire complète en environnements contraints.

Notre méthode génère en quelques secondes une trajectoire guide pour une méthode d'optimisation, qui vise à réduire l'espace de recherche et facilite la convergence de l'optimisation. Our second step is the design of a global motion planner built upon this contact generator (Figure 10.6).

Nous proposons donc 2 contributions:

Contribution 1: Avec EFORT nous proposons une méthode temps réel pour la generation de configurations de contact, ce qui a pour effet d'améliorer l autonomie de mouvement des personnages virtuels.

Contribution 2: Nous calculons des trajectoires guide pour un problème d'optimisation. Ceci nous permet de trouver des solutions dans des environnements très contraints, pour lesquels les méthodes existantes ne trouvaient pas de solutions (Figure 10.6).

plan du résumé

le reste du résumé de la thèse s'organiser ainsi:

- la section 10.2 présente rapidement les définitions essentielles à la compréhension des points techniques de la thèse;
- la section 10.3 présente la méthode EFORT pour la génération temps réel de contacts pertinents pour une tache;
- la section 10.4 présente la première étape de notre planificateur global de mouvement, RB-PRM;
- la section 10.5 présente les 2 et 3eme étape de notre planificateur de mouvement.
- la section 10.6 présente les résultats obtenus;
- la section 10.7 conclue le résumé.

10.2 Définitions

10.2.1 Environnement.

Un obstacle $O \in W$ est un ensemble de triangles en 3 dimensions. Un triangle de O est écrit t_O.

10.2.2 Définition et représentation d'un personnage virtuel

10.2.2.1 Squelette

Un personnage virtuel est défini par une chaine cinématique R. R comprend l sous chaines $R^k, 0 \leq k < l$. Chaque articulation possède au plus un fils. une articulation sans fils est un effecteur (Figure 10.7).

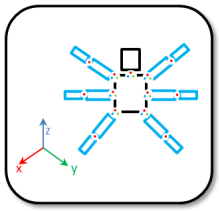

Figure 10.7: Un insecte virtuel avec 6 membres et 33 degrés de liberté.

10.2.2.2 Configuration

Un personnage possède $n \geq 6$ articulations. 6 degrés de liberté sont utilisés pour décrire la position et l'orientation de le racine de R dans W.

Les définitions suivantes sont illustrées par la Figure 10.8. Une configuration $\mathbf{q}$ est vecteur de dimension n qui décrit les valeurs de chaque degré de liberté de R. $\mathbf{q}^k, 0 \leq k < l$ décrit la configuration d'un membre R^k.

$\mathbf{p}_{\mathbf{q}^k}$ décrit la position dans le monde de l'effecteur associé au membre R^k, étant donnée la configuration $\mathbf{q}$.

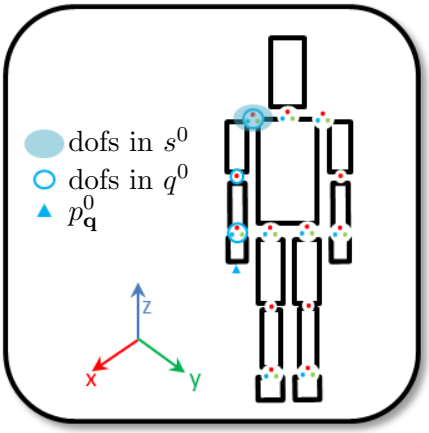

Figure 10.8: Humain virtuel au repos.

10.2.2.3 Abstraction d'un personnage virtuel

Nous définissons l'abstraction A d'un personnage virtuel R:

$$A = A_{trunk} \cup A_{ROM} \tag{10.1}$$

où A_{trunk} and A_{ROM} sont 2 ensembles d'objets 3d. A_{trunk} représente le tronc du personnage (Figure 10.9 - Red cylinder). A_{ROM} représente la zone d'atteignabilité de chaque membre $R^k, 0 \leq k \leq l$.

A_{ROM} est représenté par les quatre ellipsoides verts de (Figure 10.9).

10.3 EFORT

Considérant nos exemples cibles (Figure 10.10), nous nous posons la question suivante: Etant donnée une tâche qui nécessite d'exercer une force importante dans un environnment contraint, et la configuration actuelle d'un personnage virtuel dans l'environnement, comment créer le prochain contact pour qu'il permette de réaliser la tâche efficacement?

Répondre à cette question est essentiel car c est un premier pas vers la résolution d'un problème de planification globale (Figure 10.6 - 2).

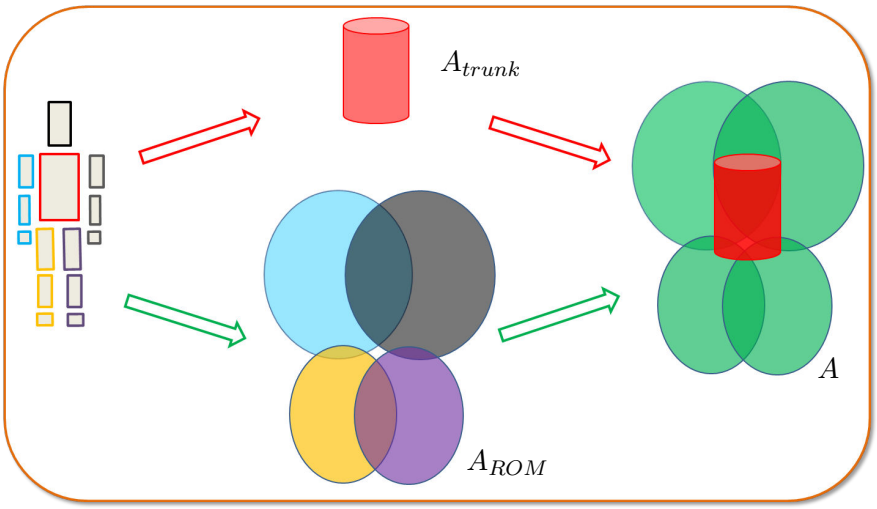

Figure 10.9: Abstraction d'un humain virtuel.

