# Stable Humanoid Whole Body Motion Generation

Sharon Yalov-Handzel

Department of Computer Science

**Ph.D. Thesis**

Submitted to the Senate of Bar-Ilan University

**Ramat-Gan, Israel** **September 2015**

**This work was carried out under the supervision of Prof. Gal Kaminka**

The Department of Computer Science, Bar-Ilan University

# Acknowledgments

It would not have been possible to write this doctoral thesis without the help and support of the kind people around me, to only some of whom it is possible to give particular mention here.

Above all, I would like to thank my supervisor Prof. Gal Kaminka. This thesis would not have been possible without his help, support and patience, not to mention his advice and unsurpassed knowledge. I could not have asked for better role mentor, inspirational, teaching me how to think on academic problems, always knows to ask leading questions and creates fertile atmosphere.

Then, I would like to express my deep gratitudes to the adults of my family and especially my parents, Arye and Havazelet for their unequivocal support and great patience at all times, for which my mere expression of thanks likewise does not suffice.

I would like to express my heartfelt gratitude to the lab team: Gabriella, Mor, Matan, Noa and her group for their good advices, support and friendship. It has been invaluable on both an academic and a personal level.

I would like to acknowledge the financial, academic and technical support of the Bar-Ilan University and its staff, particularly in the award of a Postgraduate Research Studentship that provided the necessary financial support for this research.

This thesis was co-funded by several resources. I would like to thank especially Mr. Sami Segol for his generous stipend.

I also thank the Department of Computer Science for their support and assistance since the start of my studying in 2010, especially the head of department, Prof. Doron Peled and Prof. Sarit Kraus.

I am most grateful to the Israeli group that participated in Darpa Grand Challenge for providing me with ideas and directions. Special thanks to Prof. Hugo Guterman from BGU, Prof. Miriam Zacksenhouse from the Technion, Dr. Amir Shapiro from BGU and Eliya Shaviv from the IAI. All of them had provided a rich and fertile environment to explore new ideas. My other affiliation in the BIU group: Dudi Mass, Shimshon Winograd, Gal Lavi and Ella Sharkansky, thank you

Last, but by no means least, I thank my beloved kids Eran and Noga for being there.

For any errors or inadequacies that may remain in this work, of course, the responsibility is entirely my own.

# Abstract

The stability of humanoids is fragile and depends on the robot dynamics which are difficult to plan in advanced. Keeping humanoids stable along their non-repetitive motion sequences is a great challenge in robotics, and it is usually resolved by using feedback control to consider the dynamic impacts. Off-line planning of whole body stable motion cannot be resolved by motion generators based on feedback control. We propose an algorithm to plan stable whole body postures, based on the IKP (Inverse Kinematics Problem) solver. The IKP is extended to solve the kinematic equations under any condition that can be represented in a geometrical form. Such a condition is the robot stability. This numerical iterative algorithm can be applied to any robot structure. In addition, the algorithm can find a kinematic solution that obey a certain optimization criteria. But is not complete. In order to cope with the algorithm's incompleteness, we propose some improvements that increase the convergence probability of the solution. The algorithm was analyzed and simulated and the core contribution of this work is that it gives a general method for off-line planning of complex postures applied to high DOF (degree of freedom) robots that should obey some condition. Moreover, the improvements demonstrate that there is a tradeoff between the completeness and the generality of the solver.

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Chapter 1

# Introduction

This dissertation describes research on robot motion planning under constraints. In particular, the research is focused on complex robot structures such as humanoids, that requires coping with crucial challenges in robot motion planning, such as keeping stability, following a path accurately, coordinating limbs and more.

Robot motion planning is a wide term relating different aspects. It ranges from environment considerations such as obstacle avoidance to robot controller considerations such as minimizing the frequency of sending motion commands. It includes optimal path planning as well as obeying kinematics constraints and limitations. The term robot motion planning covers the topics of translating high level tasks into low level motion, the optimality of the motion, keeping stability, reuse of repetitive motions, saving energy, reduction in joint engines power consumption, coping with dynamic environment, path predictability, the impact of the sensing limitations and more.

This research is focused on the problem of planning robot postures that reach given target poses, without losing stability during the transition from the posture. We propose and investigate a family of algorithms that plan motions by solving inverse-kinematics problem under constraints.

The proposed approach is relevant to any robot. However, we will focus on humanoids, that embody the most general requirements coping with complex kinematic structure with many degrees of freedom, kinematic chains in both serial and parallel relations, fragile stability and singular points.

Motion planning for non-humanoid robots require coping with sub-sets of these challenges. For example, planning the motion of an industrial manipulator does not typically require coping with stability, since the robot is mounted to the ground or a wall. Also, an industrial manipulator has one sequential kinematic chain which eases

the computation. Another example is most wheeled robots, that also do not require accounting for their stability. Considering multi-legged robots (e.g. insect-like), their kinematic chains are always parallel and do not change their characteristics. So, biped robots are a general structure that embody all sub-problems in robot motion planning.

## 1.1 Brief Overview of Robot Motion

The history of robot motion is a background for understanding current robotics research. The evolution to walking robots deals with complex motion control, structural design and walking patterns. An advantage of walking robots in relative to manipulators or even wheeled robots is in their versatility which enables wide range of locomotion capabilities which means supporting variety of tasks. The disadvantage of the walking robots is their fragile stability.

Motion planning is a term used for the process of breaking down a desired high level task into discrete low level motions that satisfy movement constraints and possibly optimize some aspect of the movement.

This process is composed of the following sub-processes:

1. Transform a task into atomic substasks
2. Sense, measure and calibrate a model of the environment
3. Construct a view of the world and the robot locomotion within this environment
4. Translate each atom subtask into a sequence of paths for the motion of the robot end-effector.
5. Ensure continuity of the transition between paths.
6. Plan a trajectory for the robot end effector that ensures obstacle avoidance and feasibility
7. Translate each trajectory into a sequence of via points in terms of the robot end-effector in relative to the world coordinate system.
8. Convert the via points into machine robot postures.
9. Make sure that these points are valid in terms of reachability, stability, feasibility and singular points avoidance.

10. Translate these postures into machine commands

11. Optionally change the interpolated path generated by the controller in each piecewise transition.

This research is focused on sub-problems 8–11, while dealing with biped robots. In its framework we relate any whole body motion under the stability constraint.

Each of the above sub processes embodies a wide research domain. Some major research problems in motion planning are outlined below:

- motion planning of biped robots in human environment and working in cooperation with humans
- the motion mechanism of human and animals as the basis for understanding how robot motion should be implemented.
- What is the most appropriate stability criteria for maintaining stability during the walking motion which takes into account whole body biped dynamics.

As described in [53], there are two major approaches in robot motion planning: the classical and the heuristic. Both approaches relate stages 4-6 in the above list of sub processes. Other stages are less investigated by computer science scientists.

While sub processes 1-5 are covered by the domain of robot motion planning, it is common to consider sub processes 6-11 as being handed by the controller.

Dealing with biped robot, the most common task is walking to a target. This can be divided into clear two phases: planning the ground projected walking path, and walking along this path. There is a clear separation between the planning and the controlling phases. Our research is not limited to walking, but to any whole body motion tasks. In this case the separation is not important.

There are three major approaches in planning and controlling in general motion tasks:

- Sensor-based (e.g. Feedback Control)
- Human physiology-based (e.g. CPG - Central Pattern Generator)
- Geometry-based (e.g.IKP - Inverse Kinematics Problem)

While the two later approaches can belong to both motion planning stage or to the Control stage, the first belongs only to the control phase.

In this research we are using geometric approach, and focusing on the IKP method, which enables both off-line planning and real-time control.

## 1.2 Humanoids

Humanoids, sometimes refereed to as biped robots, have structure resembling the human body. There are many types of biped robots that all have in common the following characteristics:

- They have two legs and two hands
- They have at least four kinematic chains
- The number of DoF ranges often in 18–32, and could be larger
- Their controllers support a stability mechanism

Currently they are mainly used for research, but the vision is that they will replace humans in dangerous, repetitive and tiring tasks, such as rescue forces in radioactive environments, unskilled assistants in elderly care, etc. They have wide range of motions that enable variety of tasks and their configuration space is unlimited.

These structures are challenging, and there are many open problems yet to be solved, such as: What is the best stabilizing mechanism, How to fuse the information from different sensors, What is the optimal motion planning, and more.

Although biped robots are a specific type of robots, the challenges they raise are general. We choose to focus our research on this robot type since they are most challenging in the computational kinematics aspect and each solution that is relevant to them will hold for any other type of robots.

Uncertainty is a crucial issue in robotics since there is no robotic system with certainty, from one hand, while on the other hand, the uncertainty has to be considered in any motion planning.

The uncertainty exists due to several reasons: the environment is dynamic, the calibration of the robot in relative to the world is not accurate, the precision of the robot motion is bounded. Moreover, there are some systematic deviations of any dynamic system, such as the gap between the theoretical and physical robot initial pose. Finally, there is always "white noise" in measuring, especially in systems that their perception is based on information fusion from different resources.

An inherent property of the uncertainty is that it is unknown. But estimation of its maximal boundaries can be determined. This estimation is important for predicting maximal deviations from the "known" scenario, in order to prevent failures such as losing stability, falling, damaging, etc.

## 1.3 The contributions of the research

This research extends the classical kinematics solution and adjusts it to complex robot structures such as humanoids.

- First, it modifies a traditional IKP method to meet the characteristics of multi-chains robots.
- Then, it extends the IKP to be solved under constraints. As an example, the stability constraint is described. This enables solving the IKP, which is a geometrical problem under different physical constraints in general and particularly those stemming from dynamics. This is an advantage, since it enables working with complex devices such as biped robot on diverse whole body motions, not limited to walking only. The algorithm is offline and using the CoM criterion implies quasi-static movement.
- Another topic that is covered by this research is the analysis of the impact of the interpolation done by the robot controller in the joints' configuration space. In this context different interpolations are compared.
- As a consequence of this analysis, an algorithm that determines the density of the via points between two consequent postures is proposed, analyzed and simulated.
- The problem of motion planning for stability which can be determined off-line without reliance on real-time sensory inputs is tackled. The key idea in our approach is to translate stability and dynamics constraints into geometric constraints and then utilize a novel, general IKP solver to plan provably stable trajectories.

The result is a toolbox for off-line planning of complex motions of complicated robot structures in dynamic and uncertain environments. It is based, in part, on twenty years of experience in the kinematics of industrial robot, and some insights for the "Grand Challenge" 2013 competition initiated by DARPA.

This dissertation compounds of 9 chapters, where Chapter 3 extends the general IKP definition to solve multiple targets and multiple supports. Thereafter, in Chapter 4, a novel algorithm to solve the IKP under conditions (ICKP) is proposed. In Chapter 5 we propose some improvements to the ICKP to treats its weaknesses. Then, some constraints are given as example, while focusing on the stability constraints

(Chapter 6). In Chapter 7 the noncontinuous property of motion planning is discussed. Chapter 8 demonstrates some results of the ICKP algorithm while applying stability constraints. Finally, Chapter 9 concludes this dissertation.

# Chapter 2

# Related Work

The spectrum of literature on robot motion planning varies from high level tasks to low level planning and execution. The lowest level tasks are the commands to robot controller and the highest tasks are those that should be translated into a complicated set of lower level commands (e.g. "pick an object", which includes walking, reaching, recognizing an object and grasping it). A common denominator of all motion planning methods is their goal to allow the robot to determine its motion autonomously. As robots become more complex in their structure and more versatile in their mobility, this challenge becomes more crucial and more difficult. A comprehensive survey of motion planning methodologies for robot manipulators appears in Sciavicco et al. [67].

In the 80's and part of the 90's robot motion planning was focused on finding collision-free path in the joints space. Today, there are many other constraints that need to be taken into account. Among them are: equilibrium, optimality across multiple criteria, kinodynamic (i.e. simultaneous kinematic and dynamics constraints), uncertainty in actuation, maximal coverage of reaching zones, visibility and more [11, 30, 47].

Our focus is on planning stable motion in general and specifically for humanoid robots. We survey related work below.

## 2.1 Robot Motion Planning

The term "robot motion planning" covers two distinct research domains: (i) path planning and (ii) joint configuration planning. In this research we concentrate on the later domain. i.e. given the target point after a pace, what is the most appropriate robot posture that reaches the target under stability constraint at the end point and during the transition.

The majority of works in the domain of algorithmic robot motion planning are focused on four aspects: (i) manipulators motion planning, (ii) walking of humanoids, (iii) humanoid grasping and (iv) whole body OMPL (Optimal Motion Planning).

The concrete requirements of the motion planning, determines which solution is relevant. Some aspects that have to be considered before selecting the planning solution are:

- On-line or off-line planning
- The robot is inherently stable (e.g. manipulator, wheeled robot) or not (e.g. humanoid)
- The planning is in the Cartesian space (e.g. accurate path tracking) or in the joints space (e.g. only reachability is important)
- Motion is repetitive (e.g. walking) or not (e.g. standing after falling)

There are three families of algorithms for robot motion: physical-based (e.g. feedback control), physiological-based (e.g. Central Pattern Generator or Incremental sampling-based motion planning algorithms) and geometrical-based (e.g. Inverse Kinematics Problem). The former one is used for real-time motion rather than planning a motion. The last one is useful in manipulator motion planning [30] while the other approaches are common in biped locomotion.

There are two types of robot motion controllers: open loop and closed loop. In the open-loop approach the next movement is based only on the robot current state. In closed loop controllers the sensory inputs affect the next step, as well. This means that motion planning for robots with open-loop controllers can be fully predetermined, while for closed-loop controllers a pre-planning can be done partially only (i.e. roughly in the Cartesian level rather than in the joint space).

The most common planners belong to the Feedback control family of algorithms which is an on-line response of the robot controller to the current robot state as achieved from sensory-based information fused with the controller parameters. The planning is in the path level and the joint actuation is mainly performed as a feedback response. In most feedback methods the controllers are designed at the torque input level and the actuator part is neglected. The generated motion is as to obey the robot goal, and the feedback model as was defined. There are different feedback models in different levels, such as: angular momentum, pendulum, torque, stability and more [33, 34, 40, 43, 72, 93]. Feedback control methods are also relevant in motion

generation of these robots, but the feedback is on reachability parameters rather than on stability.

The most popular planner of the physiological-based approach is the CPG (Central Pattern Generator) family of algorithms. The term CPG was originally related to neural networks that exist in the spinal cord of animals and is used as muscles activation signals. These networks produce rhythmic patterned outputs without sensory feedback [52]. This idea was adopted in many robot motion planners and controllers and became popular mainly in generating repetitive motions in complex robots that do not rely on any input sensory information [28]. Our motion planner either does not rely on sensory inputs. While CPG planner can generate motions that are already defined, our planner can generate new motions in the same quality as those it already planned.

The most significant planners of the geometrical approach are those solving the IKP (Inverse Kinematics Problem). This family of algorithms is useful in manipulators motion planning since they are stable and have low degree of freedom. These characteristics match the common assumptions of traditional IKP solvers. The traditional IKP is not efficient in humanoid motion planning due to two major problems: (i) stability is not addressed and (ii) the complexity of solving the IKP of high order kinematic equations. Therefore, dealing with humanoid, IKP is used in two cases, where only partial motion planning is required. One case is to determine grasping positions, that are similar to manipulators. i.e., the joints state of the hand relative to a stable body [7]. Usually, it does not affect the stability much. The other case is to determine the internal joint values within a given kinematic chain, where the position of the chain's origin is well defined. In this dissertation we do not limit ourselves to these two cases.

## 2.2 Constraints in Motion Planning

There are many robot types that differ in their number of joints, number of kinematic chains, type of kinematic chains, size, degree of freedom, whether they are mounting or not, and more. The constraints that should be satisfied along the robot movement are partially derived from its characteristics.

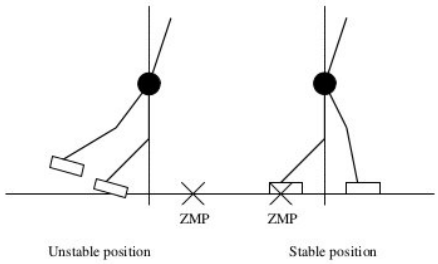



Figure 2-1: *Zero Moment Point( [13]). Dynamic walking is achieved by ensuring that the robot is always rotating around a point in the support region*

### 2.2.1 Stability

There are three common criteria for stability: CoM (Center of Mass), ZMP (Zero moment Point) and FRI (Foot Rotation Indicator). Both the CoM and the ZMP criteria state that as soon as the Center of Mass or the Zero Moment Point is contained within the interior of the support polygon, the robot is stable. The FRI is the point on the ground where the net ground reaction force would have to act to keep the robot stationary. Thus, this point is within the convex hull of the stance foot.

Stability became a crucial issue in robotics with the rising of humanoid motion control. Until this period, when most of the robots were manipulators, their stability was not a crucial issue since they were mounted. Comparing biped and wheeled robots, while the former have better mobility, they tend to tip over easily [50]. A pioneer of researching biped robot stability was Kobratović who introduced in 1969, together with Jurićić, the ZMP (Zero Moment Point) concept and its dynamic equilibrium, based on the problem of humanoid gait modeling [89]. Thirty five years later, in 2004, Vukobratović and Borovac had published a review of ZMP [87]. A significant progress was made in the analysis of the dynamic balance loss. In this case an advanced stability model of the FZMP [88] was adopted. In 1999 Goswami had formalized the FRI (Foot Rotation Indicator) stability criteria [22], which is useful in generating biped robot stable running or other biped moments that have no static stability. Two other stability criteria are keeping the CoM (Center of Mass) or the CoP (Center of pressure) within the convex hull of the robot supported polygon. These two criteria are described in [24, 34, 65].

The use of CoM, ZMP and FRI stability criteria might varied from planning to controlling stable postures. Our research is focused on off-line planning of stable

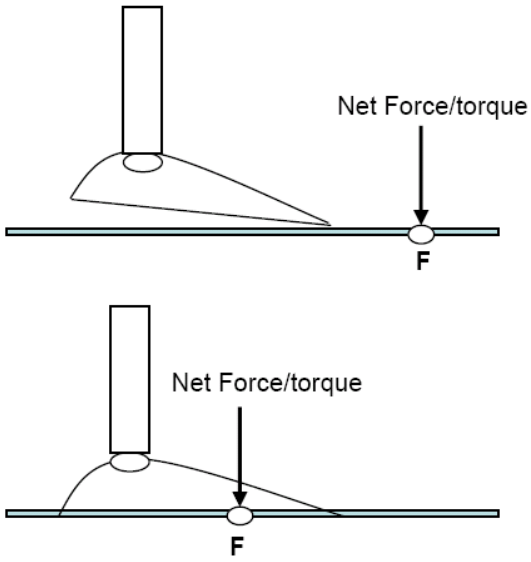



Figure 2-2: *FRI point ( [85]). The further away the FRI point from the support polygon boundary, larger the unbalanced moment and greater is the instability*

postures. The more frequent use of these criteria is in feedback control. Usually it is implemented as a part of the controller which should react in real-time. This type of implementation requires: (i) the performance of the stability criteria will be fast [13] (ii) there should be an indication regarding the stability resistance. i.e. it is not enough to know the binary value whether a posture is stable or not, one should know the amount of "fragility" of the stability. The only criteria that hints about it is the FRI. There are some works that attempt to predict situations that the robot loss its stability and prevent them [96].

Stability mechanism is usually embedded in the controller, as a feedback control on the robot move as for example can be seen in [54]. The advantage of it is that the stability is resistant also to external forces. The disadvantage of this implementation is that it is impossible to preprocess stable motions.

There are many papers about robot stability that describe principles of designing stable robots in the context of its mechanism (e.g. [31]). The reviewed stability models will hold for any robot mechanism.

Although there is a wide research on robot stability conditions, there are very few works on the stability measurement criteria, especially for the case of dynamic balance. The paper of Clark and Cutkosky [12], describes three different measures of stability that can be applied to a runner robot. In this research we solve the IKP subject to stability constraint, which is actually the CoM criteria translated

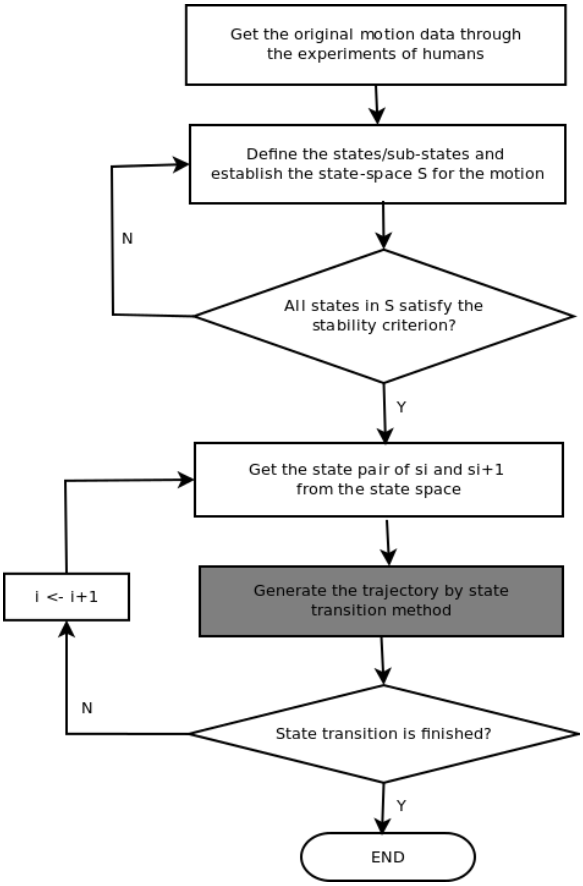



Figure 2-3: *Flowchart of keeping stability during transition*

into geometrical equation. The two other stability criteria can be applied to the constrained IKP as well.

### 2.2.2 Continuity

It is well known how to keep stability discretely, i.e. at a fixed posture. There is no complete solution for keeping the stability during transition between two stable postures. This problem is described in [95] which summarized it into the flow chart 2-3.

### 2.2.3 Minimal Uncertainty

Uncertainty is an inherent challenge of robotics and it stems from four different sources:

- uncertainty of the robot structure (systematic deviations)

- uncertainty of the robot motion (random deviations)
- the dynamics of the environment
- Deviations in the calibration process between the robot and its environment.

The robot stability might be affected by these uncertainties. In the last two decades, the importance of this issue had intensified and many research works had been issued. There are several strategies to efficiently plan motion with imperfect state information. A common direction is by modeling the uncertainty by using stochastic analysis or Markov chains [44]. Another direction is to estimate the motion range and then apply "worst case analysis" [76]. An additional approach is dividing a complex task into primitives that are more tolerated to deviations [82]. Notice that uncertainty exists not only in the planning stage, but also in the execution stage.

Uncertainty reduction can be performed by a calibration process of the kinematics and the environment and the referencing of the robot in relative to the environment. In order to achieve an appropriate correction for the kinematic errors, a huge number of measurements with low- numerical correlations have to be performed. However, uncertainty will always exist. It can be reduced but not canceled. So a proper methodology to cope with is required.

The results of the uncertainty computational models traditionally found in the lower levels in robot systems but may have applications in the upper planning levels as well [83]. Considering the uncertainty is done in the control level. Improved uncertainty models that allow estimation of the deviation range and its implication on the overall performance, enable off line motion planning that will require minimal on-line changes. In this research we use the methods for uncertainty estimation to determine the number of via points that are required during motion interpolation to generate a stable motion in high probability.

## 2.3 Motion Interpolation

In robots, interpolation is used to reach the destination point. Smooth interpolation in Euclidean spaces has many applications in robot motion planning. Due to the complexity of interpolation implementation in the Euclidean space, the motion is interpolated in the joints space.

In robotics the interpolation is performed by the controller and it is a built-in module that in some controller enables choosing between several types of interpola-

tions, and in others it is a predefined module. There are several interpolation methods such as joint interpolation, linear interpolation or circle interpolation.

The background for interpolation design and analysis is related to screw theory and Lie algebra. Selig [68] shows how to represent the kinematics of a robot by using Lie Algebra. This enables different manipulations and such as motion planning and analysis in general and motion interpolation in particular. It uses the fact that the exponential maps $e^{\mathfrak{so}(3)} \rightarrow SO(3)$ and $e^{\mathfrak{se}(3)} \rightarrow SE(3)$ are surjective. So, some matrices associated with the joints can be written as exponentials of the form $e^{\theta s}$, where $\theta$ is the joint value and $s$ is the joint screw, whose determination details are described in [68]. The advantage of using this representation is described in [18], which in brief states that the surjective property of the map implies that there is another multivariate map: $\log : SO(3) \rightarrow \mathfrak{so}(3)$ that can be used to perform motion interpolation.