Premièrement, nous commençons par définir ce que nous entendons par "efficacement", et proposons une heuristique mathématique pour évaluer cette efficacité: EFORT.

Ensuite, nous proposons une méthode qui permet de générer des configurations candidates et de les comparer au moyen de cette heuristique afin de choisir la meilleure, tout celà en temps réel.

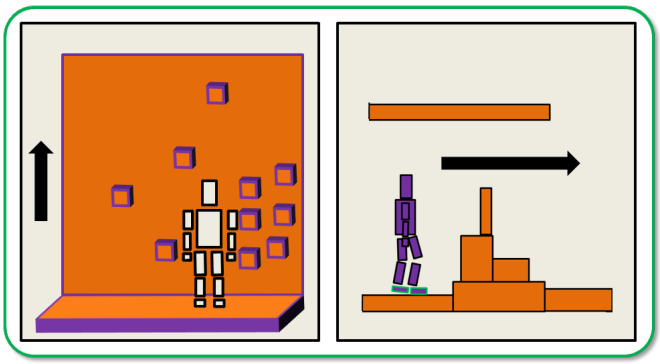

Figure 10.10: Quand et comment créer le prochain contact pour satisfaire la tâche?

10.3.1 EFORT: une nouvelle heuristique pour évaluer la compatibilité d'une configuration.

Etant donné l'environnement courant et un ensemble de configurations candidates, EFORT retourne la configuration la plus appropriée pour réaliser une tâche donnée.

Les tâches que nous considérons nécessitent d'appliquer une force importante Figure 10.10); pour cette raison nous définissons la compatibilité d'une configuration

comme la mesure de la capacité d'une configuration à appliquer une force dans une direction donnée.

Pour celà, nous nous basons sur le "force transmission ratio" proposé par Chiu dans [Chi87]. Il est classiquement utilisé pour optimiser une configuration étant donné un point de contact déjà connu [GKNK06]. Seulement dans notre cas, nous ne connaissons pas à l'avance avec quelle surface nous désirons créer un contact (Figure 10.10). Chaque surface étant un candidat potentiel, nous étendons le force transmission ratio pour proposer un moyen de choisir la surface la plus pertinente.

Considérant un membre R_k d'un personnage virtuel R, auquel est donné une tâche d'actuation de force dans une direction donnée par le vecteur $\mathbf{v_t}$; considérant de plus un ensemble $Q^k \in C^k$ de configurations possibles, en contact avec différentes surface; Nous nous posons la question suivante: Quelle est la configuration la plus appropriée pour réaliser $\mathbf{v_t}$?

Etant donnée la formule du "force transmission ratio" f_T:

$$f_\mathsf{T}(\mathbf{q}^k, \mathbf{v_t}) = f_\mathsf{T}(\mathbf{q}^k, -\mathbf{v_t}) = [\mathbf{v_t}^T (J(\mathbf{q}^k) J(\mathbf{q}^k)^T) \mathbf{v_t}]^{-\frac{1}{2}} \tag{10.2}$$

nous considérons chaque candidat et la surface O_i. avec laquelle il est en contact. La formule de EFORT est donnée par la fonction α:

$$\alpha(\mathbf{q}^k, \mathbf{v_t}) = f_\mathsf{T}(\mathbf{q}^k, \mathbf{v_t}) \mathbf{v_t} \cdot \mathbf{n_{Oi}} \tag{10.3}$$

où $\mathbf{n_{Oi}}$ est la normale à O_i.

10.3.2 Génération de contact en temps réel

Nous intégrons EFORT dans une méthode pour générer des configurations de contacts en temps réel. Pour des raisons d'efficacité la méthode est décomposée en 2 étapes; une hors ligne et une en ligne.

Echantillonnage hors ligne. Indépendamment de l'environnement, un large ensemble Q^k de configurations est généré aléatoirement pour chaque membre du personnage virtuel; ces configurations respectent les butées articulaires définies par l'utilisateur pour chaque articulation (Figure 10.11).

Requête en ligne. Lorsque la simulation est lancée, une requête est lancée sur l'ensemble Q^k. Les configurations présentement en contact avec l'environnement sont retenues en tant que candidates potentielles. De toutes ces configurations, nous choisissons celle qui maximise EFORT (Figure 10.12).

10.4 Etape une de notre planificateur: RB-PRM

Nous présentons la première étape de notre planificateur (Figure 10.6 - 1). Il s'agit d'une variation des planificateurs de mouvement de la famille des PRMs. Nous nous basons

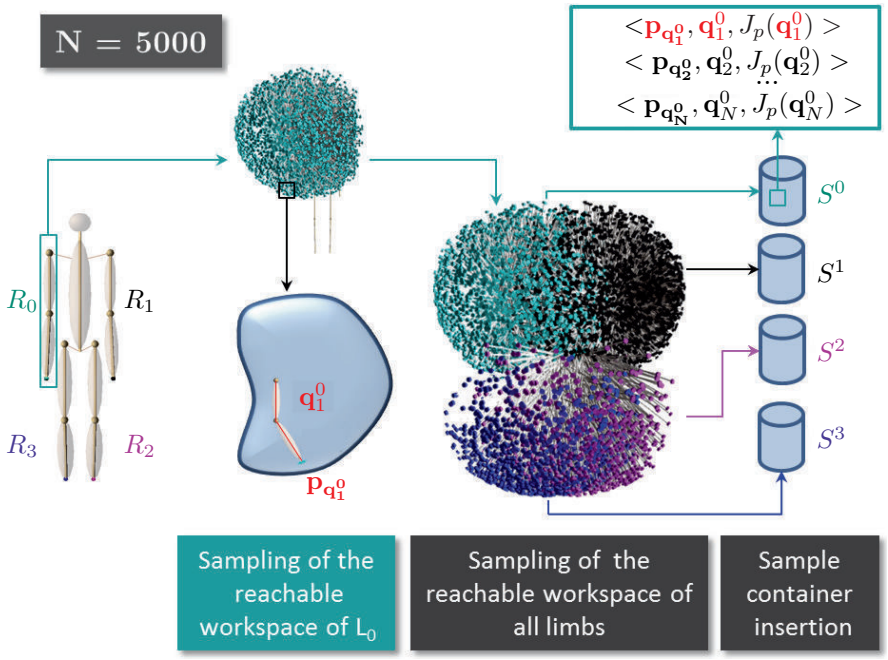

Figure 10.11: Illustration de la génération hors ligne de configurations.