Dealing with motion planning, the interpolation phase is more interruptible in robot simulators and animation than in real-time controllers. There are few interpolation methods that are more common in this context:

- linear joints interpolation
- B-Spline
- Beźier

They are differed in the resultant path, in its smoothness and in the size of the interpolated segment. The last parameter affects the complexity of the computation which is critical in some real-time applications, as described in [10].

Though robot motion is continuous, motion planning is often a discrete task. Some via points are determined and the motion between them is performed by an interpolation, executed by the robot controller [71, 73, 79]. Indeed, there are many methods for robot motion generation [47]. Some of them are based on mapping a 3D trajectory into the robot joints space [46]. Other methods follow a given pattern [6] or imitate human motion [19, 61, 63, 99]. The last approach is a feedback control which generates the motion dynamically [5,80,93]. The common characteristics of all generators is that they are discrete.

Whether in Cartesian or Joint space, the motion is planned piecewise, as a transition from one point to another. These points are called "via points" or "intermediate knots" or "control points". Keeping the robot stability in the via points can be determined in advanced as will be described in chapter 4. But preserving the stability

during the interpolation phase is not easily guaranteed. Therefore, the via points are usually close, so that there is a good chance that the CoM will not exceed the supported polygon between two stable postures. The hidden assumption motion generated in a piecewise fashion is that the via points are close to each other, so the robot can stably reach the next via point from its current position, i.e., that such a trajectory is feasible and the robot will stay stable. There are two problems with this assumption: (i) it does not always hold (ii) even if it holds it is not necessary optimal.

A comprehensive description of interpolation in the joint space is found in [77]. This path planning is mainly used for obstacle avoidance. It uses as much via-points as needed to avoid obstacles. Beside the joints values, also some dynamics values (i.e. velocity and acceleration) are interpolated. This holds for optimal robot dynamics under obstacles constraints. It does not account for the stability and not for the accuracy of following a given Cartesian path.

In the evolution from CNC machines to manipulator robots and then to humanoid robots, the interest in interpolation methods has declined. Joint interpolations had been investigated in the 80's when CNC machines became very popular. Usually, those machines have 3 perpendicular translational joint axes [57]. In these machines there is an option to choose between linear to circular interpolation in the Euclidean space. Dealing with manipulators that usually have rotational joints only, using any interpolation in the Cartesian space requires high density via-points and their corresponding inverse kinematics calculation. So, interpolation in the Euclidean space became less relevant. Therefore, robots are left with the only interpolation method that is valid, which is linear interpolation in the joints' configuration space.

Interpolation of the $SO(3)$ group is commonly done by using Bézier curves [101]. Those curves are very useful for obstacle avoidance in dynamic environment, since they are parameterized curves and can be easily controlled by changing accelerations [48]. Both approaches relate the location of the robot and the smoothness of the motion in terms of PVA (Position, Velocity, Acceleration) continuity. But, do not account for other constraints, such as stability. We demonstrate the impact of different interpolation methods on the deviation of the robot positions from the planned path. This measurement is essential while dealing with off-line motion planning, as no robot motion is accurate, and one should estimate in advance the volume of the gap between the planned and the actual paths.

# Chapter 3

# Inverse Kinematics for Motion Planning

Robot Kinematics is the geometrical representation of the position along the motion. This area evolved as a combination of mathematics and mechanics, since the 1960s. The area emphasizes the geometry of the robot links as a rigid body.

Robot kinematics covers the relationship between the robot structure, i.e. number of links, type of joints, and links length, and its position, velocity and acceleration. This enables computing the actuator forces and torques. It does not cover some robot dynamics parameters such as inertia.

The position of a robot can be represented as a set of kinematic equations. This representation is termed as the "Forward Kinematics". These equations are used the length of the links as constants and the joints values as variables. The forward kinematics equations enable the translation between the joints' configuration space into the Cartesian space. The opposite translation from the World coordinate system into the robot configuration space is called Inverse Kinematics. Once the target position of the end effector is given, the problem is to find a set of joints values that solve the kinematic equations for the given position. The inverse kinematics problem is cast into a system of nonlinear equations or an optimization problem.

This problem is complex due to its properties:

- Highly nonlinear
- Coupled
- There are multiple solutions

As the robot structure becomes more complex, the IKP is much more difficult to

be solved [17]. IKP solves only the geometric problem of a robot postures and it does not deal with any dynamics or stability considerations. This limits its use to mounted devices, wheeled robots or animation that does not account for physical constraints.

There are two major approaches for solving the IKP: Analytic and Numeric. Analytical methods are generally more efficient and reliable than their numerical counterparts, but require specific kinematic structure. Analytic IKP is hardly used in humanoids because of the complexity of solving non linear equations of high order with many coupled variables. Traditionally, IKP which had 5-7 DoF (degrees of freedom) was solved analytically or numerically. This under constrained equations' system is reduced by implying some optimization constraints [21] or some dummy conditions.

The analytic solutions are usually used when the number of joints is relatively small. These are closed form solutions. In some works, by parameterizing some joints as a function of other joints, reduce the DoF and the problem becomes to be analytically solvable [75]. Other works make restrictive assumptions to enable the analytical solution [18].

Numerical methods are usually general and not specific to closed-form robot structure. They are dealing with humanoids, this especially relevant to [36]. A significant advantage of numerical algorithms is that they can be generalized to accommodate additional constraints and objective functions or optimization criteria, whereas the analytic approaches are restricted to 5-7 DoF systems. The disadvantages of numerical methods are that they are slower and they might not find all the solutions [74].

Most numerical methods are using linearization algorithms. Therefore, the IKP solution becomes to be an iterative process [20]. The disadvantage of it is that the complexity might be of high order. The advantage is that it enables implying more constraints rather than using the kinematic equation solely. Some algorithms manipulate the kinematics equations before the numerical solution [2].

The most common numerical algorithms based on lineariztion are using Newton Raphson or the Jacobian matrix. Some remarkable works using Newton Raphson method were presented by Khalil and Pieper in [37, 58]. Among the algorithms that linearize the kinematics equations by using the Jacobian is [8]. The drawback of them is that in some configurations, the alternative Jacobian can lead to "jerky" behavior. This is particularly true for rotational joints when the multibody's links are folded back on each other trying to reach a close target position.

Sometimes, this domain is termed "differential kinematics" and in its general form $f(\theta)$ is defined to be the set of kinematic equations which is a function of the joints

values $\theta$.

Define,

$x = f(\theta)$ and $\dot{x} = J \times \dot{\theta}$

The Jacobian is defined to be $J \equiv \dfrac{\partial f}{\partial \theta}$

The Jacobian matrix of the forward kinematics equations linearly relates end-effector change to joint angles change. In order to solve the inverse kinematics $J$ should be inverted. Since it is not always invertible, we use the pseudo-inverse of this matrix.

$J^+ = J^T(JJ^T)^{-1}$

The minimal norm solution is $\dot{\theta} = J^+\dot{x}$ and the general solution is

$\dot{\theta} = J^+\dot{x} + (I - J^+J)\dot{\phi}$

Where

$\dot{\phi}$, the null space of $J$ is defined to be the set of vectors which have no influence on the constraints.

There are some iterative numerical algorithms that rather than using this linearization, utilize more efficient functions, like [91] that is based on a combination of two nonlinear programming techniques and the forward recursion formulas, with the joint limitations of the robot being handled implicitly as simple boundary constraints.

A basic assumption that is relevant to most of the numerical algorithms is the although positioning a character with rotational degrees of freedom is non-linear, for small changes of an articulated figure's DoFs, it is sufficiently smooth to allow use of an iterative linear IK solver [10].

Beside the analytical and the numerical approaches, there are some algorithms that use neural network or genetic concepts to solve the IKP [14]. It demonstrates how to reduce the complexity by using neural network solution for the IKP. An example of a IKP solution uses a genetic algorithm can be found in [56]. The objective of this optimization is to simultaneously minimize the end-effector's positional error and the robot's joint displacements.

There are other works that demonstrate how to solve the IKP of varying robot structures by using learning algorithms or genetic algorithms [9, 26]. Some research demonstrates analytic solution of complex robot structures, but these solutions hold for a specific robot rather than being general solvers [45]. There are some general IKP solvers that divides the humanoid structure into segments (correlated to the kinematic chains), and each segment has a representative joint that is solved by the solver [94].

Some solutions are coping with the nonlinearity of the problem by utilizing an

apriori knowledge on the equations properties [100]

Some IKP solutions parameterized part of the joints in order to reduce the DoF. In most conventional robots, the joints are independent and the joint limits are simple linear inequality constraints. In a human skeleton many of the joints are coupled because they may easily form closed loops or because they move simultaneously when a single muscle contracts. Thus, in complex joint systems, the number of degrees of freedom can be less than the number of joint variables. In these cases, it is useful to find ways of parameterizing the kinematics other than with joint variables.

A common approach to solve the IKP in humanoids, is to subdivide the robot joints structure into kinematic units for which analytical solutions can be derived and partition an inverse kinematics problem into subproblems for each of these units.

The complexity of such methods is high and does not enable real-time solutions. There are some works that optimize the method to reduce the complexity like [55], that by using well defined character hierarchies reduces the complexity of each iteration. Using methods like neural networks or genetic algorithms is that their averaged performance is reasonable [39].

As soon as an IKP algorithm or a set of algorithms ensures convergence into a solution for any robot structure, and it can be packaged as a "black box" that its inputs are the robot structure, the robot zero-point and a target point, the package outputs at least one IKP solution if there exists one. Most robot simulators have such a mechanism. Such solvers are common as part of the SDK of specific robots like the Aldebran Ltd. package [62], and the ROS package [60].

There are two common techniques that are used by these solvers. The first, partition the robot to sub-units and solves the IKP for each unit separately. The second requires intervention of the user that decides which joints will be changed (since it is a redundant problem that might have few solutions). Also, there are some academic works on these solvers, like [64], that proposes a combination of two algorithms. The first introduces the concept of splitting the kinematic chain in order to satisfy boundary constraints and the second method directly calculates the configuration space approximation in the preprocessing phase.

As mentioned before, originally, the IKP was defined to solve the geometric equations describing the robot kinematics. However, sometimes, additional constraint should be applied to the solution. The basic IKP constrains the position and orientation of a terminal segment or the end effector. But, actually more constraints should be considered, such as constraining certain points on nonterminal segments (e.g. some supports), aiming the end effector, keeping balancing, and collisions avoidance. It is

not always easy to incorporate these constraints into a conventional inverse kinematics formulation. To make matters worse, multiple and possibly conflicting constraints can be simultaneously active, and the system is usually underdetermined or might be overdetermined.

The IKP approach is useful in manipulators motion planning since they are stable and have low degree of freedom. These characteristics match the common assumptions of traditional IKP solvers. Traditional IKP is not efficient in humanoid motion planning due to two major problems: (i) It does not address the stability and (ii) the high complexity of solving the IKP of high order kinematic equations.

The reason is that the forward kinematic equations have the following properties:

- The number of variables is relatively large (identical to the number of joints)
- The variables are coupled
- The equations are high degree polynomials of transcendental functions, which means that they are highly nonlinear.
- The kinematic equations set is redundant
- Although the kinematic equations are continuous, there exists singular points, that the solution should avoid them.

As the number of joints increases, the equations are more complex, since the variables are coupled, and the dependency of some joints in the other variables has more complicated connection.

The number of chains affects the complexity of the IKP in their structure. A kinematic chain might be serial or parallel to another chain. Different kinematic constraints arise in these cases. Dealing with biped robots, its kinematic chains dual purpose, there might behave as a serial chain (e.g. the left hand and the right leg in the case of standing on the leg and reaching an item with the hand), or in a parallel relation (e.g. the left hand and the right leg in 4- supports "dog posture").

Therefore, dealing with humanoid, IKP is used in 2 cases, where only partial motion planning is required. One case is to determine grasping positions, that are similar to manipulators. i.e., the joints state of the hand relative to a stable body [7]. Usually, it is not affected the stability much. In the second case a pseudo inverse kinematics is used, i.e. is of using IKP approach in humanoids is where small number of joints are allowed to change their values and those joints are defined in advanced [81]. While dealing with a whole body motion where the stability has to be kept,

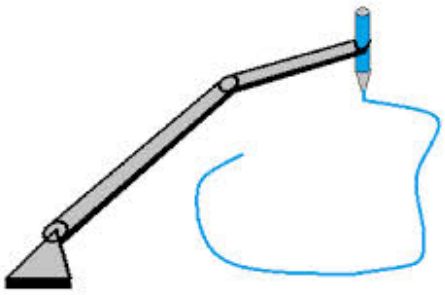


Figure 3-1: *Configuration space of 2 links robot*

this motion planning will be supported with a feedback control mechanism to correct stability exceptions [78].

The IKP solver we introduce is general and in each iteration corrects each joint that brings the output of the iteration closer to the desired solution. The solver uses linearization by the Jacobian matrix, but it includes some constraints that increase the convergence rate. This is an iterative process that is repeated until the solution convergences up to a certain $\epsilon$, or until there is an indication that it cannot be converged. Also, the IKP geometric equations are solved under constraints modeling dynamics considerations.

While robot motion planning in the configuration space is continuous, planning in the Cartesian space should be discrete. Therefore, a central challenge in robot motion planning is to transform the problem of planning the motion of a dynamic object into the problem of planning the motion of a point or a set of points.

We use the following terminology in defining the motion planning problem:

- The configuration $\mathbf{q}$ of a certain point on robot $A$ is specification of the position $T$ and the orientation $\Theta$ of the relevant frame of $A$ with respect to the work-frame.

- the **configuration space** of $A$ is the space $C$ of all the configurations of $A$.

- Path of $A$ from the configuration $q_{init}$ to the configuration $q_{goal}$ is the continuous mapping function $\tau : [0, 1]$ with $\tau(0) = q_{init}$ and $\tau(1) = q_{goal}$.

The configuration space of all robot configurations in terms of world coordinate system is described in Figure 3-1. The end-effector frame of $A$ as a function of the configuration $q$ is commonly represented by the Denavit-Hartenberg matrices.

In 1955 Denavit and Hartenberg introduced a representation of coordinate frames for spatial linkages sometimes called D-H. It is most useful for describing kinematics chains, since it reduces the number of parameters in any case that two links are perpendicular to each other, and it simplifies the configuration space representation. Any robot structure can be represented by this form, even if there are few kinematics chain in parallel or serial order. This representation is not limited to a certain robot structure or bounded by the number of joints.

In order to demonstrate the D-H convention, we will use a manipulator which is the simplest robot structure as described in Figure 4-1. It is important to mention that this representation is flexible and can be applied to any robot structure. A robot manipulator is composed of a set of links connected together by various joints. The joints can either be very simple, such as a revolute or prismatic joint or even more complex. Under the assumption that each joint has a single degree-of-freedom, the action of each joint can be described by a single real number: the angle of rotation in the case of a revolute joint or the displacement in the case of a prismatic joint. The objective of forward kinematic analysis is to determine the cumulative effect of the entire set of joint variables.

A robot manipulator with $n$ joints has $n+1$ links, since each joint connects two links. The joints are numbered from 1 to $n$, and the links indices are ranged from 0 to $n$. The location of joint $i$ is fixed with respect to $i-1^{th}$ link. The joint variable $q_i$ is associated with the $i^{th}$ joint if it is revolute, otherwise will be denoted as $d_i$. $A_i$ is the homogeneous transformation matrix expressing the position and orientation of the $i^{th}$ link with respect to link $i-1$. This matrix is a variable of the configuration variable $q_i$. Denote the homogeneous transformation matrix representing the position and orientation of link $j$ with respect to link $i$ as $T_{ji}$, then

$T_{ji} = A_{i+1}A_{i+2}\cdots A_{j-1}A_j \qquad if \qquad i < j$

Denote the position and orientation of the end-effector with respect to the base frame by a three-vector $O_n$ and the $3X3$ rotation matrix $R_n$ . Define the homogeneous transformation matrix of the end effector

$H = \begin{pmatrix} R_n^0 & O_n^0 \\ 0 & 1 \end{pmatrix}$

Or,

$H = T_n^0 = A_1(q_1)\cdots A_n(q_n)$

Each matrix $A_i$ is associated with four parameters:

- $a_i$ - link length
- $d_i$ - link offset (in relative to the previous link)

- $\alpha_i$ - link twist (in relative to the previous link)
- $\theta_i$ - the joint angle. (Similar to $q_i$ but not necessary identical. There might be a difference in their offset or direction)

Then $A_i$ in its general form looks like:
$A_i = R_{z,\measuredangle i} Trans_{z,d_i} Trans_{x,a_i} R_{x,\alpha_i}$
Where,

$$R_{z,\measuredangle i} = \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$Trans_{z,d_i} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$Trans_{x,a_i} = \begin{pmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_{x,\alpha_i} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Calculating the forward kinematics is done straight forward by substituting the relevant parameters in each matrix $A_i$, and multiply consequently those matrices. The inverse kinematics is much more complex, since there might be few valid solutions, and their calculation is not trivial. Moreover, the inverse kinematics is solved by applying two types of constraints:

- holonomic constraints - 6 DoF equations as a function of the time $t$. These constraints reduce the dimension of the configuration space.
- non-holonomic constraints. These are more difficult constraints to cope with since they do not reduce the dimension of the configuration space. These are equations on the configuration space itself or its derivatives.

The IKP (Inverse Kinematics Problem) can be defined as how to translate the Cartesian coordinate of the end effector (or target location) into the robot joints

configuration space. This is a redundant problem, that might have several valid solutions. So, actually, it covers the sub problem of selecting the best solution among all the feasible solutions.

Motion planning is common in the context of manipulators that do not have to account for stability, since they are mounted to the ground or wall. Indeed, any planning method that is based on the robot's geometry without considering the dynamics of the robot is not satisfied in humanoid robots [23]. Also, stability is less crucial in other devices that are inherently more stable, such as wheeled or quadruped robots. Their motion can be planned efficiently by using IKP solvers. The advantage of using the IKP is that it is planning a wide range of motions, even those outside the scope of a given CPG. It also enables whole body movement. An IKP solver determines the joints values prior to motion generation [57]. Dealing with manipulators, this method is efficient due to the sequential structure of the robots, the relative low number of joints and the anchoring point of the robot which stabilizes the manipulator [92].

In humanoids, this method is no longer effective since their structure is quite complex (mix of serial and parallel joints), large number of joints and the stability is not guaranteed. So, IKP is rarely used in humanoids, and whenever it is used it is just for partial body motions that do not affect its stability much, such as grasping tasks.

In this dissertation we propose an algorithm for off-line quasi static motion planning without relying on real-time sensory inputs. The approach we take is to extend existing stability criteria (for biped standing, walking and running), to whole body stability. We show in Chapter 4 how to turn these into equations that are in the general form of the IKP. Unfortunately, existing IKP solvers are unable to successfully solve it.

In Chapter 5, we present a novel IKP solver that can solve the kinematic equations under given constraints (e.g. stability constraints, optimization functions, etc.). This solver is not restricted to a certain robot structure or a family of robots; it can solve any given IKP with as many joints and multiple kinematic chains, both serial and parallel. This allows solving the IKP for many robot posture which do not necessarily supported by foot. Such postures can be sitting, laying, etc.

The downside of this solver is that while it is sound it is incomplete. Therefore, in Chapter 5 we empirically demonstrate that the solver is able to handle many common humanoid motions. In Chapter 6 we demonstrate how the solver accounts for constraints and especially for stability constraints.

# Chapter 4

# Multiple IKP: A Novel General Solver

The Inverse Kinematics Problem (IKP) is a central problem in robotics. IKP solutions map target positions from the Cartesian space to the robot joints space. There exist efficient algorithms that work well for robot manipulators that are mounted and have relatively small number of joints. However, Humanoids often have more than 22 joints divided into several kinematic chains that can be both parallel or serial. Existing IKP algorithms do not work well, or at all, in such settings.

There are three different situations that might occur with the IKP. It might be that there are either no solution at all, finitely-many solutions, or an infinite number of solutions. In general, this depends on the relation between the number of constraints and the number of variables. If the number of constraints equals the number of variables, then there will be numerous solutions. If the number of variables is larger than the number of constraints, there might be infinite number of solutions and if it is smaller than the number of constraints, there is no solution at all. But, there might be exceptions, in case that there are constraints that are not relevant (outside the feasible range), or there might be constraints that are depended on other constraints.

Usually, the IKP in humanoids is under-constrained, and has many degrees of freedom which results infinite number of solutions. There are just few solutions that are reasonable, those solutions have to perform smooth and stable motions and to consume not too much energy.

Any solution method for solving IKP on humanoid robots has to cope with the following problems:

- Finding at least one solution within the set of valid solutions, or choosing an

optimal solution within all valid solutions

- Solving a set of non-linear equations that are polynomials of transcendental functions.
- Converge into a solution within a relatively short time (depends on application).
- Obey different kinds of constraints, such as some optimization functions, boundary constraints, differential constraints, etc.

In this chapter, I present a general inverse kinematic solver. First, I formulate the humanoid IKP (Section 4.1), and discuss its characteristics (Section 4.2). Then, I present an algorithm for solving this problem (Section 4.3). Some improvements of the algorithm, its complexity, soundness and completeness analysis, as well as some results are presented in Chapter 5.

## 4.1 The Problem

The standard definition of IKP is presented in this section. Then, we explain the limitations of this definition while dealing with humanoids or other complex robots. Finally, a new general formulation of the IKP that is applicable for motions that involve several varied "mounting" points with few targets in each query.

### 4.1.1 The standard IKP definition

The IKP refers to mapping the task space to the joints space. The forward kinematics of a robot is the resultant $4 \times 4$ matrix achieved by multiplying the chain of D-H matrices of the robot. The target of the end-effector is given in the world coordinate system. Then, the IKP is defined as finding a set of robot joints values, $\bar{\Theta}$, that solves the forward kinematics equations. i.e., substituting $\bar{\Theta}$ in the forward kinematic equations of a given robot, will result the $4 \times 4$ target frame $F_t$. Assume that $R_0$, is the $4 \times 4$ reference frame of the robot in relative to the world is given.

A formal definition of the IKP is described in [84]. Let $f : \Theta \in \Re^n \rightarrow SE(3)$ represent the forward kinematics map of a kinematic chain. $SE(3)$ represents the Euclidean group for translation of a rigid body. In other words, given the values of $n$ joints variables $f$ returns the position and orientation of the end effector. The inverse kinematics problem can be stated as follows: given $G \in SE(3)$ find $\Theta \in \Re^n$ such that $f(\Theta) = G$ or determine that no solution is possible. The problem is rewritten by using homogeneous matrices.

Find $\Theta$ such that

$$\Pi_{i=1}^{n} A_i(\theta_i) = G$$

Where,

$$A_i(\theta_i), G \in SE(3)$$
$$A_i(\theta_i) = \begin{bmatrix} R(\theta_i) & p \\ 0 & 1 \end{bmatrix}$$

Since $G$ defines six constraints, the problem is well posed only if the number of independent joint variables is equal to 6.

Six different assumptions are embedded in this definition. None of them is valid while dealing with humanoid.

1. The robot has only one anchor point
2. The anchor point is constant
3. There is just one target point
4. The target point refers the end-effector
5. The robot is constructed of a unique serial kinematic chain
6. The number of variables (joints) equal to the number of constraints.

The support surfaces of the robot are analogous to the anchoring point of a robot manipulator. But humanoids might have several support surfaces in each posture and they vary between postures. The number of joints is usually much larger than the number of constraints. A typical posture has several supports and might have few targets relating different parts of the robot body, not necessarily the end-effectors. Due to all these limitations, the traditional definition of the IKP is not relevant in the case of humanoids. Therefore, an extended definition is required.

### 4.1.2 Formulation of Multiple IKP

To be applicable to humanoids, the original IKP is modified. We extend this definition to have several target points, each represented by a frame $\bar{F}_t$. Each target point is defined by a 2-tuple of entities. The first is the reaching point on the robot (relative

to the reference point of the robot), and the second is the target position (in the world coordinate system).

Similar to the traditional IKP, the output of the problem is the joints values $\bar{\Theta}$ that satisfies the kinematic constraints. But unlike the original problem, there might be several supports and few targets in the extended problem. Each pair of support and target generates an additional kinematic constraint. In contradiction to the original problem that the support (mounting point) is constant, in the extended problem it is varied. Therefore, we will distinguish between two phases of the problem. The first is the setup phase, that occurs once in a session, and the second, is a frequent query that is performed for each new posture. So, we will separate the inputs according the partition of the problem: (i) configuration parameters and (ii) IKP queries.