sur l'abstraction d'un personnage virtuel pour définir une condition d'atteignabilité, illustrée par la Figure 10.13:

$$C_{reachability} = \{(\mathbf{q} : A^{\mathbf{q}}_{ROM} \cap W \neq \emptyset \wedge A^{\mathbf{q}}_{trunk} \cap W = \emptyset\} \tag{10.4}$$

Une configuration qui remplit cette condition est intéressante car elle va potentiellement permettre de générer des contacts, puisqu'au moins un obstacle se retrouve dans la zone d'atteignabilité.

Nous nous basons sur un algorithme classique de planification de mouvement, le visibility-PRM, pour générer un graphe de navigation. Cet algorithme est modifié pour que les configurations générées vérifient la condition d'atteignabilité, permettant ainsi de calculer des chemins proches des obstacles.

En sortie de l'étape une, nous obtenons donc un chemin sans collision permettant de relier deux configurations dans l'espace. A ce stade, la création de contacts n'est pas directement considérée, mais les configurations retournées sont suffisament proches des obstacles pour permettre leur création.

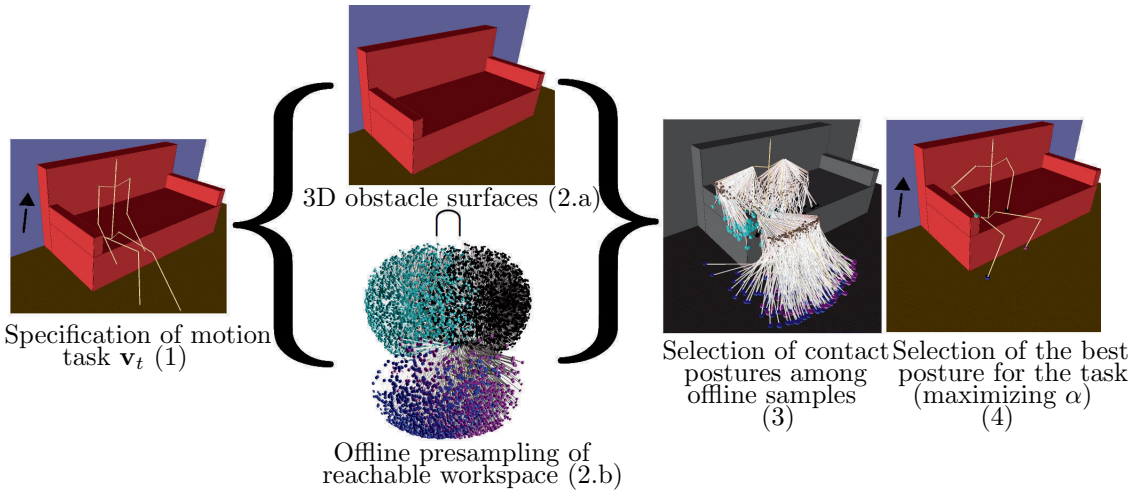

Figure 10.12: Illustration de la requête en ligne d'un contact pertinent.

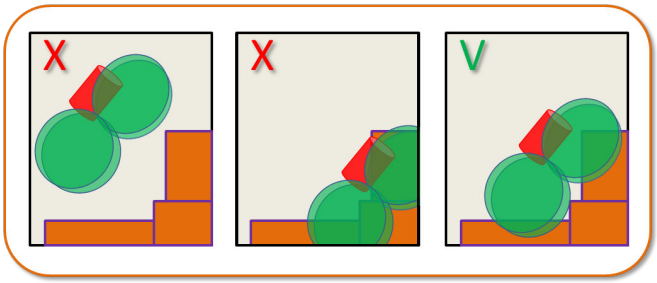

Figure 10.13: Illustration de la condition d'atteignabilité: (A_{trunk}) est sans collision et (A_{ROM}) est en collision

10.5 Etape 2 et 3: génération d'une trajectoire de contacts

10.5.1 Etape 2: génération d'une séquence de contacts

A l'étape 1 de notre méthode nous avons généré une trajectoire sans collisions reliant une configuration de départ à une configuration d'arrivée. Nous transformons cette trajectoire en une séquence de contacts en utilisant le générateur de contacts précédemment proposé. De plus, nous ajoutons une condition de stabilité dynamique afin de générer des configurations plus pertinentes.

Pour celà on choisit un intervalle on considère les configurations de la trajectoire à intervalles réguliers et on crée ou rompt des contacts en fonction de la configuration précédente et des possibilités offertes par la nouvelle position. On essaie autant que possible de maintenir les contacts existants et on en génère de nouveaux dès que la condition d'équilibre dynamique est brisée.

Ceci est illustré par la Figure 10.14.

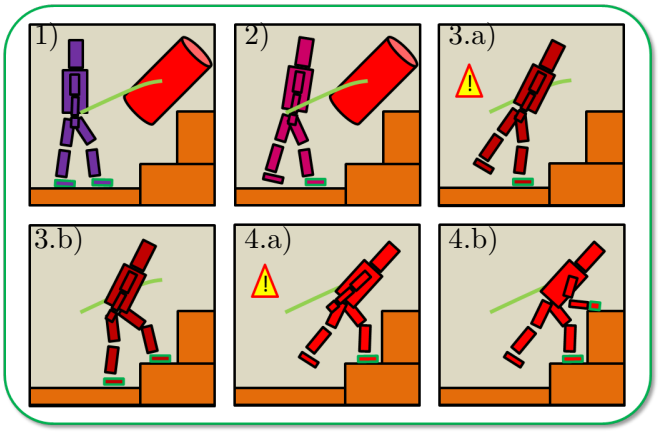

Figure 10.14: Illustration de l'étape 2

10.5.2 Etape 3: optimisation

La trajectoire générée à l'étape 2 va être optimisée dans une dernière étape. Elle est utilisée comme trajectoire guide pour la méthode d'optimisation ITOMP développée par Park et al. [PPM12]. Cette optimisation va servir à rendre la trajectoire dynamiquement stable, plus fluide, et par conséquent plus naturelle.

Le fait d'utiliser une trajectoire guide va faciliter grandement la convergence et permettre de trouver des résultats là où les méthodes précédentes échouaient.

10.6 Résultats

Dans cette section nous présentons les résultats obtenus au cours de cette thèse les résultats que nous avons obtenu en utilisant notre générateur de contacts dans une simulation interactive.

10.6.1 Scénarios de test

Dans cette section dans chaque scénario la tâche est indiquée avec un joystick, ou bien selon une trajectoire manuellement définie. On considère une créature dans un environnement virtuel. La position est l'orientation de la racine du personnage sont décrites par 6 coordonnées. Par défaut la posture de base du personnage est sa posture de référence.

On considère une tâche en effort donnée par un vecteur. Nous utilisons notre méthode pour calculer des configurations de contact compatibles avec la tâche.

Se lever –Figure 10.12 and Figure 10.15–. L'environnement se compose d'une chaise et d'une table. Le personnage est un humanoïde. Dans sa configuration initiale il est assis sur la chaise. La tâche de se lever est donnée par un vecteur vertical.

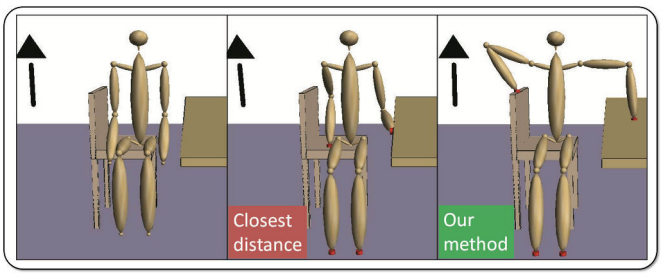

Figure 10.15: Comparaison de notre méthode (droite) avec de simples heurisques comme celle de choisir le contact le plus proche (gauche).

Créatures en environnements contraints. Cette fois l'environnement se compose d'un ensemble de livres placés sur une étagère. Un insecte virtuel avec 6 effecteurs traverse l'environnement sur ces livres. La tâche est donnée par une trajectoire qui pointe vers l'avant de la créature.

Tirer et pousser des objets –Figure 10.16 et Figure 10.17–. La créature est un humanoïde.

Deux environnements sont utilisés: pour le scénario qui consiste à pousser, l'environnement se compose d'une armoire et du sol; dans l'autre scénario, on considère également une corde attachée à l'armoire, ainsi qu'un muret près du personnage.

La tâche consiste à tirer (pousser) une armoire. Elle est formulée comme un vecteur horizontal. Nous calculons des configurations de contacts pour la jambe gauche et les bras.

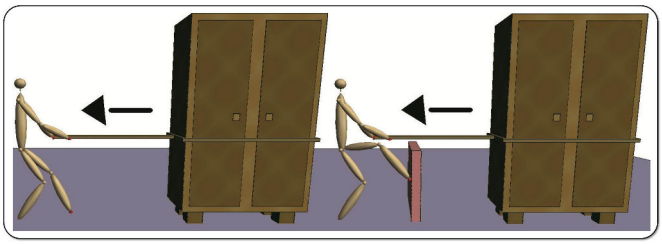

Figure 10.16: Configurations calculées par notre algorithme pour des tâches de poussées.

Calcul d'une séquence de configurations de contact –Figure 10.18. Un personnage virtuel est situé dans un environnement dans sa posture de référence. Etant données une trajectoire pour le centre géométrique, nous calculons une séquence de contacts le long de la trajectoire. La première configuration calculée est donnée en

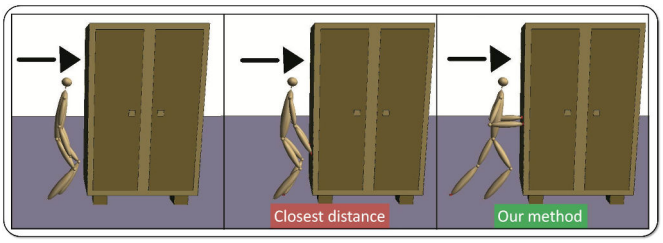

Figure 10.17: Notre méthode (droite) est comparée avec l'heuristique simple de point le plus proche (milieu) dans cet exemple de poussée d'armoire.

entrée à la suivante, et ainsi de suite.

Ces exemples démontrent l'intéractivité que notre méthode permet d'atteindre.

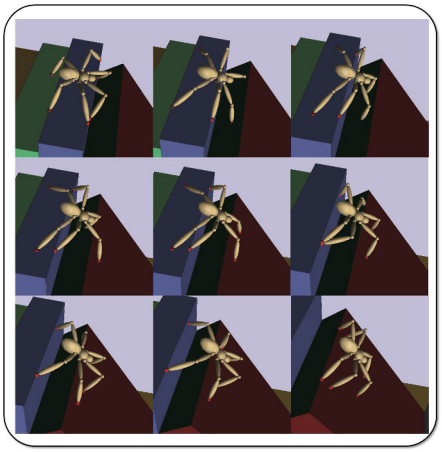

Figure 10.18: Séquence de configurations pour un insecte virtuel.