The configuration parameters are set once for a given robot in the stage of the IKP solver setup. The second type of inputs are frequently sent to the solver for each new movement segment.

There are two configuration inputs:

1. A reference frame of the robot's reference point relative to the world coordinate system. Usually, it is the frame represented the pelvis locomotion. This is a $4\times4$ matrix, representing 6 independent parameters of position and orientation, $T_r$.

2. The set of the D-H (Denavit-Hartenberg) matrices representing the robot kinematic structure. Each D-H matrix $A_{c,j}$ represents the location of link $j$ in chain $c$ in the kinematic structure. The number of D-H matrices equals the number of joints.

Each D-H matrix has the following form:

$$A_{c,j} = \begin{pmatrix} \cos\theta_{c,j} & -\sin\theta_{c,j}\cos\alpha_{c,j} & \sin\theta_{c,j}\sin\alpha_{c,j} & a_{c,j}\cos\theta_{c,j} \\ \sin\theta_{c,j} & \cos\theta_{c,j}\cos\alpha_{c,j} & -\cos\theta_{c,j}\sin\alpha_{c,j} & a_{c,j}\sin\theta_{c,j} \\ 0 & \sin\alpha_{c,j} & \cos\alpha_{c,j} & d_{c,j} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Where $\alpha_{c,j}$, $a_{c,j}$ and $d_{c,j}$ are constants of a robot and $\theta c, j$ is a variable, which is the required solution of the IKP.

Figure 4-1 describes a partial serial manipulator with 2 links. Therefore, each parameter has just one index, represents its order within the kinematic chain. Since this robot has a unique kinematic chain, the index of the chain $c$ is ignorable. As can be seen in Figure 4-1, the parameter $a$ is the length of the common normal (in the case of a revolute joint, this is the radius about the previous $z$-axis). $\alpha$ is the angle

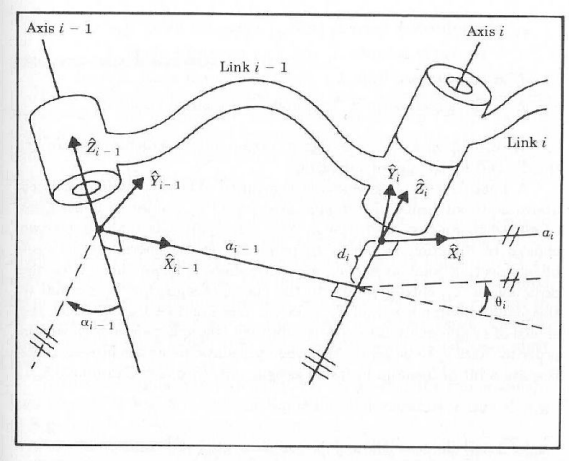



Figure 4-1: *The Denavit-Hartenberg convention. Construction of the link frame ( [90]). The parameters representing the relations between 2 consequential joints and links.*

about common normal, from old $z$-axis to new $z$-axis. The parameter $d$ is the offset along previous $z$-axis to the common normal.

Finally, the variable $\theta$ is the angle about the previous link $z$-axis, from old $x$-axis to new $x$-axis.

A typical humanoid robot consists of five kinematic chains: neck, left and right arm, left and right legs. Each of these consists of multiple joints. An example description of a humanoid robot, the Nao robot by Aldebaran, is detailed in appendix A. The kinematics is based on the structure of the Nao robot, but with general links size.

After the robot configuration parameters are set, the second stage can be made. This is a query for an IKP solution for given target. The inputs for these queries should be:

- The current joints values, $\bar{\Theta}_0$, an $n$ length vector
- The set of supported points $\bar{S}_p$
- The set of target points $\bar{T}_p$

Each supported point and target point is actually a pair of elements (a 2 tuple):

- The point on the robot that has to reach the target. The point is defined by the link index and the offset from its zero point ($4 \times 4$ matrix)
- $4 \times 4$ target frame (in world coordinate system)

Although both the supports ($\bar{S}_p$) and the targets ($\bar{T}_p$) are usually surfaces, they will be represented by a point, for the sake of simplicity.

The output, as was mentioned above is a set of joints value $\Theta$ that satisfies the kinematic equations at the target points.

Notice that each pair of support point and a target implies a kinematic constraint, which is represented by an equation. If two supports are on the same chain, then the kinematic constraint corresponds the two supports will replace one of the constraints represent the relation between a support and a target.

A support point is a point that supports the robot. The robot is in static equilibrium when the sum of all forces around all the support points equals zero. Support points have two important properties, the first is geometric and the second concerns the robot dynamics.

- The position of the support point does not change along the motion
- The infinitesimal sum of the different forces in these points is zero.

The position of each support point can be determined by substituting the initial joint values, $\bar{\Theta}_0$ in the relevant kinematic chain, $c$. The point might be in any place along the chain, not necessary at the end effector. Also, the point is not necessary at the reference point of the link. Its relative place is defined by the transformation and translation $4 \times 4$ matrix $T_{c,j}$. Notice that this support point position is determined in the robot coordinate system and has to be transformed into the world coordinate system.

$S_p(i_s)$ is the $i^{th}$ supported point within $\bar{S}_p$, the vector of all supporting points. Assume there are $N_{sp}$ supporting points, then the positions of a supporting point $i_s$ located on the $j^{th}$ link within the chain $c$ can be presented as:

$$S_p(i_s) = T_r A_{c,1}(\bar{\Theta}) \cdots A_{c,j}(\bar{\Theta}) T_{c,j} \tag{4.1}$$

A condition on the target point $T_p(i_T)$ demands that a point on the $i_T^{th}$ link in chain $c_T$ will reach this point. Notice that the point is not necessary located at the origin of the link. It might be transformed by $T_{c,i_T}$ in relative to the link origin.

$$T_r A_{c_T,1}(\bar{\Theta}) \cdots A_{c_T,i_T}(\bar{\Theta}) T_{c,i_T} = T_p(i_T) \tag{4.2}$$

Recall that manipulators have anchoring point, which is usually used as the reference point of the robot. Therefore, the robot reference is given and there is no need to determine it. In humanoids, the supports are analogous to this anchoring point.

Since they are changing along the motion, this reference has to be repeatedly determined. Its calculation is performed by combining equations 4.1 (a specific supported point) and 4.2 (a specific target point) into whole kinematic chain, as described by 4.3.

$$(T_r A_{c,1}(\bar{\Theta}) \cdots A_{c,j}(\bar{\Theta}) T_{c,j} T_p(i_T))^{-1} T_r A_{c_T,1}(\bar{\Theta}) \cdots A_{c_T,i_T}(\bar{\Theta}) T_{c,i_T} = (S_p(i_s))^{-1} T_p(i_T) \tag{4.3}$$

The left hand side of the equations above is a function of the variables $\bar{\Theta}$ and the right hand side is a given parameter.

The number of such kinematic equations is multiplication of the number of supports, $N_{sp}$, and the number of target points, $N_t$. There are cases that the support and the target are in different chains, and in the same kinematic chain. Equation 4.3 holds in both cases. Three constraints are derived from each such kinematic equation, for the $x$, $y$ and $z$ coordinates.

The basic inverse kinematics problem is to find a set $\bar{\Theta}$ that satisfies this set of equations. Notice that the problem in its general form is under-constrained, which means that there are infinite number of solutions. Also, in its raw form, there is no definition of some additional geometric constraints such as self collisions and physical constraints such as keeping the robot stable.

Here is a summary of the formal input and output specification. The inputs are:

- $C$ - number of kinematic chains
- $N_c$ - number of joints in each chain
- Set of the robot D-H matrices $A_{c,j}$
- $R_0$ - the reference frame of the robot in relative to the world
- $\bar{S}_p$ - the set of support points
- $\bar{T}_p$ - the set of target points
- $\bar{\Theta_0}$ - the current joint values
- $\Delta\bar{\Theta}$ - the amount of change in the joint values in the current iteration

The output is $\bar{\Theta}$, the values of the joints angles in the target position ($\bar{T}_p$).

## 4.2 Characteristics of the problem

In order to better understand the nature of such equations, let us examine some example cases. A sample equation for $x$ coordinate where the initial pose of the robot is in its zero position, its support is the left foot and the target is a given position of the right palm. The parameters of the links' length were selected to be identical to those of the NAO Robot [62]. The equation is presented in appendix B.

Although this example of $x$ as a function of the joints variables $\bar{\Theta}$ is a specific case of robot structure in a certain posture, the characteristic of the function is representative of most of the kinematic functions of humanoids. Moreover, the posture was selected to demonstrate the problematical issues of humanoids at their extent, i.e. the longest kinematic combined chain involves coarse and fine motions links. Therefore, the analysis of the kinematic function is demonstrated on this specific function, but it represents the kinematic properties in general.

Actually, each kinematic constraint is a function representing a coordinate, either $x$, $y$ or $z$, as a function of $\bar{\Theta}$, the joints values. This is an $n$–dimensional function. Later on, we propose an algorithm that solves the IKP, which is a set of equations (constraints) of $n$ variables. The algorithm is based on a linearization method. First, we have to investigate the characteristic of this family of constraints, in order to be convinced that linearization is an appropriated approach in this case.

A constraint is the equation representing the forward kinematics of a certain chain that starts in a support point and ends in a target point. We examined many constraints in different postures in all the three coordinates (each of them generates one constraint), and we concluded that they are polynomials of the chain's length order. These are polynomials of transcendental functions (i.e. the sin and cos functions). In order to illustrate the behavior of such functions and to justify the use of linearization, we pick one such function which represents the whole family. We demonstrate the properties of this $n$–variables functions relatively to each variable separately. The function that was selected represents a constraint of the $x$ coordinate. The $y$ coordinate constraints are similar while the constraints relating $z$, are usually much simpler. The kinematic chain that was selected in this example is the longest possible in humanoid. In this chain, the support is in the left foot and the target is in the right palm, so the total number of joints is 12.

Linearization methods are applicable to functions that have no critical points or only a few of them within an interval. The following study demonstrates that although a kinematic constraint might have several critical points, it is piecewise linear between

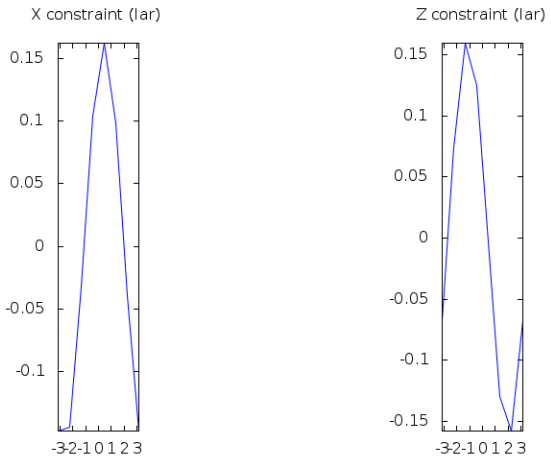



Figure 4-2: *X, Y, Z Kinematic equations as a function of the left ankle roll (lar) angle. Each function has up to 2 extremum points within the joint's interval. The maximal impact is on Z and it is bounded by 0.30 cm. It is the same order of impact on X. The effect of lar on Y is negligent.*

two consequent such points. Therefore, by using linearization it should find a solution if such exist within a local proximity.

We developed the kinematic constraints by substituting the values of the zero position, each time for all variables except one. Figures 4-2–4-15 represent a kinematic equation of a standing posture as a function of each variable $\theta_{c,i}$ (i.e. all joints except the variable are zero).

Figure 4-2 demonstrates the behavior of the kinematic function of end effector coordinates, i.e. the right palm, as a function of the left ankle roll (lar) angle, while all other joints are constant in their zero-value. The *lar* joint is varied within the interval $[-3\pi, 3\pi]$ (the horizontal axis). The vertical axis represents the transition of the end-effector.

The function of the X coordinate behaves like a parabola within this interval, and its peak is at the value of $lar = 0.5$ radians. The function has just one extremal point within the feasible interval, which is its peak value. If the major axis of the foot is parallel to the X-coordinate, then the Y-coordinate is not affected by changing the *lar* joint. The difference between the minimum and maximum values the can be generated by this joint in coordinate $X$ is 0.30 cm. The Kinematic equation of the Z-coordinate as a function of the *lar* joint, as demonstrated in the right figure has two extremal points (minimum and maximum) within the interval. These points are symmetric to the zero value. The maximal impact of changing this joint on $Z$ is 0.30

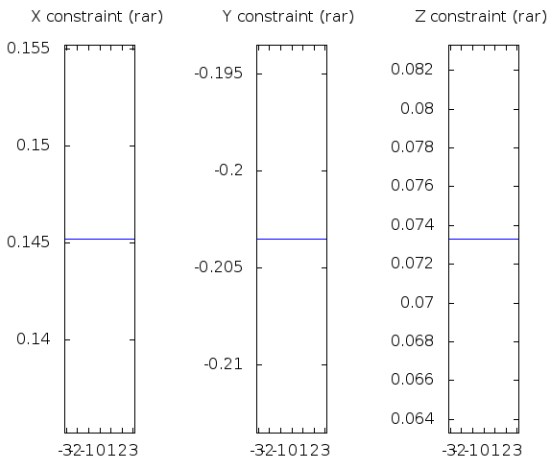



Figure 4-3: *X, Y, Z Kinematic equations as a function of the right ankle roll angle, which has no impact on the target*

cm, which is the same as the order on $X$. This means that within a finite number of steps the coordinate function will converge into a solution if one exists.

Since the right leg does not affect the position of the right palm in a case that the support is the left leg and there are no other constraints, the graphs of the right leg joints (rkp, rhp, rhr and rhy) look like Figures 4-3 and 4-5, as horizontal line.

It should be emphasized that the resolution of the vertical axis changes from graph to graph. The impact of the hip yaw angle (Figure 4-9) is very significant in relative to the other joints.

It is obvious that the joints that are not located along the kinematic chain represented by a given constraint have no influence on the target position. In our example, the kinematic chain starts at the left foot and ends at the right palm. This means that all the joints belong to the right leg and the left hand have no influence on the target position. It is clearly seen in Figure 4-10, for example, which represents how change of the left shoulder pitch affects the position of the right palm. The horizontal line in all three coordinates shows that there is no effect at all. The right hand's joints affect the position of the end-effector of the right palm, as can be concluded from Figures 4-11 through 4-15.

The solution of this set of equations is equivalent to finding the zero of the overall functions. The functions behave as typical transcendental functions; this means that any iterative method that is based on linearization might converge into a solution. Such methods are Newton-Raphson or using the Jacobian matrix.

Different problems might arise with these methods: First, the convergence and

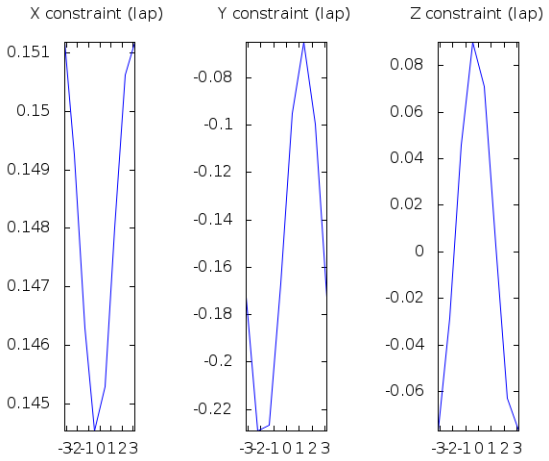



Figure 4-4: *X, Y, Z Kinematic equations as a function of the left ankle pitch*

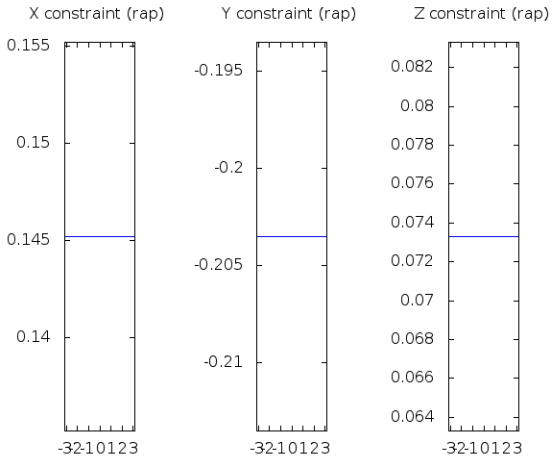



Figure 4-5: *The X, Y, Z of the right palm as a kinematic function of the right ankle pitch*

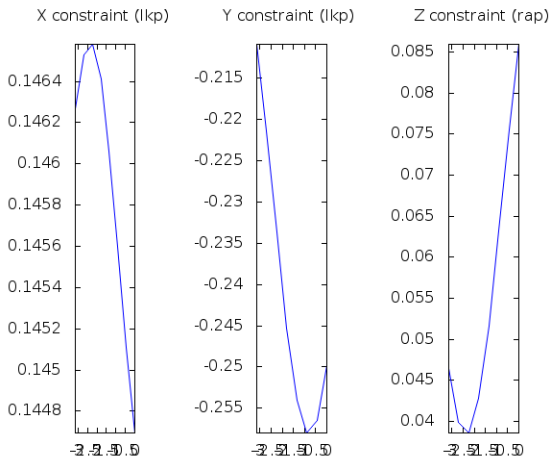



Figure 4-6: *X, Y, Z kinematic function as a function of left knee pitch*

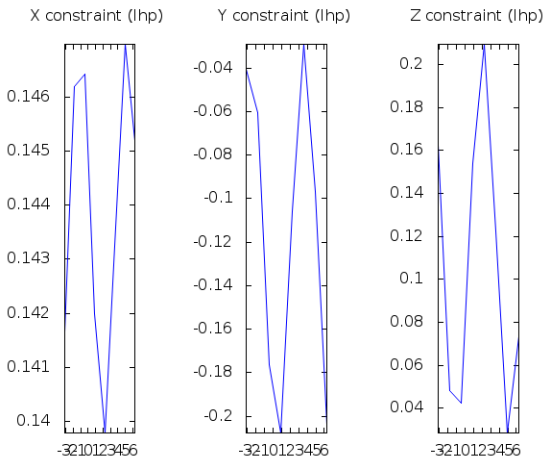



Figure 4-7: *X, Y, Z kinematic function as a function of left hip pitch*

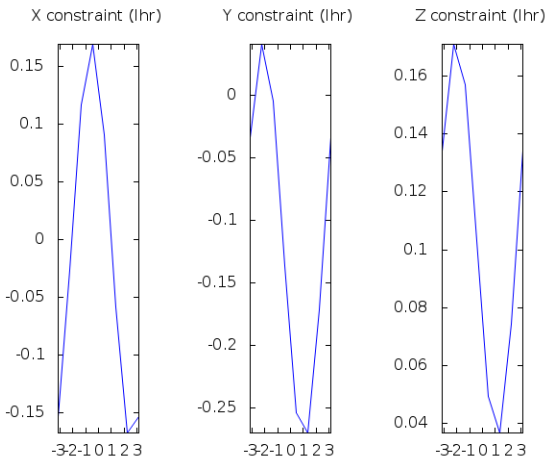



Figure 4-8: *X, Y, Z kinematic function as a function of left hip roll*

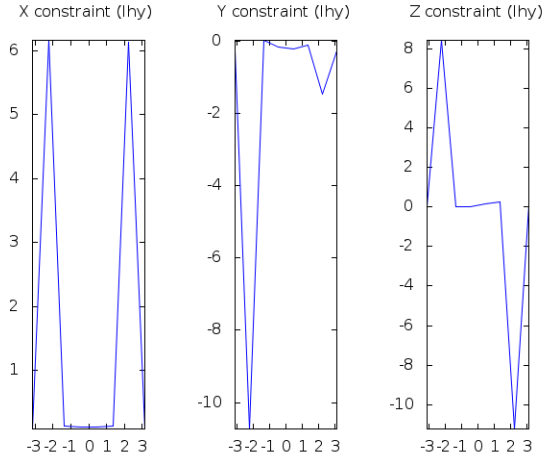



Figure 4-9: *X, Y, Z kinematic function as a function of left hip yaw*

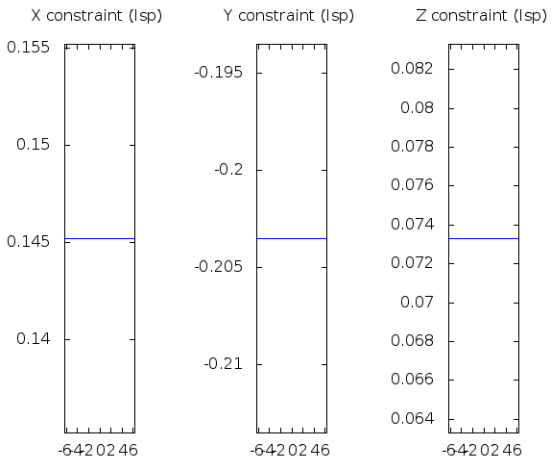



Figure 4-10: *The kinematic function of the right palm as a function of the left shoulder pitch.*

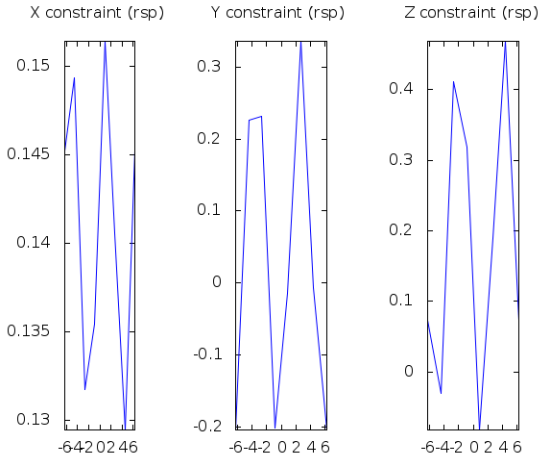



Figure 4-11: *The kinematic function of the right palm as a function of the right shoulder pitch.*

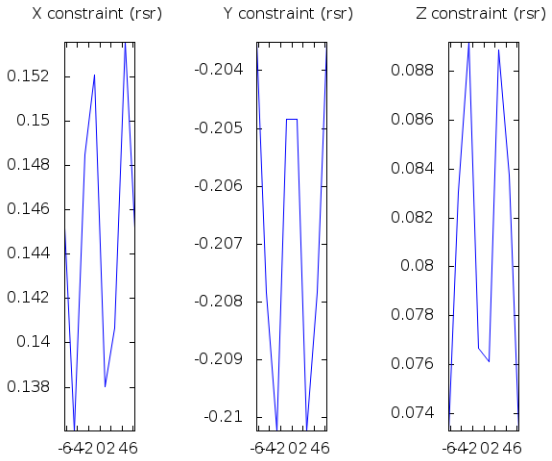



Figure 4-12: *The X, Y, Z equations of the right palm as a function of the shoulder roll*

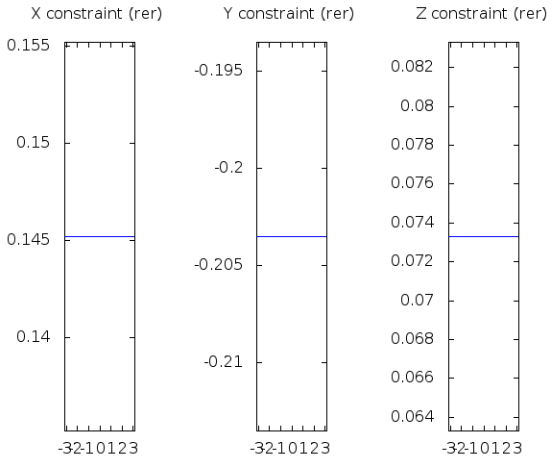



Figure 4-13: *The kinematic function of the right palm as a function of the right elbow roll angle*

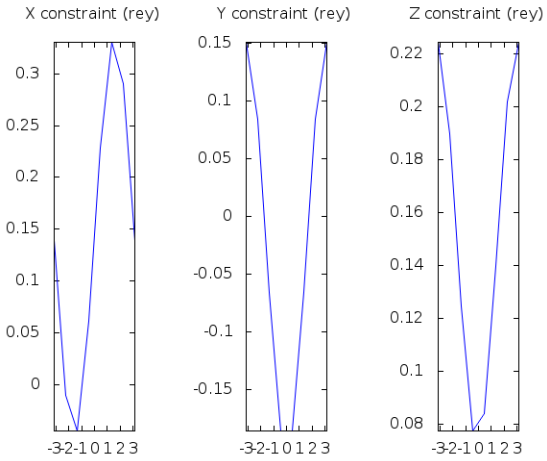



Figure 4-14: *X, Y and Z functions of the right palm as a function of the right elbow yaw*

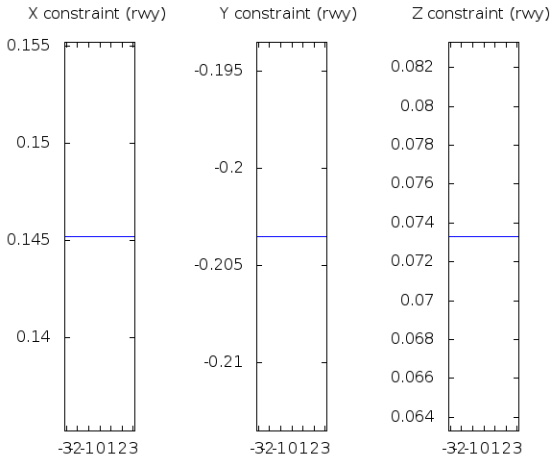



Figure 4-15: *X, Y and Z functions of the right palm as a function of the right wrist yaw*

its rate are dependent on the initial state of the joints. It might be that in a certain case the process will not converge into a solution, or will converge too slowly. Also, implying an external constraint equation on the kinematic set of equations, might convoluted. Convoluted functions cannot be solved simultaneously, especially not by linearization. In the next section, a general IKP algorithm is proposed which we will show can be extended to tackle each of the above problems.