10.7 Conclusion

Le but de cette thèse est l'exploration de nouveaux moyens de doter des personnages virtuels et des robots d'une plus grande autonomie de mouvement, comprise comme la capacité à synthétiser automatiquement un mouvement adapté à une tâche donnée dans un environnement contraint. Parmi ces tâches on s'intéresse à l'escalade, à la poussée d'objets, où encore à des mouvements consistant à se relever d'une chaise. Ces tâches nécessitent de générer des efforts importants résultants de la création de nombreux contacts avec l'environnement.

Les applications sont multiples, depuis l'amélioration de l'expérience d'immersion du joueur de jeu vidéo jusqu'au déploiement de robots lors de manoeuvres d'assistance après catastrophe.

Cette thèse est une contribution à la question de recherche suivante: **Etant donné un personnage virtuel à la géométrie quelconque dans un environnement**

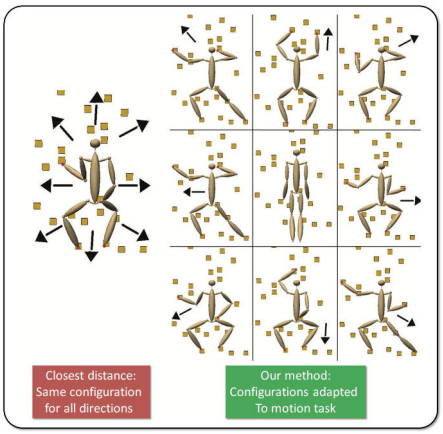

Figure 10.19: Configurations choisies pour un humanoïde pratiquant l'escalade.

quelconque, comment générer automatiquement un mouvement permettant de réaliser efficacement une tâche de locomotion nécessitant d'importants efforts?

La littérature existante sur le sujet ne permet pas de proposer une solution entièrement satisfaisante à cette question complexe, en particulier dans le cadre des environnements contraints.

Dans ces conditions la topologie du terrain est si complexe que les moyens de déplacement classiques tels que la marche ou la course ne s'appliquent plus.

Constatant l'impossibilité pour ces modèles classiques à fournir des solutions génériques et robustes, cette thèse s'inscrit dans la lignée des travaux robotiques procéduraux qui reformulent le problème comme celui de la recherche d'une séquence de contacts permettant d'atteindre l'objectif.

Plus précisément, deux questions de recherche ont été considérées:

1. Quelles nouvelles heuristiques peuvent être proposées afin de générer des postures de contact pertinentes pour une tâche et un environnement donné?

2. Comment les intégrer dans une méthode de planification de mouvement qui calcule des trajectoires pertinentes dans des environnements très contraints?

En réponse à la première question, nous avons proposé une nouvelle heuristique appelée EFORT, pour Extended FORce Transmission ratio. EFORT est un outil qui évalue la compatibilité d'une posture avec une tâche demandant d'exercer une force dans une direction donnée. Conjointement à EFORT nous avons proposé une méthode permettant de générer en temps réel une posture de contact maximisant localement cette heuristique. Cette méthode est automatique, très simple à implémenter, et intégrable directement au sein de méthodes d'animation existantes.

Pour répondre à la deuxième question, nous nous sommes appuyés sur l'heuristique EFORT afin de proposer un nouveau planificateur de mouvement, le "Reachability Based PRM", ou RB-PRM. Il s'appuie sur des heuristiques géométriques très simples

qui permettent de générer très rapidement des séquences de contact dans des environnements contraints. Ces séquences de contact servent ainsi de trajectoires guides à des méthodes d'optimisation de trajectoire, qui autrement n'arriveraient pas à converger dans des environnements contraints.

Parmi les limitations de notre méthode, on retiendra que EFORT est une heuristique purement géométrique et incomplète, qui ne tient pas compte d'autres aspects biomécaniques comme les modèles musculaires. De la même manière, notre planificateur de mouvement s'appuie sur des heuristiques géométriques simples pour trouver une trajectoire, dont on ne peut garantir la pertinence absolue.

Ces limitations vont être étudiées dans le but d'une application directe à la robotique au cours de travaux futurs.

Author's publications

Articles

Journal papers

J1. **S. Tonneau**, J. Pettré, F. Multon. "Using task efficient contact configurations to animate creatures in arbitrary environments ". *Computer & Graphics*, vol. 45, pp. 40-50, 2014.

Conference papers

C1. **S. Tonneau**, J. Pettré, F. Multon. "Task efficient contact configurations for arbitrary virtual creatures". *Proceedings of the 2014 Graphics Interface conference*, GI '14, pp. 9-16, 2014.

Bibliography

[01h01] High-level specification and animation of communicative gestures, 2001.

[AAKC13] Rami Ali Al-Asqhar, Taku Komura, and Myung Geol Choi. Relationship descriptors for interactive motion adaptation. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '13, pages 45–53, New York, NY, USA, 2013. ACM.

[ABL+98] Nancy M. Amato, O. Burchan, Bayazit Lucia, K. Dale, Christopher Jones, and Daniel Vallejo. Choosing good distance metrics and local planners for probabilistic roadmap methods. In *In Proc. IEEE Int. Conf. Robot. Autom. (ICRA*, pages 630–637, 1998.

[AdLH12] Mazen Al Borno, Martin de Lasa, and Aaron Hertzmann. Trajectory Optimization for Full-Body Movements with Complex Contacts. *IEEE transactions on visualization and computer graphics*, pages 1–11, December 2012.

[AW96] N.M. Amato and Y. Wu. A randomized roadmap method for path and manipulation planning. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 1, pages 113–120 vol.1, Apr 1996.

[BB04] Paolo Baerlocher and Ronan Boulic. An inverse kinematics architecture enforcing an arbitrary number of strict priority levels. *The Visual Computer*, 20(6), June 2004.