## 4.3 The Solution to the Multiple IKP

The Multiple Inverse Kinematics Problem is usually an under-constrained problem, with infinite number of solutions. There are just few that are valid among these solutions. The transition between the current posture and the target posture should be smooth and reasonable in terms of the energy, or more intuitively, the magnitude of joints change. For example, consider a case of an extended arm that is in a distance of $1_{mm}$ from an item that has to be reached by a humanoid. The optimal motion is to change a bit the shoulder pitch joint. But there are infinite number possible motions by changing all the joints of the hand so that one joint compensates on the motion of the others. Moreover, most solutions do not preserve stability; this will be discussed in detail in Chapter 6.

The proposed algorithm finds one solution within those possible. This solution is often the closest to the current posture, but this cannot be guaranteed.

The proposed method is based on the Jacobian matrix of the partial derivatives of the kinematics equations. First, we will define the set of equations. Assume, there are $N_{sp}$ support points and $N_t$ target points. Then, there are $N_{sp} \times N_t$ kinematic constraints. Each kinematic constraint is composed of three equations for $X$, $Y$ and $Z$ coordinate. The total number of kinematic equations are $3N_{sp}\ N_t$. Each equation is a polynomial of transcendental function. In its general form this is an under-constrained problem that has infinite number of solutions. In order to limit the number of solutions, some constraints might be implied on the kinematic set of equations. This is discussed in Chapter 6

The Jacobian $J$, the partial derivative matrix of all equations in relative to each variable, has $3N_{sp} \times N_t$ rows and its number of columns is $N$, which is the number of joints. The kinematic constraints (Eq. 4.3) will be partially derived in order to construct $J$.

$$\bar{E} = \bar{T}_{support} T^{-1}_{k-link_i} A^{-1}_{ki} \dots A^{-1}_{k0} A_{jo} \dots A_{ji} T_{link_i} - \bar{T}_{target} \tag{4.4}$$

$$J(\bar{\Theta}) = \frac{\partial \bar{E}(\bar{\Theta})}{\partial \bar{\Theta}} \tag{4.5}$$

The solution presented here is based on a numerical approach that linearizes the set of equations and finds the solution that has minimal deviation from the target. Assume the current value of the joints angles is $\Theta_0$. Define the vector of the residuals, i.e, the difference between the actual values of the equations and the desired value to be:

$$\bar{L} = E(\bar{\Theta}_0) - J(\bar{\Theta}_0)\Delta\bar{\Theta} \tag{4.6}$$

The method is based on calculation of the corrections vector of each joint ($\bar{V}$). Notice that since J is not necessarily a square matrix, the pseudo inverse is calculated rather than using its inverse:

$$\bar{V} = (J^T J)^{-1} J^T \bar{L} \tag{4.7}$$

If the kinematic Equations $\bar{E}$ were linear, then the solution of the kinematic equations would be:

$$\bar{\Theta} = \bar{\Theta}_0 + \bar{V} \tag{4.8}$$

But the linearization is rough in the case of these transcendental kinematic functions. Due to this limitation, the process of correcting $\Theta$ should be repeated iterative. In its general fashion, Equation 4.8 should be rewritten as:

$$\bar{\Theta}_{i+1} = \bar{\Theta}_i + \bar{V}_i \tag{4.9}$$

The iterative process repeats solving Equations (4.7) and (4.9) until a certain threshold of convergence is achieved.

Figures 4-4 through 4-15 demonstrate that the equations of the coordinates have at least one solution within the range of the relevant joints. The magnitude of change is varied between the different joints, so there are joints that impact the position much more than others. Therefore, there is a good chance that the set of kinematic equations will converge into a solution by using linearization, after few iterations.

The basic algorithm is an iterative solution of the linearized set of equations, i.e. iterating through Eq. (4.9). The improvements aim to increase the rate of convergence and cope with ill-conditioned situations of divergence. In a case of convergence, the algorithm always converges to a valid solution (See Proof in 5.6). The improved algorithms vary in their complexity and in the success rate in different domains. The strategy is to execute the basic algorithm with some additional calculations that indicate whether to proceed with one of the improved algorithm and if so, with which of them. If none of the improved algorithms converges, then, there is a high probability that no solution exists. Such a case might be when some constraints are dependent on each other, or that the target position is not reachable.

### The Multiple IKP Algorithm

Algorithm 1 applies the minimum least square approach to the set of kinematic equations. This algorithm is based on minimization of the forward kinematic deviation under linearization of both the forward kinematic equations and the constraints.

Recall that $\bar{J}$ is the Jacobian matrix of the left hand side of Equation 4.4. Let $\bar{\theta_0}$ be the joints values at the current (initial) position, and $L$ will be the vector of the residuals as defined in Equation (4.6).

The input of the algorithm is its structure, its joints' state in its origin posture, its targets and the support points that actually generate the stability constraints. There are optional other constraints. In lines $1-3$ the setup configuration is determined. Lines $4-6$ are the commands for the iterative loop initialization. The core of the initialization is performed iterative in lines $8-13$, which locally solve the kinematic equations and constraints by using linearization.

The output of this procedure is $\Theta$, the joint values that reach the targets. The main iterative loop might terminate due to two reason, either the iterative process converged and the norm of the residuals is below a certain threshold, or the number of iterations is too high. In the later case, it might be that there is no solution at all, or at least the procedure cannot converge into a solution. In the case of the former, the process terminates successfully and the required IKP solution is achieved. Notice that in this case a solution is found but it is not necessarily unique.

The convergence indication is determined in line 12 of algorithm 1. This is a norm of the joints' correction $\bar{V}$. The norm, $\Delta = \sqrt{\frac{V^T V}{N}}$ is the root of the averaged square. This criterion is in the joints' configuration space. There could be another norm, such as the averaged absolute value of the corrections, $\frac{\sum |v_i|}{N}$. Also, the norm can be applied to a different criterion such as the deviation of the target in terms of world

**Algorithm 1** Basic Constrained IKP algorithm

**Require:**

$C$ - number of kinematic chains
$N_c$ - number of joints in each joint
Set of the robot D-H matrices $A_{c,j}$
$R_0$ - the reference frame of the robot in relative to the world
$\bar{S}_p$ - the set of support points
$\bar{T}_p$ - the set of target points
$C^\ell$ - the set of additional constraints
$\Theta_0$ - the current joint values
MaxIterations - maximum number of iterations

1: Calculate the $3 \times |\bar{S}_p| \times |\bar{T}_p|$ kinematic constraints
2: Calculate $\bar{E}$ (Section 4.3)
3: Calculate the Jacobian $J$ (Eq. 4.5)
4: $\Theta \leftarrow \Theta_0$
5: $\Delta \leftarrow \infty$
6: $i \leftarrow 0$
7: **repeat**
8: $J_s \leftarrow J(\Theta)$
9: $L \leftarrow RHS(\bar{E}) - J(\bar{\Theta}_0)$ (Eq. 4.6)
10: Calculate $V$ (Eq. 4.7)
11: $\Delta \leftarrow \sqrt{\frac{V^T V}{N}}$
12: $\bar{\Theta} \leftarrow \bar{\Theta} + V$
13: $i \leftarrow i + 1$
14: **until** $\Delta < \epsilon$ or $i >$ MaxIterations
15: **if** $i >$ MaxIterations **then** print "No Solution"
16: **else return** $(\Theta)$

coordinates. The advantage of this criterion is that it is in Cartesian coordinates, which makes it more intuitive.

This algorithm suffers from the following drawbacks:

- It does not necessarily converge into a proper solution
- The convergence process might be slow
- The solution is an arbitrary one among the existing solutions.
- In any case that it does not converge into a solution it is not clear whether no solution exists or it could not find the solution
- There is no prioritization of the constraints, which means that all constraints have the same weight. Sometimes there are more crucial constraints that the solution must obey them more accurately than others.

In order to address the above disadvantages we will propose in the next chapter some improvements to the basic multiple IKP algorithm. The first improvement proposes a method to choose the initial joint values prior to begin the convergence process. The second, third and fourth improvements are two methods to divide the convergence process into two sub-processes in order to accelerate the convergence rate and to find solutions in case they exist and the basic algorithm does not find them. The fifth algorithm proposes a preprocessing stage to get some indications about the nature of the problem and as a consequence choose the relevant improvement within the above ones.

### 4.3.1 Soundness

Whenever the algorithm converges, it converges into a valid solution. This is proved by contradiction.

**Theorem 1.** *Algorithm 1 is sound. If a solution is found then it is valid.*

*Proof.* Assume that the algorithm stopped after $i_{itr}$ iterations and the set of joints values achieved at the end of the process is $\Theta_i$. The loop might stop due to 2 different reasons:

- The number of iterations is beyond the allowed maximal number.
- The norm is smaller than $\epsilon$.

The first case, means that the algorithm did not converge into a solution.

We will prove that in the second case, the process always converge into a valid solution.

Assume for contradiction that $\Theta_i$ is not a valid solution to the set of equations $E$ (4.4). Then, it should be that also the norm is smaller than a given $\epsilon$, at least one of the joint value is not a valid one. This means that the following equation should be satisfied:

$$\sum_j |RHS(E_{j,i}(\theta_i)) - LHS(E_{j,i}(\theta_i))| < \epsilon \tag{4.10}$$

Let's examine the value of $\epsilon$, while determining a norm. The norm is an aggregated value of the absolute deviations of each equation $e_j$ within $E$. In one extent, it might be that there is one equation that deviates from $\epsilon$ and all the rest do not deviate at all. On the other extent there might be that if there are $|E|$ equations, each of them deviates in $\frac{\epsilon}{|E|}$. Assume that $\epsilon$ was chosen to satisfy the first case which is stricter.

Then, there should be at least one equation within $E$, that is not satisfied, i.e.:

$$\begin{aligned} &RHS(e_j(\Theta_i)) - \epsilon > LHS(e_j(\Theta_i)) \\ &Or, \\ &RHS(e_j(\Theta_i)) + \epsilon < LHS(e_j(\Theta_i)) \end{aligned} \tag{4.11}$$

But this contradicts the existence of equation 4.10. Therefore, $\Theta_i$ should satisfy the set of equation $E$.

□

Notice that this proof is valid if the aggregated norm equals the maximal allowed deviation of each equation. This is quite strict assumption, that can be modified by adjusting the norm of the aggregated error. The norm should be determined by the error propagation analysis which is discussed in Chapter 8.

Alternatively, one can use a combination of norms to ensure the validity of the solution under less strict norms.

### 4.3.2 Completeness

Algorithm 1 is not complete, i.e. there might be existing solution that is not discovered by the algorithm. In this section, we discuss some strategies to increase the probability of finding the solution, if such exists.

The proposed constrained IKP algorithm is based on linearization of the set of kinematic conditions and constraints, $E$. This set is non-linear. Dealing with the kinematic constraints, they behave as oscillating functions with a certain number of peaks and a bounded magnitude (Figures 4-2– 4-15). The impact of stratification all relevant variables (i.e. multi variable condition) should be carefully investigated. Let us examine the mutually impact of two variables. It can be either of the following:

1. They are of the same order of magnitude and wave and oscillation length (like waves, but not necessarily all oscillations are of the same size). Their peaks are similar.

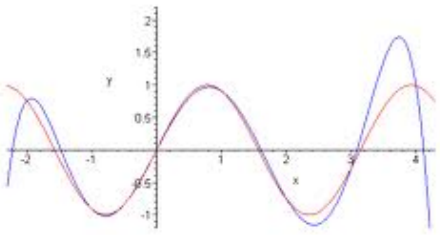
2. They are of the same order of magnitude and wave and oscillation length. Their initial value is approximately the same. Their peaks are opposite

3. They are of the same order of magnitude and wave and oscillation length. Their initial value is approximately the same. Their peaks are shifted at about half of the oscillation width.

4. They are of different order of magnitude.

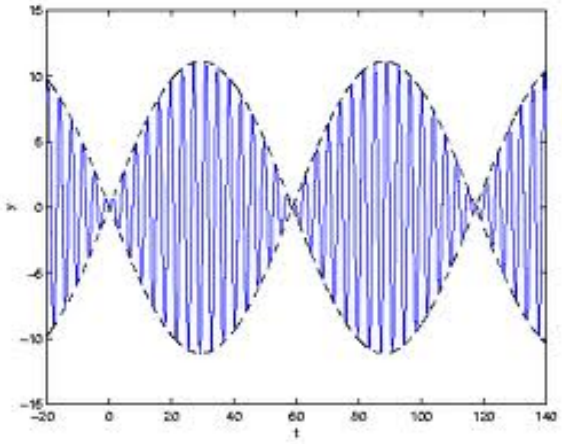
Relating the first case (see Figure 4-16), the resultant function is an oscillating function with the same amplitude with a bigger magnitude (the sum of the two magnitudes). In that case, searching for a solution is equivalent to searching in one variable function, which is piecewise linear, and the number of "pieces" is bounded by the reasonable low integer (e.g. 8) in the case of humanoids. (In other robots it is even a smaller number).

Figure 4-16: Two Oscillating functions with the same order of magnitude and amplitude



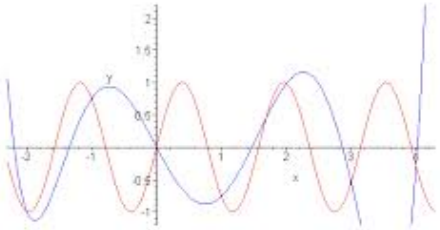

The second case, as is demonstrated in Figure 4-17, is problematic, since the two functions are convoluted. In that case, there might be that there is no solution.

Figure 4-17: Two Oscillating functions with the same order of magnitude and opposite amplitude



The resultant function of the third case (Figure 4-18) is an oscillating function with more oscillations (higher frequency). Again, this function is close to be linear piecewisely, but the size of these "pieces" is smaller than in the original functions. So again, the linearization method should work in high probability.

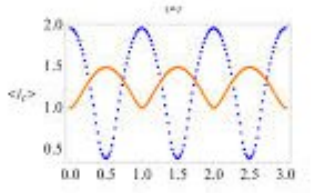
Figure 4-18: Two Oscillating functions with the same order of magnitude and shifted amplitudes



The last case, where the magnitudes of the functions is of different order (Fig. 4-19) is also piecewise linear. The composition of such two functions might cause any linearization method to alternate between converge-divergence trends.

Notice that this rough explanation refers to the composition of two functions only. In the case of a kinematic structure we deal with much more functions (e.g. in the case of humanoid - there are 26 functions). We can repeat this analysis for any number of functions. Moreover, in typical humanoids, there are no more than 3 coupled variables. The improved algorithms in Chapter 5 increase the probability of convergence.

Figure 4-19: Two Oscillating functions with the same order of magnitude and shifted amplitudes




Basically, linearization methods can be applied to piecewise linear functions. But, not all the solutions will be discovered. Therefore, in order to find more solutions, or to avoid cases that the process alternates between convergence and divergence in a certain proximity, there can be implemented a sequential search algorithm to find high probability solvable "zones" on the solution envelope. This, of course, can be implemented only in off-line planning tasks.

# Chapter 5

# Addressing the Incompleteness of MIKP

This chapter addresses the weaknesses in the basis algorithm (Alg. 1) proposed in the previous chapter. The following improvements will be applied to the basis algorithm:

- The initial joint values should be selected carefully in order to make sure the process is converged and moreover, converge into the desired solution
- A multi staged process will enable relating different levels of constraints
- A multi staged process enables accelerating the convergence
- A multi level process enables convergence skipping convolutions that are caused by coupling and convoluted constraints.
- An indication whether a solution exists and if so, what is its proximity.

## 5.1 First Improvement

This algorithm is embedded in a function that is executed before line 4 in Algorithm 1. The algorithm calculates the set of initial values of the joints, $\Theta$. In the basic algorithm, the initial values relate the initial posture of the current robot value. Since the kinematic constraints are polynomials of the transcendental functions sin and cos, they are twisted, which means that the function has few extermum points for each variable. Since the basic algorithm is based on linearization of these functions, there will be obtained one solution at the most. It is not necessarily the best solution and it might not be achieved at all. Choosing an initial values of the variables that are in the proximity of the solution might increase the probability of achieving a solution, ensuring that the achieved solution is an appropriate one and even accelerating the convergence rate.

The approach is similar to the CPG family of algorithms. We will define a library of common postures and their corresponding joints values. For example, standing with straight legs and hands down is defined as the "zero postures" where all joints are zero. Other common postures are sitting, whole body leaning down, fetus posture, etc. As a preprocessing stage, before the iterative loop of the basic algorithm, the value of $\Theta_0$ will be determined so as the values of the posture that is reaching the target most closely. The disadvantage of the algorithm is that it aims to predict proximity in the Cartesian space by changing the proximity in the joint space without defining the precise relation between the spaces.

Algorithm 2 in additional to the input of the basic algorithm is the library of CPG postures. The output is $\Theta_0$, the initial values of the joints. Notice that the constant of maximal iterations number is not required.

## 5.2 Second Improvement

There are some cases whose part of the constraints are more crucial than others. The solution is numerical and it is valid up to a certain precision. It might be

**Algorithm 2** Finding the initial value of $\Theta$

**Require:**

1: $\bar{E}$ - the set of kinematic constraints that are determined in the basic algorithm
2: $\Theta_0$ - the current joint values
3: CPGlib - the predefined library of common postures and their corresponding joints values (lookup table)

4: **function** FIND$\Theta_0(E, \Theta_0, CPGlib)$
5: $\quad\lambda \leftarrow \infty$
6: $\quad$**for all** $\Theta_i \in CPGlib$ **do**
7: $\quad\quad V_t \leftarrow RHS(E) - LHS(E(\Theta_i))$
8: $\quad\quad \lambda_t \leftarrow \sqrt{\frac{V_t^T V_t}{|V_t|}}$
9: $\quad\quad$**if** $\lambda_t < \lambda$ **then**
10: $\quad\quad\quad \lambda \leftarrow \lambda_t$
11: $\quad\quad\quad i_\Theta \leftarrow i$
12: $\quad V_t \leftarrow RHS(E) - LHS(E(\Theta_0))$
13: $\quad \lambda_t \leftarrow \sqrt{\frac{V_t^T V_t}{|V_t|}}$
14: $\quad$**if** $\lambda_t < \lambda$ **then return** $\Theta_0$
15: $\quad$**else** $\quad$ **return** $\Theta_{i_\Theta}$

that the precision of satisfying the constraints is not identical in all equations. For example, some targets have to be reached precisely while others might be reached roughly. Another example is that a certain target should be reached precisely in one coordinate (e.g. $Z$), while it is enough to reach the proximity of the target in the other coordinates ($X$ and $Y$).

The basic algorithm grants equal weight to all equations in $E$. The following algorithm divides the set of equations $E$, which includes the kinematic constraints into $M$ subgroups, $E_1, \cdots E_m$, ordered by their significance from the highest to the lowest. It is obvious that $M \leq |E|$. Now, the basic algorithm becomes multistage, as described in 3. Notice that the input and output are identical to those of the basic algorithm and improvement 1 can be applied in the same manner.

Line 13 in the algorithm requires explanation. The priority of each equation in $E$, is an integer number ranged $[1..M]$. There are different way to predetermined this value. It might be an input of the user. It might be determined by the type of constraint that was also predefined by the user. Since there are few ways to determine it, but all of them are based on some user input, we will not get into these details in the framework of the algorithm description.

This algorithm belongs to a family that in opposite to the basic algorithm, that solves all the equations simultaneously. The equations are partitioned into sub groups

**Algorithm 3** Multi Stage Constrained IKP algorithm

**Require:**

$C$ - number of kinematic chains
$N_c$ - number of joints in each joint
Set of the robot D-H matrices $A_{c,j}$
$R_0$ - the reference frame of the robot in relative to the world
$\bar{S}_p$ - the set of support points
$\bar{T}_p$ - the set of target points
$C^\ell$ - the set of additional constraints
$\Theta_0$ - the current joint values
MaxIterations - maximum number of iterations

Calculate the $3 \times |\bar{S}_p| \times |\bar{T}_p|$ kinematic constraints
Calculate $\bar{E}$ (equation 4.4)
$M \leftarrow 0$
**for all** $e_i \in \bar{E}$ **do**
    determine $p_i$ ▷ Determine the priority of each equation in $E$
    **if** $p_i > M$ **then** $M \leftarrow p_i$
**for** $1 \leq j \leq M$ **do**
    $E_G \leftarrow \{e_i | \quad \forall i, \quad p_i \leq j\}$
    Calculate the Jacobian $J$ of the set $E_G$(Eq. 4.5)
    $\Theta \leftarrow \Theta_0$
    $\Delta \leftarrow \infty$
    $i \leftarrow 0$
    **repeat**
        $J_s \leftarrow J(\Theta)$
        $L$ (Eq. 4.6)
        Calculate $V$ (Eq. 4.7)
        the new joint values $\Theta$ (Eq. 4.9)
        $\Delta \leftarrow \sqrt{\frac{V^T V}{N}}$
        $\Theta \leftarrow \Theta + V$
        $i \leftarrow i + 1$
    **until** $\Delta < \epsilon$ or $i > MaxIterations$
    **if** $i > MaxIterations$ **then**
        terminate with no solution
    **else** $\Theta_0 \leftarrow \Theta$
**return** $(\Theta)$

according to a certain criterion. In its extent, this algorithm solves each equation at a time (schematic description in 4). $\bar{J}$ and $\bar{L}$ are defined as in Eq. 4.5 and 4.6.

**Algorithm 4** Another version of the Multi Stage Constrained IKP algorithm. In each iteration only one kinematic equation is solved

**Require:**

$C$ - number of kinematic chains
$N_c$ - number of joints in each joint
Set of the robot D-H matrices $A_{c,j}$
$R_0$ - the reference frame of the robot in relative to the world
$\bar{S}_p$ - the set of support points
$\bar{T}_p$ - the set of target points
$C^\ell$ - the set of additional constraints
$\Theta_0$ - the current joint values
MaxIterations - maximum number of iterations

Calculate the $3 \times |\bar{S}_p| \times |\bar{T}_p|$ kinematic constraints
Calculate $\bar{E}$ (equation 4.4)
$M \leftarrow 0$
$\Theta \leftarrow \Theta_0$
**repeat**
    $\bar{M} \leftarrow \bar{J}^T \bar{J}$
    $k \leftarrow j \quad | \max_j(M_{j,j})$
    $\theta_j \leftarrow \theta_j + (\bar{J_j}^T(\bar{\theta})\bar{J}(\bar{\theta}))^{-1}\bar{J_j}^T(\bar{\theta})\bar{L}$
**until** $|\bar{L}| < \epsilon$ or $i > MaxIterations$

Algorithm 4 is based on minimization of the forward kinematic deviation and the constraints under linearization of both types of equations. This algorithm weights $\theta_i's$ differently than the first algorithm. This might find the solution of the whole system in a different localization than the first algorithm does.

## 5.3 Third Improvement

This improvement chooses the most effective joints in every iteration, and the set of equations is solved just for them. Every iteration the process of choosing the joints is repeated.

In contrast to the second improvement, that every iteration selects a set of rows in the Jacobian, the current improvement selects a set of columns within the Jacobian and solves them. In other words, the second improvement solves all the variables for part of the equations in each iteration while the current algorithm solves part of the variables for all the equations.

The advantage of this approach is that it avoids the coupling difficulties as well as problematic constraints that convoluted with other equations. Its disadvantage is that it might be slower due to more iterations and a complex preprocessing procedure.

It is based on the basic algorithm with an extension procedure that selects the three variables that advancing the robot the most towards reaching the targets. Thereafter, the Jacobian is updated to include only the three corresponding columns. From here on, the iteration proceeds the same as in the basic Algorithm 1. This approach might be very useful for some postures while for the others it might be inefficient. In general, the robot structure contains some links for coarse motions (e.g. thigh, arm, etc.) and some links, or even more precisely, some joints that tend to perform the fine motions (e.g. foot, wrist, etc.). In motions such as reaching an object that is located in front of the robot in the height of it upper body, the proposed algorithm will be most efficient. But if for example, the initial pose is some scrolled posture and the target position extremely downward or upward. The details are described in Algorithm 5.

Notice that the first improvement of choosing an appropriate initial solution will hold here in the same way as in the basic algorithm. The inputs and output are the same as in the basic Algorithm 1.

The third improvement is actually selects those variables that contribute the most to the convergence of the next iteration, according to a given norm (line 11).

There might be different versions of selecting the group of joints to be solved in a certain iteration. All these versions belong to the same family of solvers. Below is another example of an algorithm of this family.

Algorithm 6 is another variation of the previous algorithm. It also solves the variables sequentially. The only difference is in the criteria to choose the variable that is solved in each iteration. Here, the criteria is to take the variable that its variance covariance is maximal. i.e. the one that affects the other variables, the most. The complexity, soundness and completeness analysis are the same as in the second algorithm.

## 5.4 Fourth Improvement

This improvement, like the previous two improvements, solves partial problem at each iteration. But, in contrast to these that select the most significant parts of the problem, in the following algorithm the partial set variables (joints) to be solved is predefined.

**Algorithm 5** Improved Multiple IKP algorithm (3) - I/O

**Require:**

$C$ - number of kinematic chains
$N_c$ - number of joints in each joint
Set of the robot D-H matrices $A_{c,j}$
$R_0$ - the reference frame of the robot in relative to the world
$\bar{S}_p$ - the set of support points
$\bar{T}_p$ - the set of target points
$\Theta_0$ - the current joint values
MaxIterations - maximum number of iterations

**Ensure:**

$\Theta$ - the joint values at the target

Calculate the $3 \times |\bar{S}_p| \times |\bar{T}_p|$ kinematic constraints
Calculate $\bar{E}$ (equation 4.4)
Calculate the Jacobian $J$ (Eq. 4.5)
$\Theta \leftarrow \Theta_0$
$\Delta \leftarrow \infty$
$i \leftarrow 0$
$i_1 \leftarrow i_2 \leftarrow i_3 \leftarrow 0$
$norm_1 \leftarrow norm_2 \leftarrow norm_3 \leftarrow \infty$
**repeat**
    **for** $1 \leq i \leq N$ **do**
        $norm \leftarrow \sum_j |RHS(E_{j,i}) - LHS(E_{j,i}(\theta_i))|$
        **if** $norm < norm_1$ **then**
            $norm_3 \leftarrow norm_2$;  $norm_2 \leftarrow norm_1$;  $norm_1 \leftarrow norm$
            $i_3 \leftarrow i_2$;  $i_2 \leftarrow i_1$;  $i_1 \leftarrow i$
        **else if** $norm < norm_2$ **then**
            $norm_3 \leftarrow norm_2$
            $norm_2 \leftarrow norm$
            $i_3 \leftarrow i_2$
            $i_2 \leftarrow i$
        **else**
            **if** $norm < norm_3$ **then**
                $norm_3 \leftarrow norm$
                $i_3 \leftarrow i$
    **for** $1 \leq k \leq N$ **do**
        $J_s \leftarrow \{J_{*,k}| \quad \forall k, \quad k = \{i_1|i_2|i_3\}$
    Calculate $L$ (Eq. 4.6)
    Calculate $V$ (Eq. 4.7)
    Calculate the new joint values $\Theta$ (Eq. 4.9)
    $\Delta \leftarrow \sqrt{\frac{V^T V}{N}}$
    $\bar{\Theta} \leftarrow \bar{\Theta} + V$
    $i \leftarrow i + 1$
**until** $\Delta < \epsilon$ or $i > MaxIterations$
**if** $i > MaxIterations$ **then** print "No Solution"
**else** **return** $(\Theta)$

**Algorithm 6** Improved Multiple IKP algorithm (3) - sequential solution of the kinematic equations. At each iteration the variable to be solved is the one that generates the maximal deviation

**Require:**

$C$ - number of kinematic chains
$N_c$ - number of joints in each joint
Set of the robot D-H matrices $A_{c,j}$
$R_0$ - the reference frame of the robot in relative to the world
$\bar{S}_p$ - the set of support points
$\bar{T}_p$ - the set of target points
$\Theta_0$ - the current joint values
MaxIterations - maximum number of iterations

Calculate the $3 \times |\bar{S}_p| \times |\bar{T}_p|$ kinematic constraints
Calculate $\bar{E}$ (equation 4.4)
Calculate the Jacobian $J$ (Eq. 4.5)
$\Theta \leftarrow \Theta_0$
Calculate the $3 \times |\bar{S}_p| \times |\bar{T}_p|$ kinematic constraints
Calculate $\bar{E}$ (equation 4.4)
$M \leftarrow 0$
$\Theta \leftarrow \Theta_0$
**repeat**
    $\bar{M} \leftarrow \bar{J}^T \bar{J}$
    **for all** $j \in |V|$ **do**
        $v_j \leftarrow \sum_i |J_{i,j}|$
    $k \leftarrow j \quad | \max_j(v_j)$
    $\theta_j \leftarrow \theta_j + (\bar{J_j}^T(\bar{\theta})\bar{J}(\bar{\theta}))^{-1}\bar{J_j}^T(\bar{\theta})\bar{L}$
**until** $|\bar{L}| < \epsilon$ or $i > MaxIterations$

A robot has a certain number of chains, and each of them impacts significantly on different parts of the "reaching envelope". The key in this fourth improvement is that at each iteration the problem will be solved just for one chain, i.e, only the relevant columns of the Jacobian $J$ will be selected, standing for the partial derivatives of the relevant joints.

The input and output are the same as those of the basic multiple IKP algorithm (Algorithm 1). In the proposed algorithm the inputs $C$, $N_c$ and $A_{c,j}$ are used not only to construct the kinematic equations, but also to select the relevant chain in each iteration.

Lines 1-3 are initialization of the variables that describe the robot's initial kinematic state. In lines 4-8 some variables of the iterative process are initialized. Lines 10-27 are the core of the iterative loop. Notice the sign $\bowtie$ in line 12 that stands for algebraic join of two vectors (i.e. their union).

The advantage of this algorithm is its fast convergence (proof in Section 5.6), and reduction in the impact of the convolution. The disadvantage of this method is that there are some constraints that will hardly be satisfied (e.g. whole body stability) and in some "ill-conditioned" postures it will not converge into a solution (e.g. Knee-to-Chin press posture, as described in Figure 5-1).

Figure 5-1: Knee-to-Chin press posture [1]

## 5.5 Tying it all together: MIKP*

Last improvement of the basic constrained IKP algorithm, takes an algorithm selection approach, that tries to analyze the IKP problem before choosing the best method for the current specific problem [41, 42].

It checks which of the improvements (Algorithms 2-7) is most efficient in the current phase. This improvement should be applied in a case that the basic algorithm with its previous improvement does not converge into a solution. The complexity of

**Algorithm 7** Improved Multiple IKP algorithm

**Require:**

$C$ - number of kinematic chains
$N_c$ - number of joints in each joint
Set of the robot D-H matrices $A_{c,j}$
$R_0$ - the reference frame of the robot in relative to the world
$\bar{S}_p$ - the set of support points
$\bar{T}_p$ - the set of target points
$\Theta_0$ - the current joint values
MaxIterations - maximum number of iterations

Calculate the $3 \times |\bar{S}_p| \times |\bar{T}_p|$ kinematic constraints
Calculate $\bar{E}$ (equation 4.4)
Calculate the Jacobian $J$ (Eq. 4.5)
$\Theta \leftarrow \Theta_0$
$\Delta \leftarrow \infty$
$i \leftarrow 0$
$norm_t \leftarrow \infty$
$J_{st} \leftarrow \emptyset$
**repeat**
    $J_s \leftarrow J(\Theta)$
    **for all** $1 \leq j \leq C$ **do**
        $J_{st} \leftarrow J_{st} \bowtie \{J_s(*, j) | \forall j \in kin_j\}$ ▷ $J_{st}$ contains only the columns of the joints in the relevant kinematic chain
        $L \leftarrow \bar{E} - J_{st}(\Theta_0)$
        $V \leftarrow (J_{st}^T J_{st})^{-1} J_{st}^T L$
        $\Theta_t \leftarrow \Theta + V$
        $L \leftarrow \bar{E} - J_{st}(\Theta_t)$
        $norm \leftarrow \sqrt{L^T L}$
        **if** $norm < norm_t$ **then**
            $norm_t \leftarrow norm$
            $i_{kin} \leftarrow j$
    $J_s \leftarrow \{J_s(*, j) | \forall j \in kin_{i_{kin}}\}$
    $L \leftarrow \bar{E} - J_s(\Theta)$
    $V \leftarrow (J_s^T J_s)^{-1} J_s^T L$
    $\Theta \leftarrow \Theta + V$
    $\Delta \leftarrow \sqrt{\frac{V^T V}{N}}$
    $\bar{\Theta} \leftarrow \bar{\Theta} + V$
    $i \leftarrow i + 1$
**until** $\Delta < \epsilon$ or $i > MaxIterations$
**if** $i > MaxIterations$ **then** return with "No Solution"
**else** **return** $(\Theta)$

this algorithm is high, but it selects reliably the set of equations to be solved by the IKP.

This improvement composes two independent functions, each of them checks another property of the equations set to be solved, and according to its findings, a solving strategy is determined.

The first function searches for a reasonable initial value for $\Theta$, so that the constrained IKP algorithm 1 will converge into a solution in high probability. As could be seen in Figures 4-2 through 4-15, each function does not have more than 8 extremum points. The composition of $N$ such functions might lead to $8^N$ extreme points. But actually, since the extremes of the different functions is of varied values, the general analysis can ignore those with smaller impact. Notice, that in some border cases, this assumption will fail. However, if we partition the range of the solutions into $I_{int}$ number of intervals, and testing the proximity of each interval to find if there exists a solution, there is a high probability that a solution will be found. Notice that usually, there might be more than one solution. In most cases there will be 2-8 different solutions. This search algorithm will hopefully find some of them.

The input is similar to the input of the basic Algorithm 1, except the maximum number of iterations and $\Theta_0$ that are not required. The output is an initial value $\Theta_i$. In its linear proximity, there is a solution to the constrained IKP (Algorithm 8). Below is the proof of this crucial claim.

**Theorem 1.** *If there exists a solution to the IKP within a linear proximity of $\Theta_0$, and $\Theta_0$ is used as an initial value of the joints vector, then Algorithm*1 *will converge to this solution.*

*Proof.* Assume that there is a solution within a given interval. Let's reduce the problem to two variables only, $(v_1, v_2)$. Later we will expand the problem again.

Assume that the number of solutions within the physical range of each of these variables is $m_1$ and $m_2$, correspondingly. Then, there might be three different cases:

- There is no overlapping between the solution ranges of the variables.
- There is partial overlapping between the solution ranges of the two variables.
- All the solutions of one variable resides within the solution range of the other variable.

In the first case, there is no solution and therefore the algorithm will not find one. In the third case, there are few solutions of one joint within a linear proximity of

the other variable. So, actually, there is one solution that resides within the same sub-range. In the second case, there might be that the current linear proximity of one variable is different from the other. But, since there exists at least one common linear proximity for both variables, within a finite number of trials, it will be found.

Now, let's expand the problem to $n$ variables. Two functions of two different variables are convoluted into another function, that might have more twisting points within the valid range. So, the $n$ different functions are actually one convoluted function. Since all kinematic functions and constraints of all variables are transcendental trigonometric, their convolution can be predicted.

For the other case, where there is no solution within the given interval, it is straightforward that either of the two occurs: (i) the function never achieve the zero value or (ii) the zero's of the different variables, are in disjoint intervals. In both cases the algorithm will not converge into a solution because none valid solution exists.

□

The second function checks which of the improvements (Alg. 3–7) will be most effective in the next iteration. Actually, each of these four algorithms (and their sub versions) selects a sub-problem to be solved in the next iteration. Recall that Alg. 3 selects part of the equations (rows), Alg. 5 selects part of the variables (columns) and Alg. 7 selects one chain within the robot, which is actually also selection of part of the columns. In order to maximize the effective of the convergence, if the solver will choose the improvement that affects the next iteration the most, the overall process will be improved. This can be done by calling the four improvements and comparing their results. Then, executing the one that achieved the best result, as the next iteration.

It is obvious that the performance of such a comparison is slow. Therefore, instead of running the algorithms themselves, there is an option to run some indicators of the behavior of the Jacobian matrix ($J$), but the reliability of this approach is lower than running the improved algorithms.

The complete multiple IKP solver (Alg. 9) calls two improvement indications proposed above and calls the relevant improved Algorithm 2- 5.

The performance of each iteration of the improved multiple IKP solver is much slower than of the other algorithms, but the number of iterations is decreased. The advantage of this algorithm is most cases that the other algorithms diverge due to variables coupling or convoluted equation are resolved.

The last improvement (Alg. 10) is an algorithm that rather than solving the Multiple IKP, finds the zones on the solution envelope that are suspected as containing

**Algorithm 8** Finding a linear environment for the existence of a Constrained IKP solution

**Require:**

$C$ - number of kinematic chains
$N_c$ - number of joints in each joint
$A_{c,j}$ - Set of the robot D-H matrices
$R_0$ - the reference frame of the robot in relative to the world
$\bar{S}_p$ - the set of support points
$\bar{T}_p$ - the set of target points

Calculate the $3 \times |\bar{S}_p| \times |\bar{T}_p|$ kinematic constraints
Calculate $\bar{E}$ (equation 4.4)
Calculate the Jacobian $J$ (Eq. 4.5)
**for** $1 < i < N$ **do**
    $\theta_{i_a} \leftarrow \theta_{i_{min}}$
    $\theta_{i_b} \leftarrow \theta_{i_{max}}$
    $J_{col} \leftarrow J_{*,i}$
    $i_{itr} \leftarrow 0$
    **repeat**
        $val_1 \leftarrow (J^T(\theta_{i_a})J(\theta_{i_a}))^{-1}J^T(\theta_{i_a})L$
        $val_2 \leftarrow (J^T(\theta_{i_b})J(\theta_{i_b}))^{-1}J^T(\theta_{i_b})L$
        $i_{itr} \leftarrow i_{itr} + 1$
        **if** $(val_1 \times val_2 \geq 0)$ **then**
            $\theta_{i_a} \leftarrow \theta_{i_a} + \frac{\theta_{i_{max}} - \theta_{i_{min}}}{128}$
            $\theta_{i_b} \leftarrow \theta_{i_b} - \frac{\theta_{i_{max}} - \theta_{i_{min}}}{128}$
    **until** $(val_1 \times val_2 < 0)$ *or* $(i_{itr} > 64)>$
**if** $i_{itr} > 64$ **then**
    print("No Interval")
**else**
    $\bar{\Theta} \leftarrow \{\theta_i\}$ **return** $(\Theta)$

**Algorithm 9** Improved Constrained IKP solver

**Require:**

$C$ - number of kinematic chains
$N_c$ - number of joints in each joint
Set of the robot D-H matrices $A_{c,j}$
$R_0$ - the reference frame of the robot in relative to the world
$\bar{S}_p$ - the set of support points
$\bar{T}_p$ - the set of target points
$\Theta_0$ - the current joint values

1: **repeat**
2: $\bar{\Theta}_1 \leftarrow$ Call algorithm1
3: $\bar{\Theta}_2 \leftarrow$ Call algorithm2
4: $\bar{\Theta}_3 \leftarrow$ Call algorithm3
5: $\bar{\Theta}_3 \leftarrow$ Call algorithm4
6: $i \leftarrow \bar{\Theta}_{imin}$ ▷ the index of the joint values set that achieves the minimal $norm_i$ of the accumulated deviation from the targets
7: Execute $algorithm_i$
8: **until** $norm_i < treshold$

a valid solution.

In each iteration it maps the configuration space into the Cartesian space to find the zone of the configuration space that has high probability to contain a solution. This algorithm has the highest rate of success among all algorithms, but its complexity is high, although it is polynomial (5.6).

Algorithm 10 can be used in combination with the others as an initial stage to find the initial values of $\theta_i's$, before starting the iterative process of finding the solution.

If Algorithm 10 was terminated then another set of initial values are set to $\bar{\theta}_0$ and the algorithm can be repeated. How to choose another set is described in the framework of the simulations and examples.

If the algorithm reached its end then the initial values of $\bar{\theta}_0$ for one of the previous algorithms will be set from the valid ranges that were found by executing this algorithm.

## 5.6 Analysis

The IKP algorithm and its improvements are iterative. The complexity of each iteration is polynomial. In the following sub-sections we will analyze the number of iterations as well as proving that the process will converge into a feasible solution.

**Algorithm 10** Improvement in finding the initial values of $\theta_i's$

**Require:**

$C$ - number of kinematic chains
$N_c$ - number of joints in each joint
Set of the robot D-H matrices $A_{c,j}$
$R_0$ - the reference frame of the robot in relative to the world
$\bar{S}_p$ - the set of support points
$\bar{T}_p$ - the set of target points
$\Theta_0$ - the current joint values

1: **for** each column $j$ in matrix $\bar{E}$ **do**
2: Initialize the ranges of the maximal of this variable.
3: **for** each row $i$ in matrix $\bar{E}$ **do**
4: Substitute the current values of $\theta_k$ for all $k \neq j$
5: Calculate the maximal points of the function and refine the ranges of this variable.
6: **if** the ranges of $\theta_j$ are $\ominus$ **then** then terminate

### Complexity Analysis

In this section we first analyze the complexity of one iteration of the basic multiple IKP solver (Algorithm 1). Then, the number of iterations is estimated. Finally, the effect of each improvement (Algorithms 2-10) on the overall complexity is analyzed. Before presenting the analysis some definitions are required:

- Number of variables (joints) is $N$
- Number of support points is $S_p$
- Number of target points is $T_p$
- The total number of kinematic constraints is $3 \times S_p \times T_p$

**An iteration Complexity** Each iteration performs commands 7-15 in the algorithm 1. These commands are actually construction of matrices and some manipulations on those matrices.

The size of the Jacobian matrix, $J$, is $N_r \cdot N$, where $N_r$ is the total number of rows. The number of columns is identical to the number of variables which is $N$. The number of rows equals to the total number of equations, which is the sum of the kinematic equations and the other constraints:

$$N_r = 3\ S_p\ T_p + C \tag{5.1}$$

Denote $N_r \cdot N$ as $N_j$, is the number of entries in $J$.

Each cell in $J$ is a polynomial of transcendental functions of the joints angles along the kinematic chain. The maximum degree of the polynomial equals the length of the kinematic chain. The maximum length of kinematic chain in the humanoid (from a toe to the palm), is 13, in the robot we use as model. Let denote the maximum kinematic chain length as $\ell_c$. The general polynomial structure takes the following form:

$$f_1^{\ell}(\theta_1, \cdots \theta_{\ell}) + \cdots + f_{\ell}^1(\theta_1, \cdots \theta_{\ell}) + C \tag{5.2}$$

Where each $f_i^{\ell}(\theta_1, \cdots \theta_{\ell})$ is of the following form:

$$C_1 \cdot \sin^{j_1}(\theta_{i_1}) \cos^{j_2}(\theta_{i_1}) \cdot \sin^{j_3}(\theta_{i_2}) \cos^{j_4}(\theta_{i_2}) \cdot \sin^{j_5}(\theta_{i_3}) \cos^{j_6}(\theta_{i_6}) + C_2 \tag{5.3}$$

Where, $C_1$ and $C_2$ are constants. $i_1$, $i_2$ and $i_3$ are integers in the range $[1..N]$ and, $j_1$, $j_2$, $j_3$, $j_4$, $j_5$ and $j_6$ are the power of the polynomial, i.e. they are integers in the range [0..6].

Therefore, given symbolic matrix $J$, the complexity of substituting the joints values $\bar{\Theta}$, (line 8 in algorithm 1) becomes:

$$\mathrm{O}\big((3 \cdot S_p \cdot T_p) \cdot N^4)\big) \tag{5.4}$$

In line 9, $L$ is calculated. This is a $N_r^t h$ length vector. The number of calculations of an item is actually the number of calculations in $J(\Theta_0)$. As mentioned above, this matrix has $N_r \cdot N$ entries. The maximal number of operations of calculating each entity is $N^2$. So, the complexity of line 9 becomes:

$$\mathrm{O}\big(N_r^2 \cdot N^3\big) \tag{5.5}$$

The analysis of line 10 complexity compounds of counting the number of operations in matrix inversion and matrix multiplications. As mentioned above, there are $N_r \cdot N$ entities in matrix $J$. So, the multiplication $J_T J$ requires $N^2 N_r$ operations. Inverting this multiplication is performed by $N^3$ steps. The multiplication of $J^T L$ requires $N_r N$ operations. Last multiplication of $(J^T J)^{-1}$ by $J^T L$ is preformed by $N^2$ operations. The overall complexity of calculation eq. 4.7 is:

$$\mathrm{O}\big(N^2 N_r + N^3 + N_r N + N^2\big) = \mathrm{O}\big((1+N)(N_r N + N^2)\big) \tag{5.6}$$

The complexity of each line 11 and 12 is $\mathrm{O}(N)$, and of lines 13 and 14 is $\mathrm{O}(1)$ (each). So, the overall complexity of one iteration is:

$$\mathrm{O}\big((3{\cdot}S_p{\cdot}T_p){\cdot}N^4){+}N_r^2{\cdot}N^3{+}(1{+}N)(N_rN{+}N^2){+}2N\big) = \mathrm{O}\big(N^2(N_rN^2{+}N{+}1)\big) \quad (5.7)$$

**Number of Iterations.** This part of the complexity analysis is difficult to determined since it is varied between different rates of convergence. However, we will distinguish between 3 different cases: (i) fast convergence (ii) slow convergence and (iii) divergence.

The first case is relevant if the initial values of $\Theta$ are in the linear proxy of all the equations. In this case the solution of each iteration is approaching towards the final solution (up to the predefined precision). This rate is bounded by the maximum value of:

$$\frac{f(\theta)}{f'(\theta)} \quad (5.8)$$

Practically, since there are $N_r$ functions ($f_i(\theta_j)$) and each of them has $N$ partial derivatives ($\frac{\partial f_i}{\partial \theta_j}$), we choose the strictest rate within the $N_r \times N$ different values.

The second case of slow convergence is relevant in the case that some variables $\theta_i$ are in their linear proximity while some are not, or they have different convergence directions. In this case there might be some phases of divergence before the final convergence into a valid solution. In order to analyze the complexity, the maximal number of such oscillations should be bounded. As can be seen in Figures 4-2 through 4-15, each joint might oscillate maximum 8 times in the interval of $[-\infty, +\infty]$. (Notice, that most joints are ranging in smaller intervals). In a case of two coupled joints of the same volume, the number of oscillation might reach $8^2$. Since, in the kinematic equations there are no more than 3 coupled variables of the same volume, we can roughly estimate the maximal oscillations as $8^3$. But there can be situations where a variable is alternating between two intervals, back and forth. In that case improvement 1 2 or 4 or any other improvement of that family will throw the joints value to another interval. In the case of an equation set that is not convoluted, this process will finally converge into a valid solution. Define $K_1$ as a constant, then the estimated number of iterations in the slow convergence case can be bound by $8^3K_1$. Notice that practically the number of iterations is bounded for a reasonable threshold, so that the process will always stop.

In the third case, where some equations are convoluted, there might be that a

solution cannot be found, and the number of iterations is bounded by a predetermined threshold. In a case that the number of iterations is reached that threshold without finding a solution, it can be assumed that no solution exists or another solving process can be executed to try bypassing the problem:

- Start the process with a new random initial joints values
- Perform a comprehensive analysis of the set of equations before choosing the proximity of the solution
- Reduce some equations within the set $E$
- Replace the problem with another one (different targets that are via points between the current location and the desired ones)

We leave investigation of these options to future work.