[BELK09] K. Bouyarmane, a. Escande, F. Lamiraux, and a. Kheddar. Potential field guide for humanoid multicontacts acyclic motion planning. *2009 IEEE International Conference on Robotics and Automation*, pages 1165–1170, May 2009.

[BRL+04] Timothy Bretl, Stephen Rock, Jean-Claude Latombe, Brett Kennedy, and Hrand Aghazarian. Free-climbing with a multi-use robot. In Marcelo H. Ang Jr. and Oussama Khatib, editors, *ISER*, volume 21 of *Springer Tracts in Advanced Robotics*, pages 449–458. Springer, 2004.

[Chi87] S Chiu. Control of redundant manipulators for task compatibility. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 1718–1724, 1987.

[CKJ+11] Stelian Coros, Andrej Karpathy, Ben Jones, Lionel Reveret, and Michiel van de Panne. Locomotion Skills for Simulated Quadrupeds. *ACM Transactions on Graphics*, 30(4):Article TBD, 2011.

[CLS03] Min Gyu Choi, Jehee Lee, and Sung Yong Shin. Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Transactions on Graphics*, 22(2):182–203, 2003.

[EKMG08] Adrien Escande, Abderrahmane Kheddar, Sylvain Miossec, and Sylvain Garsault. Planning Support Contact-Points for Acyclic Motions and Experiments on HRP-2. In Oussama Khatib, Vijay Kumar, and George J Pappas, editors, *ISER*, volume 54 of *Springer Tracts in Advanced Robotics*, pages 293–302. Springer, 2008.

[FHKS12] Andrew W. Feng, Yazhou Huang, Marcelo Kallmann, and Ari Shapiro. An analysis of motion blending techniques. In *International Conference on Motion in Games*, Rennes, France, November 2012.

[GKNK06] L Guilamo, J Kuffner, K Nishiwaki, and S Kagami. Manipulability optimization for trajectory generation. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2017–2022, 2006.

[HBL05] K Hauser, T Bretl, and J.-C. Latombe. Non-gaited humanoid locomotion planning. In *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, pages 7–12, 2005.

[HKT10] Edmond S. L. Ho, Taku Komura, and Chiew-Lan Tai. Spatial relationship preserving character motion adaptation. *ACM Trans. Graph.*, 29(4):33:1–33:8, July 2010.

[HLM97] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 3, pages 2719–2726 vol.3, Apr 1997.

[HMK11] Yazhou Huang, Mentar Mahmudi, and Marcelo Kallmann. Planning humanlike actions in blending spaces. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[HRB14] D Haering, M Raison, and M Begon. Measurement and description of three-dimensional shoulder range of motion with degrees of freedom interactions. *Journal of biomechanical engineering*, 136(8), August 2014.

[HST94] T. Horsch, F. Schwarz, and H. Tolle. Motion planning with many degrees of freedom-random reflections at c-space obstacles. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 3318–3323 vol.4, May 1994.

[HUF04] Lorna Herda, Raquel Urtasun, and Pascal Fua. Hierarchical implicit surface joint limits to constrain video-based motion capture. In *Computer Vision - ECCV 2004, 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part II*, pages 405–418, 2004.

[JBGR12] Julien Jacquier-Bret, Philippe Gorce, and Nasser Rezzoug. The manipulability: a new index for quantifying movement capacities of upper extremity. *Ergonomics*, 55(1):69–77, January 2012.

[Joh09] R S Johansen. *Automated Semi-procedural Animation for Character Locomotion*. Aarhus Universitet, Institut for Informations Medievidenskab, 2009.

[KE08] Abderrahmanne Kheddar and Adrien Escande. Planning of contact supports for acyclic motion of humanoids and androids: challenges and future perspectives. In *International Symposium on Robotics*, 2008.

[KGP02] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. In *ACM Transactions on Graphics*, volume 21, pages 473–482, New York, NY, USA, 2002. ACM.

[KHB10] Marcelo Kallmann, Yazhou Huang, and R. Backman. A skill-based motion planning framework for humanoids. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2507–2514, May 2010.

[KJ98] James J. Kuffner and Jr. Goal-directed navigation for animated characters using real-time path planning and control. In *In Proceedings of Captech'98*, pages 171–186. Springer-Verlag, 1998.

[KM04] M. Kallman and M. Mataric. Motion planning using dynamic roadmaps. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 5, pages 4399–4404 Vol.5, April 2004.

[KMA05] R Kulpa, F Multon, and Bruno Arnaldi. Morphology-independent representation of motions for interactive human-like animation. *Computer Graphics Forum*, 24(3):343–351, 2005.

[KNK+03] James Kuffner, K. Nishiwaki, Satoshi Kagami, Y. Kuniyoshi, M. Inaba, and H. Inoue. Online footstep planning for humanoid robots. In *IEEE*

Int'l Conf. on Robotics and Automation (ICRA'2003). IEEE, September 2003.

[KSLO96] L E Kavraki, P Svestka, J.-C. Latombe, and M H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580, 1996.

[KvdP01] Maciej Kalisiak and Michiel van de Panne. A grasp-based motion planning algorithm for character animation. *The Journal of Visualization and Computer Animation*, 12(3):117–129, July 2001.

[LaV98] S M LaValle. Rapidly-Exploring Random Trees: A New Tool for Path Planning. *In*, 129(98-11):98–11, 1998.

[LaV06] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, New York, NY, USA, 2006.

[LCH03] Tsai-Yen Li, Pei-Feng Chen, and Pei-Zhi Huang. Motion planning for humanoid walking in a layered environment. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 3, pages 3421–3427 vol.3, Sept 2003.

[LH03] P. Leven and S. Hutchinson. Using manipulability to bias sampling during the construction of probabilistic roadmaps. *Robotics and Automation, IEEE Transactions on*, 19(6):1020–1026, Dec 2003.

[LK06] Manfred Lau and James J. Kuffner. Precomputed search trees: planning for interactive goal-driven animation. In *Symposium on Computer Animation*, pages 299–308, 2006.