**Total Complexity.** The complexity of lines 1-6 and 15-16 in the basic constrained IKP algorithm is detailed below.

The complexity of calculating a kinematic constraint is analog to multiplying the D-H matrices. Each constraint compounds of up to 13 multiplications of $4\times4$ matrices which its number of operations is $\leq 13 \times 4^3$. Each entity in the D-H matrices is of $\mathrm{O}\big(1\big)$. So, the total complexity of lines 1 and 2 is:

$$\mathrm{O}\big(S_p \times T_p \times 13 \times 4^3\big) = \mathrm{O}\big(S_p \times T_p\big) \tag{5.9}$$

Notice that the 3 equations for coordinate $X$, $Y$ and $Z$ are achieved altogether. Considering line 3, the complexity of determining $J$, was analyzed before: $\mathrm{O}\big((3 \times S_p \times T_p) \times N^4)\big)$. The complexity of lines 4, 5, 15 and 16 is $\mathrm{O}\big(1\big)$.

Then, the overall complexity of the algorithm in the case of fast convergence is:

$$\mathrm{O}\big(\max(\frac{f(\theta)}{f'(\theta)}) \times N^2(N_r N^2 + N + 1) + N_r \times N^4\big) = \mathrm{O}\big(\max(\frac{f(\theta)}{f'(\theta)}) \times N_r \times N^4\big) \tag{5.10}$$

From the above equation, it is obvious that $N$, the number of joints, has the most significant effect on the complexity. Another important insight of the total complexity is that in a linear proximity, where $\max(\frac{f(\theta)}{f'(\theta)}$ is a constant, the convergence rate is mainly affected by the number of joints.

If the kinematic constraints are not dependent on a long chains, and if there are not many coupled variables, then the total complexity will be reduced by factors, and

the algorithm will be implementable in near real time (NRT) planning application. Otherwise, it can be used just for off-line path planning purposes.

In the case of slow convergence the complexity becomes much higher:

$$\mathrm{O}\big(8^3 K_1 \times \max(\frac{f(\theta)}{f'(\theta)}) \times N_r \times N^4\big) = \mathrm{O}\big(K_1 \times \max(\frac{f(\theta)}{f'(\theta)}) \times N_r \times N^4\big) \qquad (5.11)$$

But the ratio $\frac{f(\theta)}{f'(\theta)}$ is over the whole solution envelope, which is difficult to estimate.

Improvement 1, introduced by algorithm 2 improved the complexity by choosing the initial joint values that in their proximity the term $\max(\frac{f(\theta)}{f'(\theta)})$ is relatively small.

The effect of the second improvement, is that in its extent, each equation is solved solely, and the complexity of calculating $\bar{J}$ is $|\bar{E}| \times |\bar{\theta}|^2$. The complexity of calculating $\bar{L}$ is also $|\bar{E}| \times |\bar{\theta}|^2$.

The complexity of each iteration compounds of calculating the norm and the correction of $\theta_i$. The norm calculation is bounded by $|\bar{E}| \times |\bar{\theta}|^2$. The vectors multiplication is bounded by: $O(|\bar{\theta}|^2)$. So, the total complexity of an iteration bounds by $O(|\bar{\theta}|^3)$.

The total number of iterations depends on the convergence rate. However, it should be bigger than the analog parameter of the first algorithm. Therefore, the total complexity is $O(|\bar{\theta}|^3)$.

Improvement 3 and 4 as described in 5 and 7, might accelerate the convergence rate. The analysis of its impact on the complexity requires the use of algebraic geometry theorems and it is beyond the scope of this dissertation.

**Complexity Analysis** The complexity of Algorithm 10, which is an add-on to find the initial guess of the solution $\theta_i' s$ is high. The nested loops are preformed $|\bar{J}|$ times which is $|\bar{E}| \times |\bar{\theta}|$. The complexity of step 4 is $|\bar{\theta}|^2$. Step 5 is the tortuous part of the algorithm. This step itself is an iterative process of finding the roots of a high order transcendental function. This can be done by performing piecewise Newton-Raphson on the domain. There are other methods. However, their complexity depends on the function behavior (number of roots and their density) on one hand, and on the search resolution on the other hand. The higher the complexity, the higher the probability to find the solution if one exists. The complexity of step 5 will be denoted as $\Psi$.

Therefore, the total time complexity of algorithm (10) in a case that it does not terminate is $|\bar{E}| \times |\bar{\theta}| \times (|\bar{\theta}|^2 + \Psi)$.

In a case of termination, the algorithm can be repeated with a different set of

initial values $\bar{\theta}_0$.

### Soundness

Whenever Algorithm 1 converges, Algorithms 2 through 10 also converge. Theorem 1 is valid since none of the basic assumptions was changed. Moreover, Algorithms 2 through 10 are improvements of the envelope of the basic iterative loop of Algorithm 1, but they do not change its core.

### Completeness

Algorithms 2 through 10 are not complete, similarly to Algorithm 1. The improved algorithms increase the probability of convergence. The impact of each improved algorithm on the completeness is summarized below.

1. The first improvement of searching for a set of initial joints values increasing the probability of linear localization of the function. In a linear proximity, the method always converged into a solution.

2. The second improvement, of solving a sub-group of equations, reduces the probability of convoluted functions, which is "ill conditioned". So, it also increases the probability of convergence into a solution.

3. The third improvement, of solving a sub-group of the variables, reduces the probability of coupled variables, which also might prevent convergence.

4. The fourth improvement, like the third one uses only sub-group of the variables in each iteration, but this subgroup is predefined. Its impact on the completeness is identical to the impact of the third improved algorithm.

Algorithm 3 improves the probability of convergence into a solution in the case that the kinematic function and constraints are of the same orders of magnitude, wave, oscillation length, and their initial value is approximately the same, but, their peaks are opposite. This algorithm solves the problem just for sub-group of variables, and increases the probability of finding the solution since it is impossible that both convoluted variables will be chosen at the same iteration.

The fourth improvement (Algorithm 5) solves the variables in groups so that there are no two variables in the same group that have such a coupling between themselves that cause the solver to alternate between converge-divergence trends. So in any unique iteration it is guaranteed that no alternation will be caused, which means that the convergence probability will be higher.

**Summary**

We summarize and emphasize the weak spots of the proposed MIKP solver.

**No convergence** There might be two different reasons for this phenomena:

- There is no solution
- The kinematic equations are coupled. So, in each iteration one variable destabilizes the convergence of another variable.

In the first case, there is an inherent difficulty of the problem, while in the other case, there is a disadvantage of the method. So, in order to distinguish between the two cases and to solve the last one and stopping the trials of solving the first one, we will limit the number of iterations after few trials of converging in different portions of the solutions space.

**Non-optimal solution** There are different cases of convergence into an unfeasible or non-optimal solution:

- The process converges into an unfeasible solution (in terms of the joints values).
- The process converges into a solution that is "far" from the current posture

The first case is resolved by arbitrarily destabilizing the solution by changing the value of one or more joints and repeat the algorithm. There are other methods that increase the probability of achieving a stable feasible solution, but they will not be described in the scope of this paper.

The last case, in which the converged solution is "far" from the current posture, or not optimal in any other aspect (such as passing via a singular point, etc.), is not dealt in this research, since in most cases there are few solutions and our aim is to find one discrete solution.

However, despite these limitations, we found that the algorithms work well in practical (Chapter 8).

# Chapter 6

# Stability and Other Constraints

Practically, in humanoids, solving IKP is not enough. It has to be solved under some extra constraints, such as stability which is crucial in humanoid motion. In this chapter the multiple IKP algorithm is extended to support some additional constraints, rather than limit the solution to support only the kinematic constraints. It might be useful whenever the generated motion should be constrained, like the motion of biped robot, that its stability is frail. In this chapter we will develop constraints that can be assimilated in the constrained IKP problem. The focus of this chapter is on stability constraint.

First, we will extend the problem described in chapter 5 then the solver algorithm will be adjusted to the constrained IKP. Finally, the impact of these adjustments on the complexity, soundness and completeness are analyzed.

## 6.1 Formulation of Constrained Multiple IKP

In the extended problem, the IKP is solved under additional constraints. Dealing with humanoids, the most typical constraints concern the dynamics of the robot. For example, if the robot is in static equilibrium, each support point implies a constraint that the sum of all forces around it equals zero. So actually, two properties are satisfied for each support point. The first is geometric, and the second concerns the robot dynamics.

- The position of the support point does not change along the motion
- The infinitesimal sum of the different forces in these points is zero.

We first relate the additional non-kinematic conditions in general and then we will focus on specific constraints. Among these extra constraints might be geometric

boundary conditions such as self collisions' avoidance or physical constraint such as keeping the robot stable, or an optimization constraint such as minimizing the consumed energy, and others.

The challenge of solving a set of non-linear equations, as was defined in the multiple IKP (Chapter 4) is now more complicated. The set of the equations that have to be solved simultaneously is larger and there are more conditions that the solution has to obey them. The nature of the additional conditions is not known apriori as the kinematic constraints that are polynomials of transcendental functions. But, on the other hand, as the inverse kinematics problem is under-constrained, sometimes the additional constraints might limit the range for searching the solution. The major challenge with these additional constraints is that they may convoluted with the basic kinematic equations. Such convolution might cause the set of equations to be unsolvable.

Although the constrained multiple inverse kinematic problem is usually under-constrained, and has infinite number of solutions. In the extended problem we change the definition of $\bar{E}$, the set of equations to be linearized. Recall that the total number of kinematic equations are $3N_{sp} \times N_t$. Adding constraints that are implied on the kinematic set of equations will limit the number of solutions. e.g. the solutions' envelope is reduced and this leads to faster convergence.

There are three different types of additional constraints that we handle here:

- Range
- Physical or geometrical constraints
- Optimization

The first type actually limits the boundaries of the solution to be within a certain interval of the joint configuration space. These constraints always exist, because each joint has a physical range that is declared by the robot manufacturer. These constraints have the following form:

$$\begin{aligned} \theta_{i\ min} &\leq \theta_i \\ \theta_{i\ max} &\geq \theta_i \end{aligned} \tag{6.1}$$

These constraints will not be assimilated in the Jacobian matrix. Each generated solution will be validated to check whether it is within the appropriate interval. If it is not, another solution will be generated. This can be done by either initialize the process with new values of $\theta_i's$ or by using Algorithm 10 to find another zone

on the solutions' envelope. Usually, these constraints reduce the number of possible solutions, but the problem is yet under constrained.

The second type of constraints should be a function of $\Theta$, the joints values. An example of such a constraint is the requirement that the posture achieved by the solution will be stable. This type of constraints might be a geometric or a physical condition. The later, in its general form, might be a differential equation (in case it deals with the dynamics of the robot). A differential of order $\ell$ of such a constraint will be denoted as $C^{\ell}$ and it will be formed as follows:

$$\bar{C^{\ell}}(\Theta) = 0 \tag{6.2}$$

$\bar{C^{\ell}}(\Theta)$ is a set of constraints and the number of such constraints is $|\bar{C^{\ell}}(\Theta)|$. These constraints will be assimilated in the Jacobian. The details of the assimilation will be described later on.

The third type of constraints is on optimization functions. Whenever this type is used, there is usually only one constraint of this type. This limits the solution to be unique, or at least unique in a certain proximity of joints values (i.e. local minima or maxima). If there are more than one optimization function, there might be contradicting conditions. An example of such function is minimization of the rate change of joint angles. The raw form of the third type constraints is as follows:

$$\arg\min_{\Theta} F(\Theta)$$

Or,

$$\arg\max_{\Theta} F(\Theta)$$

This problem is solved by finding the zeros of the first derivative, i.e.:

$$F^{'}(\Theta) = 0 \tag{6.3}$$

The additional constraints are all functions of $\bar{\Theta}$, and can be partially derived in related to each variable $\theta_i$ in the Jacobian. Each additional constraint adds a row in the Jacobian. So, the Jacobian $J$, the partial derivative matrix of all equations in relative to each variable, has $3N_{sp} \times N_t + |\bar{C^{\ell}}(\Theta)|$ rows and its number of columns is $N$, which is the number of joints. The first type of constraints are not assimilated in the Jacobian since their validation can be easily checked due to their linearity in the joint space. The third type of constraints will be translated into the form of Eq. 6.2

(i.e. finding the zeros of the derivative of an optimization function).

Consider the kinematic constraints equations 4.3 and the additional constraints (Eq. 6.2) that will be partially derived in order to construct $J$.

$$\bar{E} = \begin{cases} \bar{T}_{support} T_{k-link_i}^{-1} A_{ki}^{-1} \dots A_{k0}^{-1} A_{jo} \dots A_{ji} T_{link_i} = \bar{T}_{target} \\ \bar{C}(\bar{\theta}) = 0 \end{cases} \tag{6.4}$$

From here on, $J$ is defined exactly as it was defined in the multiple IKP (Equation 4.5).

$$J = \frac{\partial \bar{E}(\Theta)}{\partial \bar{\Theta}} \tag{6.5}$$

If the kinematic equations and the constraints $\bar{E}$ were linear, then the solution of the kinematic equations under the constraints 6.2, were become to be:

$$\bar{\Theta} = \bar{\Theta_0} + \bar{V} \tag{6.6}$$

The equations of the constraints were not characterized, so it is unknown how close is the linearized function to the value achieved by the function itself. However, the kinematics equations are non linear so applying any method that is based on linearization requires iterative process to correct $\Theta$.

In this research we focus on constraints that are linear or polynomial or transcendental functions. These are the most typical constraints and they cover many common types. In these cases, they are of the same type as the kinematic functions and will have the same convergence nature, unless it is convoluted with the kinematics. We cannot treat this case a priori, but as part of the proposed algorithm, this is checked and treated on-line. (described later).

## 6.2 Constrained Multiple IKP Algorithm

The algorithms that were proposed in Chapter 6 to solve the multiple IKP do not require any change in order to solve the constrained problem. The only modification is that the Jacobian $J$ contains additional rows representing the extra constrains. The algorithms are based on minimization of the deviation of both the forward kinematic equations and the constraints under linearization. In order that the basic Algorithm 1 will be applied to the constrained IKP, the input should be as follows:

- $C$ - number of kinematic chains
- $N_c$ - number of joints in each joint
- Set of the robot D-H matrices $A_{c,j}$
- $R_0$ - the reference frame of the robot relative to the world
- $\bar{S}_p$ - the set of support points
- $\bar{T}_p$ - the set of target points
- $C^\ell$ - the set of additional constraints
- $\Theta_0$ - the current joint values
- MaxIterations - maximum number of iterations

There is no formal change in the algorithm itself, but notice that the calculation of $E$ (line 2) is now determined by Equation 6.4.

The second improvement is mostly important in the case of the constrained IKP, since it divides the set of equations $E$ into $M$ subgroups, $E_1, \cdots E_m$, ordered by their significance from the highest to the lowest. This enables granting different significance to the different constraints. Usually the stability constraint is the most important and has to be satisfied accurately, while the kinematics constraints have to be achieved approximately and any optimization constraint is unnecessarily be achieved.

## 6.3 Analysis

The analysis of the computational complexity is similar to the analysis of the basis multiple IKP Algorithm 1. The complexity is affected by two parameters. The first is the increased total number of constraints (rows in the Jacobian matrix). The complexity is derived from the size of the Jacobian that now has a bigger size. The second impact stems from the order of convergence, which depends on the coupling of the variables and whether there are convolution constraints. The probability to encounter these two convergence preventing factors is higher whenever there are more constraints. But unless we know the exact nature of the additional constraints, the change in the complexity cannot be accurately determined.

**An iteration Complexity.** The analysis of one iteration complexity described in Section 5.6, is changed by the number of rows, $N_r$, which affects the size of the Jacobian ($N_r \times N$).

$N_r$ is the total number of rows which is the sum of the kinematic equations and the additional constraints:

$$N_r = 3 \times S_p \times T_p + C \tag{6.7}$$

So, the complexity of substituting the joints values $\bar{\Theta}$, (line 8 in algorithm 1) becomes to be:

$$\mathrm{O}\big((3 \times S_p \times T_p + C) \times N^4)\big) \tag{6.8}$$

In line 9 $L$ is calculated. This is a $N_r^t h$ length vector. The number of calculations of an item is actually the number of calculations in $J(\Theta_0)$. As mentioned above, this matrix has $N_r \times N$ entries. The maximal number of operations of calculating each entity is $N^2$. So, the complexity of line 9 becomes to be:

$$\mathrm{O}\big(N_r^2 \times N^3\big) \tag{6.9}$$

The analysis of line 10 complexity compounds of counting the number of operations in matrix inversion and matrix multiplications. As mentioned above, there are $N_r \times N$ entities in matrix $J$. So, the multiplication $J_T J$ requires $N^2 N_r$ operations. Inverting this multiplication is performed by $N^3$ steps. The multiplication of $J^T L$ requires $N_r N$ operations. The last multiplication of $(J^T J)^{-1}$ by $J^T L$ is preformed by $N^2$ operations. The overall complexity of one iteration is:

$$\mathrm{O}\big((3{\times}S_p{\times}T_p{+}C){\times}N^4){+}N_r^2{\times}N^3{+}(1{+}N)(N_rN{+}N^2){+}2N\big) = \mathrm{O}\big(N^2(N_rN^2{+}N{+}1)\big) \tag{6.10}$$

And the total complexity of the constrained multiple IKP is formally the same as the complexity of the unconstrained multiple IKP (Equation 5.9). The difference is in the size of $S_p$ which is bigger. Recall that the total complexity of the algorithm is:

$$\mathrm{O}\big(S_p \times T_p \times 13 \times 4^3\big) = \mathrm{O}\big(S_p \times T_p\big) \tag{6.11}$$

The bottom line is that the complexity of one iteration of the constrained multiple IKP algorithm is the product of the number of supporting points and the target points.

But, the actual computation is slow since it is multiplied by a very big constant of order of the fifth power of the number of joints.

**Soundness.** There is no change in the analysis of the soundness. The additional constraints that are embedded as extra rows in the Jacobian do not change the characteristic of the algorithm. Whenever the algorithm converges, it converges into a valid solution. The same proof that was applies to the multiple IKP solver (Theorem 5.6), is still valid here since the properties of the Jacobian were not changed and the rest of the algorithm is similar to the basic algorithm 1.

Notice that there are two types of equations: the kinematic conditions and the additional constraints (e.g. keeping stability). The set of constraints, $\bar{C}$ has two types: (i) those that have to be satisfied up to a certain tolerance (usually, those exert from optimization function) and (ii) those that have to be fully satisfied (usually, exerted from a conditional function).

The first type, in terms of tolerance, behaves similar to the kinematic equations, that have to be satisfied up to a certain threshold that can be calibrated in advanced. The only thing that might be different is that the constraint is not necessary in the Euclidean space. In that case, one can estimate the mapping between the configuration space to the world coordinate space.

The second type is more difficult, and has to be treated prior to the performance of the algorithm, by adjusting the constraint itself. This process will be described in detail, later on, as various specific constraints will be described.

**Completeness.** As mentioned in Chapter 4, the algorithm is not complete. Implying constraints on the solution does not change this statement. There might be some cases that the multiple IKP solver does not converge into a solution while the constrained multiple IKP version converges and vice versa. But it does not turn over the principal argument of incompleteness.

## 6.4 Whole Body Stability Constraints

In this section the stability constraint is developed. This constraint will be implied later on the solution of the IKP. The first Section (6.4.1), describes the problem notation. The second Section (6.4.2), defines the problem as existing of a certain point within the convex hull of the robot supported polygon. In Section 6.4.3 the optional points and their computation are described, while 6.4.4 presents the algorithm to

calculate the convex hull of the supported polygon. Finally, the overall calculation of the stability constraint is described in 6.4.5.

### 6.4.1 Extended Stability Criteria: Notation

A general static stability of any physical object demands that either point: Grounded Center of Mass (GCom), Zero Moment Point (ZMP) or Foot Rotation Indicator (FRI) will be within the support surface, in a case of one supporting plane [51]. Dealing with few supports that are on the same plane, these points have to be within the convex hull (CH) of the planar supported polygon (SP) of these supports [15]. In this section we will extend the stability condition to the general case that the supports are in different planes and we will show that the CH of their projected points on the surface perpendicular to the gravity axes is sufficient.

We use the following notation for the inputs of the problem:

- Let $\bar{\theta}$ be the joints state vector.
- Let $X_r$ be the robot reference position. It is $4 \times 4$ homogeneous coordinate matrix. Usually, this input relates to the pelvis, but it might be any other reference point.
- Let $\bar{S}_p$ be a set of 3D support points. Each point $S_p$ is defined as a point on the robot (relative to a reference point of the robot). Each support point has a contact with the ground or any other stable surface.

Later on in this chapter we will use these inputs to formulate a general stability constraint. The constraint will be formulated so that the stability holds for any robot motion and any robot structure. Note, that this is a static stability constraint, that considers only the gravity and friction forces whose sum should equal zero, (assuming no sliding forces exist and the ground is stable). Each of the following subsections formulates the constraint for a different criterion.

### 6.4.2 The Problem

There are two types of stability: static and dynamic. While dealing with humanoid, the static stability is useful when the robot is in slow motion. Dynamic stability is used for fast motions such as walking and running when the robot might falls if its motion will be interrupted.

Static stability is a general term in physics that is used in robotics motion. It is useful while dealing with stationary devices or slow motion robots who are moving in a quasi static mode. The criteria for static stability is that the robot grounded projection of the center of mass (GCoM) lies within the supported polygon of its supports [35].

Dynamic stability is used in fast motions, where in each infinitesimal instance of the motion the robot might be unstable, but overall along the motion it is stable. There is no general algorithm to solve the problem of dynamic stability for bipedal robots; often used approaches are based on the zero moment point (ZMP) or the Foot Rotation Indicator (FRI). The dynamic stability criteria is that either the ZMP or the FRI point are within the support polygon every instance of time.

Dealing with static stability, only the gravity force is considered. Assuming that the support is a contact point without any push or pull forces, then no other force affects the stability. Even the friction can be ignored in such contacts. Therefore, the only impact is of the gravity, which is determined as the ground projection of the CoM. But since we deal with general motions, where the robot does not necessarily stand on a planar surface, and its support are not necessarily its feet, then the gravity will be projected on a plane perpendicular to the gravity axes, and in the height of the lowest support point.

In dynamic stability, as was defined by Vukobratović et al. ( [88]), the ZMP which is the point where the sum of the moments are zero, should be within the supported polygon. This general definition holds for any motion, as was stated in [87]:

$$F_p + F_A = 0 \tag{6.12}$$

Where $F_p$ is the force acting on the support's edge and $F_A$ is the rotation force about that edge. The point that satisfies this condition is the ZMP, whatever the motion is. While dealing with walking, both the support and the rotation point are lying on the feet, which simplifies the computation of this point. Later on in this chapter, the ZMP point and the support will be determined for any motion in the 3D space. However, the stability problem criteria is still to find whether the ZMP lies within the supported polygon. However, the calculation of the ZMP is much complex as well as the supported polygon, which no more can be assumed that it is on the ground plane.

The other dynamic stability criterion is the location of the FRI point, as was described by Goswami in [23]. Note that this criterion is defined just to motions where the support points are either one or both of the feet. We will extend it to any

support of the humanoid. It is assumed that the external forces acting on the robot are the resultant ground reaction force acting at the CoP (Center of Pressure) and the gravity. This can be determined by using the following equation:

$$M + OP \times R + \sum OG_i \times m_i g = \sum \dot{H}_{G_i} + \sum OG_i \times m_i a_i \qquad (6.13)$$

Where,

- $M$ and $R$ are the torque and force acting at the CoP.
- $OP$ is the vector between the origin and the CoP point (denoted as P).
- $OG_i$ - is the vector between the origin and the CoM location.
- $m_i$ is the mass
- $\dot{H}_{G_i}$ is the derivative of the angular momentum around the CoM.
- $a_i$ is the linear acceleration of the CoM.

From Equation 6.13 the extended FRI point can determined. Then, the stability criteria is that this point resides within the convex hull of the supported polygon of the supports.

So actually, all the above three criteria require that a certain point (GCoM, ZMP or FRI) will be within a supported polygon. This holds for any body motion, even if the supports are not the legs. There is a difference in the way the points and the supported polygon are calculated. The details of these calculations are described in the following sections. The FRI criteria, which deals with dynamic stability is beyond the scope of this research.

### 6.4.3 Stability Criteria Points

#### GCoM.

GCoM stands for Ground projection of the Center of Mass. In [88] it is stated that if the GCoM is within the convex hull of the supported polygon of the robot supports, then the robot is statically stable. This stability criteria is valid for any posture, even those whose support is not within the area of the feet.

The CoM is determined by multiplication of the static and the kinematic parameters of the robot, represented by the D-H matrices [30]. The ground projection of the

CoM can be computed by changing the Z coordinate to be the height of the ground relative the robot's origin.

Define $A_{ki}$ to be the D-H matrix of $i_{th}$ joint relative the previous joint $(i-1)$ in the $k^{th}$ chain. Notice that $A_{ik}$ is a function of $\theta_{ik}$, the kinematic value of the $i^{th}$ joint in the $k_{th}$ chain.

Denote by $B_{ik}$ the homogeneous matrix representing the location of the $i^{th}$ link center of mass relative the $i^{th}$ joint in the $k^{th}$ chain. $m_{ik}$ is the mass of the $i^{th}$ link in the $k^{th}$ chain.

Assume that the robot has $K$ kinematic chains, each of them has $n_k$ joints. Then, the center of mass of each chain relative the robot reference point can be determined as follows:

$$CoM_k = \frac{\sum_{i=1}^{n_k} A_{1k} \cdots A_{ik} B_{ik} m_{ik}}{\sum_{i=1}^{n_k} m_{ik}} \tag{6.14}$$

The CoM of the robot, will be computed as a weighted average of the CoM of all its kinematic chains:

$$CoM_{robot} = \frac{\sum_{k=1}^{K} \sum_{i=1}^{n_k} A_{1k} \cdots A_{ik} B_{ik} m_{ik}}{\sum_{k=1}^{K} \sum_{i=1}^{n_k} m_{ik}} \tag{6.15}$$

This CoM position is relative to the robot's origin. $X_r$ will be the transformation matrix of the robot's origin in relative to the world. To transform the robot's CoM also in relative to the world coordinate system, the following multiplication is performed:

$$CoM_{rw} = X_r CoM_{robot} \tag{6.16}$$

The CoM will be projected on the world XY plane, which is defined as the plane perpendicular to the gravity axis. If the robot reference point is relative the world coordinate system, then the projected Z coordinate will get the Z-value of the ground.

**The Zero Moments, Torques and Forces Constraint.** The Zero Moment Point, defined by Miomir Vukobratović in 1968 [87], is a criterion for stability of a walking biped robot. It specifies a point in the contact of the foot with the ground where the sum of the vertical forces (gravity and inertia) is zero.

Here, the definition is extended so that the ZMP is a point or a set of points

whose moment equals zero, and are located on the support surfaces. Since we limit our research only to gravity and friction forces (without considering push / pull or any other forces), then the stability constraint is that the ground projected ZMP should be within the ground projected supported polygon.

In the extended definition, the Zero Moment Point can be on any part of the body, not necessary on the foot, unless it is on a supporting surface. There might be more than one such a point.

The original assumptions that the contact area is planar and that the friction is high so that there are no sliding motions, will be valid in the extended model.

The original definition of the ZMP point as is defined in [87]:

$$\vec{OP} \times R + \vec{OG} \times m_s g + M_A + M_Z + \left(\vec{OA} \times F_A\right)^H = 0 \tag{6.17}$$

$\vec{OP}$ is the vector from the origin of the coordinate system $O_{xyz}$ to the support point. Notice that the link of the robot that is supported is given.

$R$ is the reaction of the support surface on the contact point.

$\vec{OG}$ is the vector from the origin of the coordinate system $O_{xyz}$ mass center of the body parts that are above the support point, i.e., those parts that their $Z$ coordinate is higher than the support point.

$m_s$ represents the mass of the upper body, i.e. all parts of the body that their $Z$ value is bigger than the support point's $Z$ coordinate.

$M_A$ is the moment of the joint adjacent to the support point. The details of its computation will be described later.

$F_A$ is the force induced in the joint adjacent to the support point.

$M_Z$ represents the moment of friction reaction forces that balances the vertical component of the moment $M_A$ and the moment induced by the force $F_A$ . The resultant forces of the inertia and gravity acting on a robot are:

$$F^{gi} = mg - ma_G \tag{6.18}$$

Where $m$ is the mass, $g$ is the gravity acceleration and $a_G$ is the linear acceleration of the Center of Mass.

The moment in any point $X$ is defined to be:

$$M_x^{gi} = \vec{XG} \times mg - \vec{XG} \times ma_G - \dot{H}_G \tag{6.19}$$

Where $\dot{H}_G$ is the rate of angular momentum of the center of mass.

Define $F^c$ and $M_x^c$ to be the resultant of the contact forces and the moment at $x$, respectively. Then, each contact point $x$, satisfies the following conditions:

$$\begin{aligned} F^c + mg &= ma_G \\ M_X^c + \vec{XG} \times mg &= \dot{H}_G + \vec{XG} \times ma_G \end{aligned} \tag{6.20}$$

Or, put another way

$$\begin{aligned} F^c + m(g - ma_G) &= 0 \\ M_X^c + (\vec{XG} \times mg - \dot{H}_G - \vec{XG} \times ma_G) &= 0 \end{aligned} \tag{6.21}$$

These equations show that the robot is dynamically balanced if the contact and inertia forces are strictly opposite to the gravity forces.

Define an axis $\Delta^{gi}$, where the moment is parallel to the normal vector from the surface about every point of the axis, then the Zero Moment Point (ZMP) necessarily belongs to this axis. The ZMP will then be the intersection between the axis $\Delta^{gi}$ and the ground surface such that:

$$M_Z MP^{gi} = \overrightarrow{P_{ZMP}G} \times mg - \overrightarrow{P_{ZMP}G} \times ma_G - \dot{H}_G \tag{6.22}$$

Where $M_Z MP^{gi}$ is parallel to the normal $(n)$ to the contact surface.

Because of the opposition between the gravity and inertia forces and the contact forces mentioned before, the Z point (ZMP) can be defined by:

$$\overrightarrow{P_{CoM}P_{ZMP}} = \frac{n \times M_P^{gi}}{F^{gi} \cdot n} \tag{6.23}$$

where $P_{CoM}$ is a point on the contact plane, e.g. the normal projection of the center of mass.

While extending it to a whole body motion, the contact point is not necessary in the foot and the support polygon does not necessary identical to the foot. In general, the support point (or set of points) is onto a given link which is in contact with a supporting surface. Assume that the point is supported by a horizontal plane which is parallel to the ground.

In [70], the dynamic stability model is extended to include kinematics constraints, as well as the contact forces of the interaction with the environment. The robot is represented as free floating articulated system pulled to support surfaces by gravity. The contacts can be between any part of the robot and a surface with any size or orientation.

The constraints of the contacts are that the velocities and accelerations at these points are zero. The robot dynamics is affected by its geometry, the gravity forces, the actuating torques and the contact forces. The dynamic constraints are developed under the premise that the contact bodies lay flat against the support surface.

Let $\vartheta_{6|S_p|}, \dot{\vartheta}_{6|S_p|}$ be the vectors of the linear and angular velocities of each support point. Then the following conditions hold:

$$\vartheta_{S_p} = \dot{\vartheta_{S_p}} = [0, 0, ..., 0]_{6|S_p|} \tag{6.24}$$

Consider the following definitions:

- Let $F_{ref}$ be the reference frame of the robot in relative to the world.
- Let $\bar{\theta}$ be the vector of the joints pose.
- Let $I$ be the Inertia matrix, which its size is $(|\bar{\theta}| + 6) \times (|\bar{\theta}| + 6)$.
- Let $\vartheta, \dot{\vartheta}$ be the end effector velocity and acceleration respectively.
- Let $b$ and $g$ be the Coriolis and gravity forces respectively. The size of each of these vectors is $(|\bar{\theta}| + 6)$.
- Let $\dot{q}$, $\ddot{q}$ be the velocity and acceleration of the joints, respectively
- Let $J_s$ be the matrix of the cumulative Jacobian of all supporting bodies in contact with the ground. $J_s = \Sigma_{i=1}^{|S_p|} J_{s_i}$
- Let $F_r$ be the vector of reaction forces and moments.
- Let $\Gamma$ be the vector of the actuation torques.

Then, the dynamic of the system including the actuating and contact forces is as follows:

$$I \begin{pmatrix} \dot{\vartheta} \\ \ddot{q} \end{pmatrix} + b + g + J_s^T F_r = \Gamma \tag{6.25}$$

As described in [69], we solve Eq. 6.25 under the condition represented in Eq. 6.24, which yields:

$$F_r = \bar{J_s^T}(\Gamma - (b + g)) + \Lambda_s \dot{J}_s \begin{pmatrix} \vartheta \\ \dot{q} \end{pmatrix} \tag{6.26}$$

Where $\Lambda_s$ is the apparent inertia matrix projected in the space defined by the support bodies, $J_s$ is the associated Jacobian matrix, and $\bar{J}_s$ is its dynamically consistent generalized inverse [38].

Plugging the above equation into 6.25, a dynamic model with contact constraints is obtained:

$$I \begin{pmatrix} \dot{\vartheta} \\ \ddot{q} \end{pmatrix} + N_s(b+g) + J_s^T \Lambda_s \dot{J}_s \begin{pmatrix} \vartheta \\ \dot{q} \end{pmatrix} = N_s^T \Gamma \tag{6.27}$$

Where $N_s$ is the dynamically consistent null space of $J_s$.

The FRI stability criterion has an inherent characteristic that it is a point on the ground (i.e. XY plane), and it is assumed to reflect a rotation point on the foot. While dealing with whole body motions, the rotation point might be on any part of the body, so will rename it to BRI (Body Rotation Indicator).

The most general model of the moments at the origin are formulated as follows:

$$M + OPXR + \sum OG_i \times m_i g = \sum \dot{H_{Gi}} + \sum OG_i \times m_i a_i \tag{6.28}$$

Where $m_i$ is the mass, $G_i$ is the CoM location, $a_i$ is the CoM linear acceleration and $H_{Gi}$ is the angular momentum of the $i^{th}$ link about the CoM.

In order to expand this formula to any body part, we should solve it for each support point in 3D without assuming that the right hand side equals zero, which leads to the following result:

$$M + OPXR + \sum OG_i \times m_i g - \sum \tau_i - \sum \dot{H_{Gi}} + \sum OG_i \times m_i a_i = 0 \tag{6.29}$$

This set of equations is computationally complex, and an iterative numerical method like Newton-Raphson ( [3]), should be applied to solve it.

### 6.4.4 The Convex Hull of the Supported Polygon

The supported polygon is the polygon generated by linking the projected support points of the robot. Assume the robot has $|\bar{S_p}|$ support points. Each support surface is represented by a point or a set of points. Algorithm to define these representing points is beyond the scope of this paper.

The points $\bar{S_p}$ are relative to the robot origin represented by transformation matrix $M_{robot}$. In order to project them, each point should be multiplied by $M_{robot}^{-1}$. Afterwards, they will be projected on the base plane, which is the one perpendicular

to the gravity axis, in height Z=0.

Denote the set of projected points as $\bar{S}_{proj}$. In order to construct the supported polygon, the points should be ordered by their azimuth relative to an arbitrary point. This order defines the polygon. Now, the convex hull of this polygon has to be constructed. The convex hull of the supported polygon is the circumscribing polygon of all projected supported points, that has no angle that is more than 180 degrees. Both the polygon and the convex hull are on the XY plane. Let define the azimuth between two points is the direction from the first point to the second point in the interval $[-\pi, +\pi]$.

The algorithm to construct the polygon is described in Alg. 11.

**Algorithm 11** Algorithm to construct the supported polygon

**Require:**

Set of points $\bar{S}_p$

The robot's origin matrix $M_{robot}$

1: **for** $\forall p \in \bar{S}_p$ **do** determine $\bar{S}_{p_{new}} = \bar{M_{robot}}^{-1} \bar{P}$
2: Set the $Z$ coordinate to 0
3: Choose an arbitrary point in the set $\bar{S}_{pnew}$ and mark it as $P_0$.
4: Calculate the azimuth of each of the other points relative to $P_0$
5: Reorder $\bar{S}_{pnew}$ in corresponding to their ascending azimuth value.

In line 1 the supported points are transformed from world coordinates to the robot coordinates. Line 2 is equivalent to projecting the points on the $XY$ plane. In lines 3–5 the supported points are ordered counterclockwise as a preparation process before constructing the supported polygon.

The output of Algorithm 0 is the set of points construct the supported polygon in counterclockwise order, where the first point was selected arbitrarily. This set of points is the input of Algorithm 12 which constructs the convex hull of the polygon.

The output of this algorithm is the set of points $\bar{S}_{pnew}$ which is the ordered points constructing the convex hull.

It should be mentioned, that the ordering of the points clockwise or counterclockwise was chosen arbitrary. By setting the points in a clockwise order $sgn$ will be positive and setting it counterclockwise will cause $sgn$ to be negative. Then, step 1 can be avoided.

Notice that our goal is to express the stability criteria as a constraint equation of the for $C(\bar{\theta}) = 0$. The CH construction is performed by an algorithmic process and its output is used as part of the constraint equation.

**Algorithm 12** Convex Hull Computation

**Require:** Set of points $\bar{S_{pnew}}$

1: Choose an arbitrary point located inside the polygon and check whether it is to the left or right side of the first line in the polygon (+/-). Mark the sign as $sgn$.
2: **repeat**
3: **for** each odd point in the set $\bar{S}_{pnew}$ **do**
4: define the line between the previous odd point and the current point.
5: if the previous even point has the same sign as $sgn$ then remove this point from $\bar{S}_{pnew}$.
6: **until** no point is removed anymore

Dealing with a walking robot (on a horizontal ground, perpendicular to the gravity axis), the supported polygon is a bounded area on the XY plane. Usually, it is the area bounded by the stance feet. This simple case is due to the fact that the stability is derived from 2 forces: the gravity which works on the XY plane and the ground reaction. Extending the discussion to any motion, this approach has to be extended to include other forces such as friction and torques (e.g. pull / push). Therefore, we should distinguish between stability (static equilibrium) and balance (dynamic equilibrium).

In this research we will limit ourselves to static equilibrium. Three conditions for static equilibrium:

$$\begin{aligned} \textstyle\sum F_v = 0 \quad & \textbf{sum of vertical forces} \\ \textstyle\sum F_h = 0 \quad & \textbf{sum of horizontal forces} \\ \textstyle\sum T = 0 \quad & \textbf{sum of torques} \end{aligned} \tag{6.30}$$

Since it is known that all points are on the same plane, then we can use Kirkpatrick-Seidel algorithm [16]. This algorithm computes the convex hull of n points on $O(nh)$ time complexity, where $h$ is the number of points define the convex hull.

Suppose, we have to calculate whether a point X resides within the convex hull, then we have to project this point to the plane, and examine the side of the point in relative to each edge of the convex hull. If all sides are the same, then the point is within the convex hull, otherwise - it is out of the border. [4]

The time complexity of testing each point whether it is inside or outside the convex hull, is $\mathrm{O}(h)$. This process has to be applied to each of the three planes. If the point resides within all three convex hulls, then the posture is stable. So, the total time complexity of determine the convex hull and calculate whether the GCoM

point resides within it is $O(nh)$.

Notice that there are some special cases. The convex hull is a good estimation for the supported polygon, but they are not identical. There might be some rare situations where the point is within the convex hull it is not in the boundaries of the supported polygon. There are some specific situations where the point resides on the boundaries of the supported polygon and due to inaccuracies the robot will not be stable.

Define $A_i$ to be the D-H matrix of joint $i$ in relative to the $i-1$ joint. Notice that $A_i$ is a function of $\theta_i$, the kinematic value of the $i^{th}$ joint. Denote by $B_i$ the homogeneous matrix of the location of the center of mass of the $i^{th}$ link in relative to the $i^{th}$ joint. Denote by $m_i$ the mass of the $i^{th}$ link.

Assume that the robot has K kinematic chains, each of them has $n_k$ joints. Then, the center of mass of each chain in relative to the reference point of the robot can be determined as follows:

$$CoM_k = \frac{A_{1k} \cdots A_{ik} B_{ik} m_{ik}}{n_k} \tag{6.31}$$

The CoM of the robot, will be computed as a weighted average of the CoM of the different kinematic chains. The CoM will be projected to the XY plane. But the supported polygon will be determined according to the scheme described in Section 6.4.3.

### 6.4.5 The Stability Constraint

A static stability exists whenever the GCoM of an object is within the CH (Convex Hull) of the SP (Supported Polygon). Given the robot state $\bar{J}$, the GCoM can be determined from Equation 6.15. The CH is determined by using the algorithm described in the previous section.

Assume the CH points are ordered counterclockwise, then if the GCoM is within the CH, the equation will have the same sign for every consequent points $s_i \in \bar{S}_{pnew}$.

$$(x_{i+1} - x_i)(y_{CoM} - y_i) - (y_{i+1} - y_i)(x_{CoM} - x_i) \tag{6.32}$$

Where, $x_{CoM}, y_{CoM}$ are the calculated coordinates of the robot's center of mass, projected on the ground. Actually, this equation represents two constraints, for $x_{CoM}$ and $y_{CoM}$. The $z_{CoM}$ is projected on the ground plane so there is no additional constraint.

If the convex hull is composed of $n_s = |\bar{S}_{pnew}|$ points, then this condition can be formulated into the following equation:

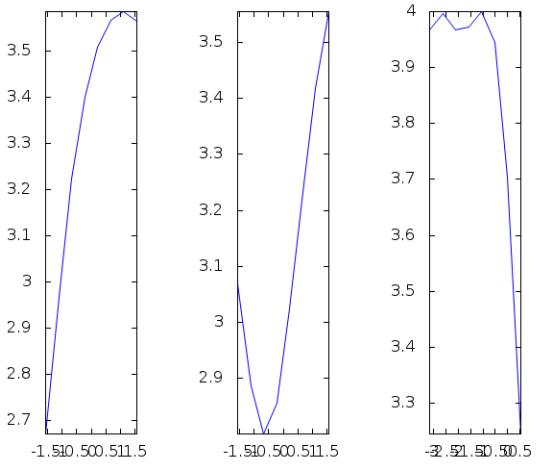
$$n - \left| \sum_{i=1}^{n} \frac{(x_{i+1} - x_i)(y_{CoM} - y_i) - (y_{i+1} - y_i)(x_{CoM} - x_i)}{|(x_{i+1} - x_i)(y_{CoM} - y_i) - (y_{i+1} - y_i)(x_{CoM} - x_i)|} \right| = 0 \tag{6.33}$$

Notice, that if the denominator equals zero, then the CoM is on the edge of the convex hull. Since this is an unstable posture, this condition will be determined prior to the calculation of Eq. 6.33. Also, the first and the last points of the convex hull are identical.

This constraint can be modified to express a different stability criteria, such as that the FRI/ZMP points are within the CH. In contradiction to kinematic conditions that can be satisfied up to a given accuracy $\epsilon$, this constraint should be strictly satisfied.

This constraint is actually a function of $\Theta$. It is interesting to see its nature before assimilating it in the constrained IKP. Figures 6-1 through 6-8 demonstrate the stability criterion as a function of each variable $\theta_1$ - $\theta_{26}$, correspondingly.

Figure 6-1: The stability constraint as a function of lar, lap and lkp




## 6.5 Other Constraints

The constrained MIKP can support many types of constraints. Similar to the stability, there are other constraints that have to be fully satisfied. On the other hand, there are optimization constraints that have to be fulfilled as much as possible, like the fastest solution, which can be translated into the posture that satisfies the kinematic

Figure 6-2: The stability constraint as a function of rar, rap and rkp

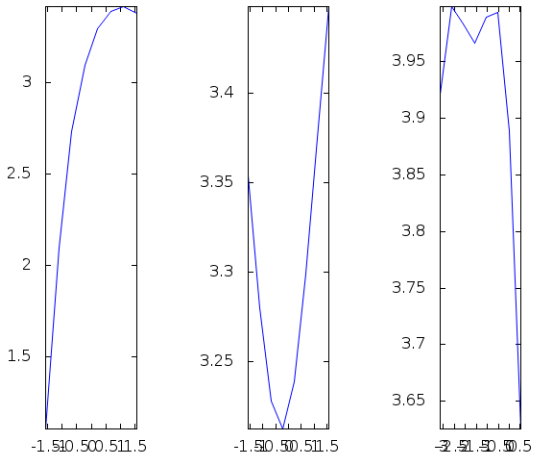
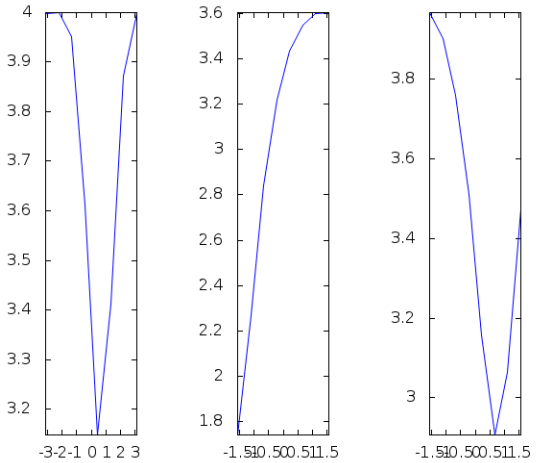



Figure 6-3: The stability constraint as a function of lhp, lhr and lhy




constraints and changes the joints at least. In this section the minimum jerk constraint will be briefly described.

This is a common trajectory generation model in robotics [86]. This model requires that the third derivative of the position, called jerk, will be minimal. It can constrain either the Cartesian coordinates or the joint configuration space. Obviously, each will result a different solution. In order to demonstrate how such a constraint should be formulated in order to assimilate it in the IKP solver, the minimum jerk in the Cartesian coordinate system will be developed.