[LL04] Jehee Lee and Kang Hoon Lee. Precomputing avatar behavior from human motion data. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '04, pages 79–87, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[LLDM05] Joanne E Labriola, Thay Q Lee, Richard E Debski, and Patrick J McMahon. Stability and instability of the glenohumeral joint: The role of shoulder muscles, January 2005.

[LMEE12] Mingxing Liu, Alain Micaelli, Paul Evrard, and Adrien Escande. Task-driven posture optimization for virtual characters. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '12, pages 155–164, Aire-la-Ville, Switzerland, Switzerland, 2012. Eurographics Association.

[LP83] Tomás Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C-32:108–120, 1983.

[LP12] Sergey Levine and Jovan Popovic. Physically Plausible Simulation for Character Animation. In *Symposium on Computer Animation*, pages 221–230, 2012.

[Lun12] J Lundgren. Inpolyhedron - are points inside a volume? *MATLAB Central File Exchange, http://tinyurl.com/ktcgohk*, 2012.

[MK12] Mentar Mahmudi and Marcelo Kallmann. Precomputed motion maps for unstructured motion capture. In *Eurographics/SIGGRAPH Symposium on Computer Animation (SCA)*, 2012.

[MKHB15] Robert Peter Matthew, Gregorij Kurillo, Jay J. Han, and Ruzena Bajcsy. Calculating reachable workspace volume for use in quantitative medicine. In Lourdes Agapito, Michael M. Bronstein, and Carsten Rother, editors, *Computer Vision - ECCV 2014 Workshops*, volume 8927 of *Lecture Notes in Computer Science*, pages 570–583. Springer International Publishing, 2015.

[MMKA04] S. Menardais, F. Multon, R. Kulpa, and B. Arnaldi. Motion blending for real-time animation while accounting for the environment. In *Computer Graphics International, 2004. Proceedings*, pages 156–159, June 2004.

[MTP12] Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics*, 31(4):43:1—-43:8, 2012.

[MWTK13] Igor Mordatch, Jack M. Wang, Emanuel Todorov, and Vladlen Koltun. Animating human lower limbs using contact-invariant optimization. *ACM Trans. Graph.*, 32(6):203:1–203:8, November 2013.

[NL06] N Naksuk and C S G Lee. Zero moment point manipulability ellipsoid. In *ICRA 2006 Proceedings*, pages 1970–1975, 2006.

[NSL99] Carole Nissoux, Thierry Siméon, and Jean-Paul Laumond. Visibility based probabilistic roadmaps. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients, October 17-21,1999, Hyundai Hotel, Kyongju, Korea*, pages 1316–1321, 1999.

[PHL91] P. Pignon, T. Hasegawa, and J.-P. Laumond. Optimal obstacle growing in motion planning for mobile robots. In *Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91. IEEE/RSJ International Workshop on*, pages 602–607 vol.2, Nov 1991.

[PHL92] P. Pignon, T. Hasegawa, and J.-P. Laumond. Basic algorithms for space structuring in path planning for mobile robots. In *Robotics and Automa-*

tion, 1992. Proceedings., 1992 IEEE International Conference on, pages 2495–2500 vol.3, May 1992.

[PLS03] Julien Pettré, Jean-Paul Laumond, and Thierry Siméon. A 2-stages locomotion planner for digital actors. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, pages 258–264, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[PMM+07] Manuel Peinado, Daniel Meziat, Damien Maupu, Daniel Raunhardt, Daniel Thalmann, and Ronan Boulic. Accurate on-line avatar control with collision anticipation. In *Proceedings of the 2007 ACM Symposium on Virtual Reality Software and Technology*, VRST '07, pages 89–97, New York, NY, USA, 2007. ACM.

[PPM12] Chonhyon Park, Jia Pan, and Dinesh Manocha. Itomp: Incremental trajectory optimization for real-time replanning in dynamic environments, 2012.

[PZM12] Jia Pan, Liangjun Zhang, and Dinesh Manocha. Collision-free and smooth trajectory computation in cluttered environments. *Int. J. Rob. Res.*, 31(10):1155–1175, September 2012.

[QEMR11] 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.

[RBC98] Charles Rose, Bobby Bodenheimer, and Michael F. Cohen. Verbs and adverbs: Multidimensional motion interpolation using radial basis functions. *IEEE Computer Graphics and Applications*, 18:32–40, 1998.

[SH07] Alla Safonova and Jessica K. Hodgins. Construction and optimal search of interpolated motion graphs. *ACM Trans. Graph.*, 26(3), July 2007.

[SLCS04] T. Siméon, J.P. Laumond, J. Cortes, and A. Sahbani. Manipulation planning with probabilistic roadmaps. *International Journal of Robotics Research*, 23(7-8), july 2004.

[SS83] Jacob T. Schwartz and Micha Sharir. On the "piano movers'" problem i. the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Communications on Pure and Applied Mathematics*, 36(3):345–398, 1983.

[SSL12] M. Sreenivasa, P. Souères, and J.P. Laumond. Walking to grasp: Modeling of human movements as invariants and an application to humanoid robotics. *IEEE Transactions on Systems, Man, and Cybernetics*, 2012.

[vBPE10] Ben JH van Basten, PWAM Peeters, and Arjan Egges. The step space: example-based footprint-driven motion synthesis. *Computer Animation and Virtual Worlds*, 21(3-4):433–441, 2010.

[vWvBE+10] H. van Welbergen, B.J.H. van Basten, A. Egges, Zs. M. Ruttkay, and M.H. Overmars. Real time animation of virtual humans: A trade-off between naturalness and control. *Computer Graphics Forum*, 29(8):2530–2554, December 2010.

[Wel93] Chris Welman. Inverse kinematics and geometric constraints for articulated figure manipulation. Master's thesis, Simon Frasor University, 1993.

[WHD+12] Jack M. Wang, Samuel R. Hamner, Scott L. Delp, Vladlen Koltun, and More Specifically. Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Trans. Graph*, 2012.

[WP95] Andrew Witkin and Zoran Popovic. Motion warping. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 105–108, New York, NY, USA, 1995. ACM.

[WPP13] Kevin Wampler, Jovan Popović, and Zoran Popović. Animal Locomotion Controllers From Scratch. *Computer Graphics Forum*, 32:153–162, May 2013.

[WPP14] Kevin Wampler, Zoran Popović, and Jovan Popović. Generalizing locomotion style to new animals with inverse optimal regression. *ACM Trans. Graph.*, 33(4):49:1–49:11, July 2014.

[YKH04] Katsu Yamane, James Kuffner, and Jessica K Hodgins. Synthesizing Animations of Human Manipulation Tasks. *ACM Trans. on Graphics (Proc. SIGGRAPH 2004)*, 2004.

[YLvdP07] KangKang Yin, Kevin Loken, and Michiel van de Panne. Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics*, 26(3):Article 105, 2007.

[Yos85] Tsuneo Yoshikawa. Manipulability of Robotic Mechanisms. *International Journal of Robotic Research*, 4(2):3–9, 1985.

[YTEA] Hsin-Yi Yeh, Shawna L. Thomas, David Eppstein, and Nancy M. Amato. Uobprm: A uniformly distributed obstacle-based prm. In *IROS*, pages 2655–2662. IEEE.

List of Figures

List of Tables

INSA de RENNES
Service des Formations

AVIS DU JURY SUR LA REPRODUCTION DE LA THESE SOUTENUE

Titre de la thèse:
Synthèse et planification de mouvement pour des personnages virtuels en environnements contraints

Nom Prénom de l'auteur : TONNEAU STEVE

Membres du jury :
- Monsieur ARNALDI BRUNO
- Monsieur MULTON Franck
- Monsieur PETTRÉ Julien
- Monsieur LAUMOND Jean-Paul
- Monsieur KALLMANN Marcelo
- Monsieur KOMURA Taku
- Monsieur EGGES Arjan

Président du jury : Bruno ARNALDI

Date de la soutenance : 27 Février 2015

Reproduction de la these soutenue

Thèse pouvant être reproduite en l'état
~~Thèse pouvant être reproduite après corrections suggérées~~

Fait à Rennes, le 27 Février 2015

Signature du président de jury

Le Directeur,

M'hamed DRISSI

Résumé

Avec la complexité croissante des environnements virtuels apparaît le besoin de doter les personnages qui les peuplent d'une plus grande autonomie de mouvement. En plus de marcher, courir et sauter, les simulations interactives actuelles requièrent des personnages qu'ils rampent, escaladent, poussent ou tirent des objets... Ces tâches sont caractérisées par les environnements contraints dans lesquelles elles sont réalisées, qui présentent un risque fort de collision et réduisent fortement les possibilités de mouvement; elles le sont aussi par les forces importantes qui doivent être exercées afin de les réaliser, résultant de la création de contacts. Ces deux aspects rendent la synthèse automatique de mouvements très difficile dans ce contexte.

Cette thèse a pour objectif de proposer une méthode automatique pour la synthèse de mouvements en environnements contraints. Pour ce faire, deux problématiques de recherche ont été posées et étudiées.

La première partie de la thèse porte sur la question de la génération de contacts pertinents pour la réalisation des tâches considérées. Une nouvelle heuristique appelée EFORT (Extended FORce Transmission ratio) est présentée ; elle permet d'évaluer la compatibilité d'une posture de contact avec la tâche demandée. Cette heuristique est au cœur d'une méthode pour la génération temps réel de postures de contact. Cette méthode s'applique pour des personnages et des environnements arbitraires, et peut être directement intégrée au sein de simulations interactives telles que les jeux vidéo.

La deuxième partie porte sur le problème plus global de la recherche d'une trajectoire pertinente dans un environnement contraint. Cette recherche de trajectoire passe par la recherche d'une séquence de postures de contact qui vont permettre le mouvement. Une nouvelle méthode de planification de mouvement s'appuyant sur EFORT est donc proposée.

Parce qu'elle est une des premières à simultanément considérer la complexité de l'environnement et la pertinence des configurations générées au regard de la tâche à accomplir, notre méthode constitue un pas significatif vers une plus grande autonomie de mouvement pour les personnages virtuels.

Abstract

With the growing complexity of virtual environments comes the need to provide virtual characters with a larger autonomy of motion. Additionally to walking, running and jumping, state of the art virtual applications require characters to climb, crawl, pull or push objects... Those tasks are characterized by the constrained environments in which they are achieved, where the risk of collision is high and motion capabilities are limited; they are also associated with important force exertion, resulting from contact creation. In this context, automatic motion synthesis is really difficult.

This thesis aims at proposing an automatic method for motion synthesis in constrained environments. To achieve these goals, two research problems have been identified and studied.

The first part is dedicated to the issue of generating contact postures compatible to achieve the considered tasks. We propose a new heuristic called EFORT (Extended FORce Transmission ratio). EFORT is used to evaluate the compatibility of a contact posture with the requested task. EFORT lies at the center of a new method for the real time generation of task efficient contact configurations. This generator finds its applications for arbitrary virtual characters and environment, and as such can be directly integrated within video game applications.

The second part of this thesis focuses on the more global issue of computing a relevant trajectory in a constrained environment. This issue is seen as the search for a sequence of task efficient contact postures, suited for achieving the task. Consequently a new motion planner based on EFORT is proposed.

Because it is one of the first to simultaneously address the complexity of the environment and task efficiency, our motion planner is a significant step towards an enhanced autonomy of motion for virtual characters.

N° d'ordre : 15ISAR 04 / D15 - 04

Institut National des Sciences Appliquées de Rennes
20, Avenue des Buttes de Coëmes • CS 70839 • F-35708 Rennes Cedex
Tel : 02 23 23 82 00 - Fax : 02 23 23 83 96