The jerk of a coordinate $x$ is defined to be

$$\dddot{x}(t) = \frac{d^3x(t)}{dt^3} \tag{6.34}$$

Figure 6-4: The stability constraint as a function of rhp, rhr and rhy

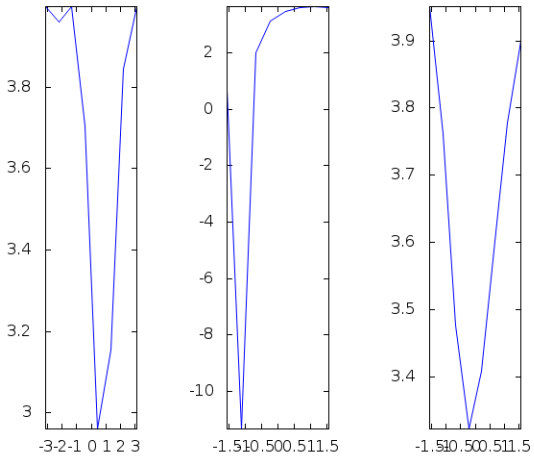
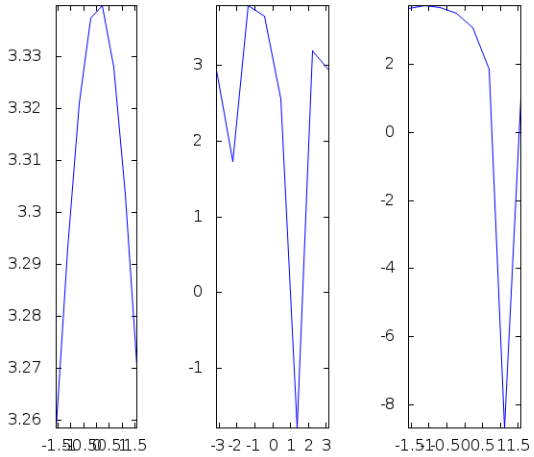



Figure 6-5: The stability constraint as a function of lsp, lsr and ler




In our case, $x$ is a function of $\Theta$ and usually $\Theta$ is a function of $t$. So, Equation 6.34 becomes

$$\dddot{x}(t) = \frac{d^3x(\Theta)}{d\Theta^3}(\frac{d\Theta}{dt})^3 + 3\frac{d^2x(\Theta)}{d\Theta^2}\frac{d\Theta}{dt}\frac{d^2\Theta}{dt^2} + \frac{dx(\Theta)d^3\Theta}{d\Theta dt^3} \tag{6.35}$$

The minimization function is along the whole trajectory, so it is defined as:

$$\min \int_0^t \dddot{x}^2(t) \tag{6.36}$$

As the trajectory is 3 dimensional, the function should minimize the sum of the squared jerk along its trajectory:

Figure 6-6: The stability constraint as a function of rsp, rsr and rer

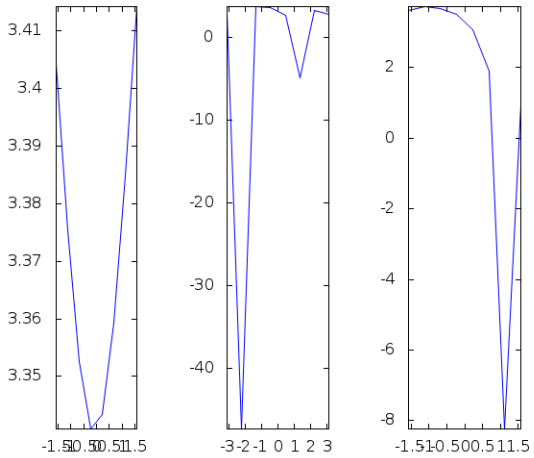
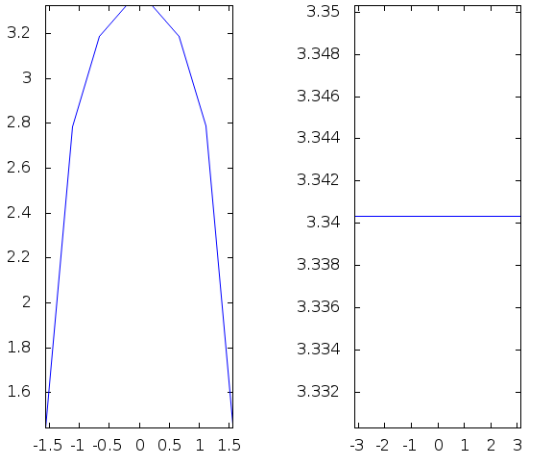



Figure 6-7: The stability constraint as a function of ley and lwy

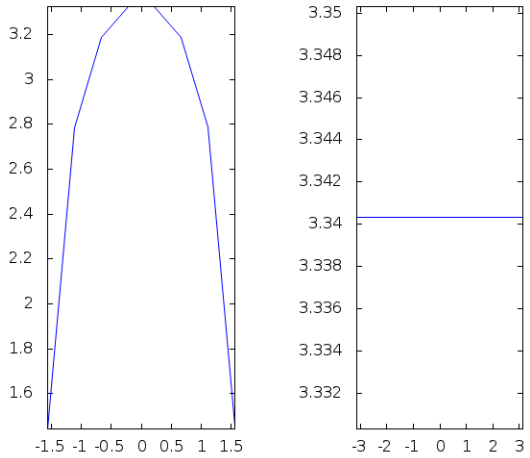



$$\begin{aligned}
&\min \int_0^t (\dddot{x}^2(t) + \dddot{y}^2(t) + \dddot{z}^2(t))dt = \\
&\min \int_0^t ((\frac{d^3x(\Theta)}{d\Theta^3}(\frac{d\Theta}{dt})^3 + 3\frac{d^2x(\Theta)}{d\Theta^2}\frac{d\Theta}{dt}\frac{d^2\Theta}{dt^2} + \frac{dx(\Theta)d^3\Theta}{d\Theta dt^3})^2 + (\frac{d^3y(\Theta)}{d\Theta^3}(\frac{d\Theta}{dt})^3 \\
&+3\frac{d^2y(\Theta)}{d\Theta^2}\frac{d\Theta}{dt}\frac{d^2\Theta}{dt^2} + \frac{dy(\Theta)d^3\Theta}{d\Theta dt^3})^2 + (\frac{d^3z(\Theta)}{d\Theta^3}(\frac{d\Theta}{dt})^3 \\
&+3\frac{d^2z(\Theta)}{d\Theta^2}\frac{d\Theta}{dt}\frac{d^2\Theta}{dt^2} + \frac{dz(\Theta)d^3\Theta}{d\Theta dt^3})^2)dt
\end{aligned} \tag{6.37}$$

$\Theta$ is a multi variable vector, so the derivatives in the above target function are partial derivatives. In the general case there are additional terms of the partial derivatives of these variables (e.g. $f(\frac{\partial\theta_i}{\partial\theta_j})$). But in our case all these terms equal zero, since none of these variables is a function of another one. [1] So, Equation 6.35

[1] In humans and some humanoids, this kind of relations between joints exist, but these are beyond the scope of this research.

Figure 6-8: The stability constraint as a function of rey and rwy




becomes

$$\begin{aligned}&\min \int_0^t (\sum_i (\frac{\partial^3 x(\theta_i)}{\partial \theta_i^3}(\frac{d\theta_i}{dt})^3 + 3\frac{\partial^2 x(\theta_i)}{\partial \theta_i^2}\frac{d\theta_i}{dt}\frac{d^2\theta_i}{dt^2} + \frac{\partial x(\Theta)\partial^3\theta_i}{\partial\theta_i dt^3})^2 \\ &+ \sum_i (\frac{\partial^3 y(\Theta)}{\partial \theta_i^3}(\frac{d\theta_i}{dt})^3 + 3\frac{\partial^2 y(\Theta)}{\partial \Theta^2}\frac{d\theta_i}{dt}\frac{d^2\theta_i}{dt^2} + \frac{\partial y(\Theta)\partial^3\theta_i}{d\theta_i dt^3})^2 \\ &+ \sum_i (\frac{\partial^3 z(\Theta)}{\partial \theta_i^3}(\frac{d\theta_i}{dt})^3 + 3\frac{\partial^2 z(\Theta)}{\partial \theta_i^2}\frac{d\theta_i}{dt}\frac{d^2\theta_i}{dt^2} + \frac{\partial z(\Theta)\partial^3\theta_i}{d\theta_i dt^3})^2) dt\end{aligned} \tag{6.38}$$

The minimization is achieved by finding the zeros of the derivative of the function, which is

$$\begin{aligned}&\sum_i (\frac{\partial^3 x(\theta_i)}{\partial \theta_i^3}(\frac{d\theta_i}{dt})^3 + 3\frac{\partial^2 x(\theta_i)}{\partial \theta_i^2}\frac{d\theta_i}{dt}\frac{d^2\theta_i}{dt^2} + \frac{\partial x(\Theta)\partial^3\theta_i}{\partial\theta_i dt^3})^2 \\ &+ \sum_i (\frac{\partial^3 y(\Theta)}{\partial \theta_i^3}(\frac{d\theta_i}{dt})^3 + 3\frac{\partial^2 y(\Theta)}{\partial \Theta^2}\frac{d\theta_i}{dt}\frac{d^2\theta_i}{dt^2} + \frac{\partial y(\Theta)\partial^3\theta_i}{d\theta_i dt^3})^2 \\ &+ \sum_i (\frac{\partial^3 z(\Theta)}{\partial \theta_i^3}(\frac{d\theta_i}{dt})^3 + 3\frac{\partial^2 z(\Theta)}{\partial \theta_i^2}\frac{d\theta_i}{dt}\frac{d^2\theta_i}{dt^2} + \frac{\partial z(\Theta)\partial^3\theta_i}{d\theta_i dt^3})^2 = 0\end{aligned} \tag{6.39}$$

This is the minimum jerk constraint to be assimilated in the IKP solver. It can be solved numerically by Newton-Raphson, using the Jacobian matrix, similar to the other constraints (e.g. the kinematics conditions).

# Chapter 7

# Motion During Interpolation

Motion can be planned discretely, along a set of via points. The transition between two via points is performed by the robot controller using an interpolation. The common approach is to use a combination of interpolation with feedback control [49]. In any case that the interpolation method is known, the full path can be predicted. However, the robot reaction to the feedback control might change this prediction.

Although there is a demand that robot motion will be as smooth as possible, motion planning is actually discrete. There are several reasons for the discretization of low level motion:

- There is a limitation for the planning duration. So actually tasks are planned piecewise.
- Robot motion planning should account for some dynamics, such as obstacles, unpredictable forces, uneven or slippery ground, etc.
- The robot might reach unstable postures
- There are some unreachable postures
- Most controllers work in the joints' configuration space, while most tasks are defined in the Cartesian space. The translation between the space is not straightforward.

In piecewise motion planning the starting point and the end point of the segment are well defined in both the joint configuration space and the Cartesian space. The motion in between the segment edges are interpolated. The points defining the segments start and end points are called "knots" or via-points.

The quality of piecewise planning is affected by the density of the via points. From one hand, as these points are more dense, the trajectory is more controllable and the path is more predictable. As a consequence, a high accuracy can be achieved and there is a higher chance to guarantee stability.

On the other hand, high density of via points requires a lot of computation resources, which leads to a slow performance, useful just for off-line planning. This is not very useful for humanoids, as it requires frequent calculations of the IKP which is a complex problem, especially in humanoids that have few kinematic chains and compound of 20–30 joints.

In a case where the accuracy of the robot locomotion is required, the path should be planned in advanced, hence feedback control cannot be used. Therefore, in order to keep the robot stable, high density of the via-points is required. Determining a stable robot posture at each such via point is a heavy computationally task.

Often, controllers use linear joints interpolation. The generated path is unpredictable, but if the via points are close enough, then the deviation from the linear path between the two via points is ignorable, and a good accuracy of following a certain path can be achieved.

There are few common interpolation methods that are used to keep the smoothness of the trajectory in the via points and between them. Among those methods are:

- Spline [27, 59, 97]
- Bezier [32, 48, 101]
- Linear interpolation [29]

The methods differ in the end-effector path that is generated.

In this chapter the effect of the knots' density and the interpolation method on the total distance, and the accuracy of the path are analyzed and demonstrated. In addition, we propose an algorithm to find the next via point towards the target point, that guarantees keeping stability.

Using the algorithm proposed in Chapter 4 allows frequent calculations of the IKP. In this section we will analyze the effect of the via points density on the path accuracy, the stability sensitivity and the performance. Finally, we will describe a method to estimate the maximum distance between two via points.

## 7.1 Properties of an Interpolated Path

Different interpolation methods and different via-points density results a different path. In robotics, the controller, in its low level, works in the joints space, so linear joint interpolation is usually used. The density of the via points is determined by the user, who defined them prior to the controller execution. This section describes some measurements to compare between the generated paths.

The characteristics of a spatial trajectory that we would like to know prior to executing a task are:

- What is the spatial length of the trajectory?
- Does the robot lose its stability along the trajectory?
- Are there any singular points of the robot along the trajectory?

In order to accomplish the analysis of a trajectory, six functions of the resultant interpolated trajectory are defined. The functions are:

- The Cartesian length
- The length of the end effector path
- The aggregated length of the joints motion
- The path feasibility
- The robot stability along the path
- The existence of singular points along the path.

The first three functions are measuring different distances. The first is the spatial distance between the current end effector location and the target location (which is not affected by the interpolation method of the via points density, it is used as a datum). The second measures the total distance of the end effector while moving along the transition between the two states. In contrast to the first two measures that relate the Cartesian space, the third one is a measure in the joint space. It is the aggregated distance that each link (relates to its previous joint) is passing along the overall transition.

Both the current position and the target position can be determined by multiplication of the relevant D-H matrices to achieve the forward kinematics of the robot. In this chapter we will develop the general formulas of the different distances, and

later on they will be solved numerically. In order to give the reader the feeling of how do they look like, their actual form relating the Nao robot, with the left foot as a support and the right palm as a target are presented in Appendix C. This kinematic chain was chosen since it is the longest in typical motions. This analysis ignores the orientation of the end effector, so only three equations will be satisfied (X, Y and Z).

The equations are useful for any humanoid with 26 DoF. The lengths of the links is kept as parameter according to table 7.1.

Table 7.1: The abbreviations of the links lengths

| | |
|---|---|
| $l_f$ | length of the foot |
| $l_s$ | length of the shin |
| $l_t$ | length of the thigh |
| $l_p$ | length of the pelvis |
| $l_{sp}$ | length of the spinal |
| $l_{sh}$ | length between shoulders |
| $l_{ua}$ | length of the upper arm |
| $l_{la}$ | length of the lower arm |
| $l_{pa}$ | length of the palm |

**Distance.** Denote $\bar{\Theta}_0, \bar{\Theta}_n$ as the vectors of the joints values in the initial and final poses. The Cartesian distance between the current end effector and the target location is not affected by either of the interpolation method and the via points density, and will be a general measure. $D$ will represent this distance and like a datum, all other measures will be compared to it.

$$D = \sqrt{\left((X(\bar{\Theta}_n) - X(\bar{\Theta}_0))^2 + (Y(\bar{\Theta}_n) - Y(\bar{\Theta}_0))^2 + (Z(\bar{\Theta}_n) - Z(\bar{\Theta}_n))^2\right)} \tag{7.1}$$

**End-effector Path Length.** This measure sums the total length of the end effector while transiting from posture $\Theta_0$ to $\Theta_n$. It is represented in its discrete fashion as follows:

$$L = \sum_{\bar{\theta}} \sqrt{\left(\left(X(\theta_{t_{i+1}}) - X(\theta_{t_i})\right)^2 + \left(Y(\theta_{t_{i+1}}) - Y(\theta_{t_i})\right)^2 + \left(Z(\theta_{t_{i+1}}) - Z(\theta_{t_i})\right)^2\right)} \tag{7.2}$$

In linear joint interpolation, each $\theta_j$ changes according to the following function:

$$\theta_j(t) = a_j t + \theta_j(0) \tag{7.3}$$

Where $a_j$ is the rate of angular change, i.e. the angular velocity which can be predetermined by the user or the controller. Then, the continuous function of the path length becomes to be:

$$L = \int^{t} \sqrt{(dx^2 + dy^2 + dz^2)}dt \tag{7.4}$$

Where,

$$\begin{array}{l} dx = \frac{d^2x}{d\bar{\Theta}^2}\left(\frac{d\Theta}{dt}\right)^2 + \frac{dxd^2\bar{\theta}}{d\Theta dt^2} \\ dy = \frac{d^2y}{d\bar{\Theta}^2}\left(\frac{d\Theta}{dt}\right)^2 + \frac{dyd^2\bar{\theta}}{d\Theta dt^2} \\ dz = \frac{d^2z}{d\bar{\Theta}^2}\left(\frac{d\Theta}{dt}\right)^2 + \frac{dzd^2\bar{\theta}}{d\Theta dt^2} \end{array} \tag{7.5}$$

Recall that the $\Theta$ is a vector of the joints variables, therefore Equation 7.5 becomes,

$$\begin{array}{l} dx = \sum_i \left(\frac{\partial^2 x}{\partial\theta_i^2}\left(\frac{d\theta_i}{dt}\right)^2 + \frac{\partial x d^2\theta_i}{\partial\theta_i dt^2}\right) \\ dy = \sum_i \left(\frac{\partial^2 y}{\partial\theta_i^2}\left(\frac{d\theta_i}{dt}\right)^2 + \frac{\partial y \partial^2\theta_i}{d\theta_i dt^2}\right) \\ dz = \sum_i \left(\frac{\partial^2 z}{\partial\theta_i^2}\left(\frac{d\theta_i}{dt}\right)^2 + \frac{\partial z \partial^2\theta_i}{d\theta_i dt^2}\right) \end{array} \tag{7.6}$$

Notice that $\frac{d\theta_i}{st}$ is the angular velocity which is often constant. The partial derivatives of the coordinates $X$, $Y$ and $Z$ are determined in advanced symbolically. The integration of the function is performed numerically, by using Quadrature method. Applying Eq. 7.6 to the Nao example is described in Appendix C.

**Aggregated Joints Movement** This criterion is analogous to an energy measurement. It aggregates the movement of each joint multiplied by the length of its next link.

$$E = \sum_j \int \ell_{j+1} d\theta_j \tag{7.7}$$

Where $j$ is the joints' number and $\ell_{j+1}$ is the length of the link following the actual joint. This summation is calculated numerically, as well.

**Feasibility** While dealing with joints value interpolation, if both values of $\theta_j^0$ and $\theta_j^t$ are feasible the postures along the transition will be also feasible, unless there are

self-collisions or passing through singular points.

To avoid self-collisions, a set of geometric equations has to be determined. This calculation has high-order polynomial complexity. Moreover, this keeps the motion planning in the field of discrete problems, which the interpolation aims to change to be piecewise continuous. However, in order to make sure that the transition is feasible this calculation has to be preformed frequently. On the other hand, these collisions can be represented as a continuous geometric condition in the solutions' envelope. The sensitive domains on the envelope will be predicted.

In a case that the solution is closed to the sensitive domain, then high density of via points is required and it demands heavy calculations. In that case, the motion planning should be off-line.

Avoiding singular points is much simpler. It can be solved by changing the motion direction, but this solution might generate non-optimal transition path that will not necessary obey the condition of minimal spatial distance.

In general, joint linear interpolation is resistant to unfeasible postures, since the feasibility of the required poses of each joint can be determined in advanced. If both initial and target poses are feasible, there should be a set of feasible poses that is achieved by interpolation of each joint. In any case that an interpolated pose generates a self-colliding posture, it can be solved by implying delays on some joints motions. This solution which is off-line implementable is not straightforward and is not in the scope of this research.

Notice that in every via point the IKP should be solved. Since there might be few feasible solutions, we have to choose the one that is most optimal. Optima can be measured in terms of minimal weighted angular changes as is defined in Eq. 7.7. Selecting the optimal solution among all feasible solutions is beyond the scope of the IKP solver proposed in Alg. 1. This means that the larger the number of via points the higher the probability of non-optimal path. But this is just because our algorithm does not choose the most optimal solution. We leave selection of the optimal solution to future work.

**Stability.** The robot might run out of stability during the interpolation between two stable poses. The sensitivity of the pose along the interpolation to the robot stability is equivalent to the sensitivity of the GCoM position along the interpolation in relative to the convex hull of supported polygon of the robot.

Define $Com_{ij}$ to be the transformation matrix of the center of mass of the $j^{th}$ joint in the $i_{th}$ kinematic chain of the robot, in relative to its origin. Then, consider the

GCoM calculation as a sequence of the D-H matrix multiplication. Then, $CoM_{ij}$, the transformation matrix represented the location and size of the center of mass of the rigid body of the $i^{th}$ kinematic chain in relative to the origin of the robot is:

$$CoM_{ij} = A_0 A_{i1} \dots A_{ij} Com_{ij} \tag{7.8}$$

Then, after projecting the GoM on the support surface, the robot GCoM is:

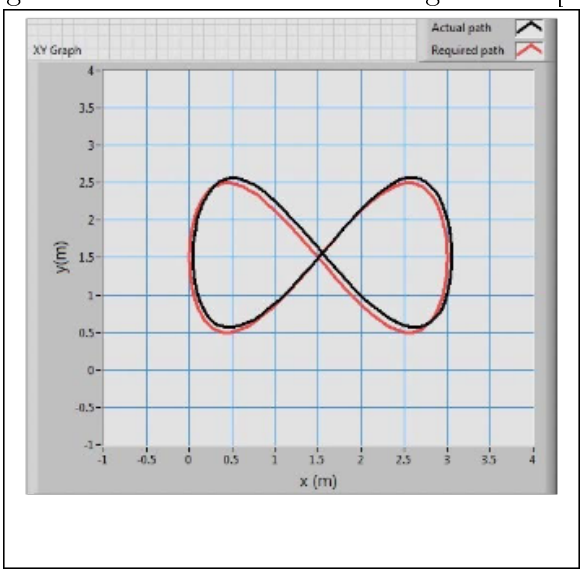
$$\begin{aligned} X_{com} &= \frac{\sum_i \sum_j CoM_{ij}[1,4] \cdot m_{ij}}{\sum \sum m_{ij}} \\ Y_{com} &= \frac{\sum_i \sum_j CoM_{ij}[2,4] \cdot m_{ij}}{\sum \sum m_{ij}} \end{aligned} \tag{7.9}$$

Notice that the $z$ value of the projected point equals to the height of the support surface. The set of points constructing the convex hull of the supported polygon are determined by the algorithm described in chapter 6.

The condition that the GCoM is within the convex hull was defined in Eq. 6.33. Let us denote this inequality $C(\theta)$.

**Accuracy.** The different interpolations methods and via points density affect the accuracy of the path. This means that the actual path is not precisely followed the planned path of the end-effector (See Figure 7-1). Different interpolation will generate deviations. It is straightforward that increased density of the via-points increase the accuracy of the path. In the next section we will analyze the effect of the via points density.

Figure 7-1: Path deviation during motion [98]



The deviation of the actual path is considered to be "white noise", i.e. random deviation from a uniform distribution. Its source is in the deviations of the joints values. These deviations happen due to calibration errors, finite precision of measuring the mechanics, and sensors limitations. Each engine is provided with producer declaration about its maximal allowed error. This error is usually percentage of the overall motion or an absolute angular value. These errors are aggregated during the motion. Let's assume that each joint has another angular deviation, that is $\pm\delta_{\theta_i}$.

According to the error propagation theory, the overall error estimation of each step in the $X$, $Y$ and $Z$ coordinates, will be denoted as $m_x$, $m_y$ and $m_z$, correspondingly and will be calculated as follows:

$$\begin{aligned} m_x &= \sqrt{\sum_i \left(\frac{\partial X(\theta_i)}{\partial \theta_i}\right)^2 \delta_{\theta_i}^2} \\ m_y &= \sqrt{\sum_i \left(\frac{\partial Y(\theta_i)}{\partial \theta_i}\right)^2 \delta_{\theta_i}^2} \\ m_z &= \sqrt{\sum_i \left(\frac{\partial Z(\theta_i)}{\partial \theta_i}\right)^2 \delta_{\theta_i}^2} \end{aligned} \tag{7.10}$$

The overall deviation, $m$ is

$$m = \sqrt{m_x^2 + m_y^2 + m_z^2} \tag{7.11}$$

The calculation of $X$, $Y$ and $Z$ and their partial derivatives is done as part of the symbolic forward kinematic calculation and the Jacobian matrix $J$, as detailed in Chapter 4.

Equation 7.11 is a function of $\delta_{\theta_i}$, which is the step size of the interpolation. The next section is a study of the different interpolation methods, which their deviation is compared in the Section 7.3.

## 7.2 The Interpolation Method

Interpolation, in general, is used to smooth the motion in terms path position, velocity and acceleration. Usually it is required to have a linear interpolation, which means to generate a motion on the linear line between the current position and the target position. Hence, there is a gap between this requirement, which constraints the Cartesian space and the actual motion generation which is implemented in the robot joint configuration space. In order to perform linear interpolation, there should be defined a mapping between these two spaces.

Using the notation of [30], assume that a linear interpolation between the starting point $A$ and the target point $B$ is required. Define the points,

$$\bar{X}_A = \begin{pmatrix} p_{x,A} \\ p_{y,A} \\ p_{z,A} \\ \alpha_A \\ \beta_A \\ \gamma_A \end{pmatrix} = \begin{pmatrix} \bar{P}_A \\ \bar{\Phi}_A \end{pmatrix} \qquad \bar{X}_B = \begin{pmatrix} p_{x,B} \\ p_{y,B} \\ p_{z,B} \\ \alpha_B \\ \beta_B \\ \gamma_B \end{pmatrix} = \begin{pmatrix} \bar{P}_B \\ \bar{\Phi}_B \end{pmatrix} \tag{7.12}$$

In linear interpolation the end effector should move along a straight line.

An intermediate position and orientation can be calculated as follows:

$$p(t) = p_A + \frac{s_p(t)}{s_f}(p_B - p_A)$$

$$\Phi(t) = \Phi_A + \frac{s_\Phi(t)}{s_{\Phi f}}(\Phi_B - \Phi_A) \tag{7.13}$$

In each intermediate point, the inverse kinematics should be implemented and in-between two consequent intermediate points, the actual interpolation implemented by the low level controller command is a linear joints interpolations.

The cubic spline is an improved interpolation procedure that replaces the straight line connecting the path points with a third degree polynomial. It affects both the path and its derivatives, i.e. the velocity and acceleration profiles. The third degree polynomial is of the form:

$$x(t) = \alpha_3 t^3 + \alpha_2 t^2 + \alpha_1 t + \alpha_0$$

$$y(t) = \beta_3 t^3 + \beta_2 t^2 + \beta_1 t + \beta_0 \tag{7.14}$$

$$z(t) = \gamma_3 t^3 + \gamma_2 t^2 + \gamma_1 t + \gamma_0$$

As with linear interpolation a new set of coefficients must be used for each interval between the available data points. These coefficients are chosen to give a smooth transition between consequent points. The smooth behavior is accomplished by computing the polynomial coefficients for each interval using more than just the adjacent points.

The last interpolation uses Bezier splines, which are useful to design trajectories with certain roundness properties. Bezier splines are general family of splines of varied order. Bezier splines of the third order are the cubic splines. In Bezier splines, the interpolated points are defined by a series of control points with associated weighting factors. Those weighting factors attract the curve towards the control points. Bezier curves are based on Bernstein polynomials with normalized intervals, which can be derived from the following formula:

$$1 = ((1-u)+u)^n = \sum_{k=0}^{n} \binom{N}{k} (1-u)^(n-k) u^k \tag{7.15}$$

In this research we focus on Bez'ier splines of the second order. $N$ is the total number of via points within a planned motion segment.

In the next section some experimental results of how the different interpolation methods affect the same parameters of path length in two spaces and the accuracy are compared.

## 7.3 The Effect of the Via-Points Density

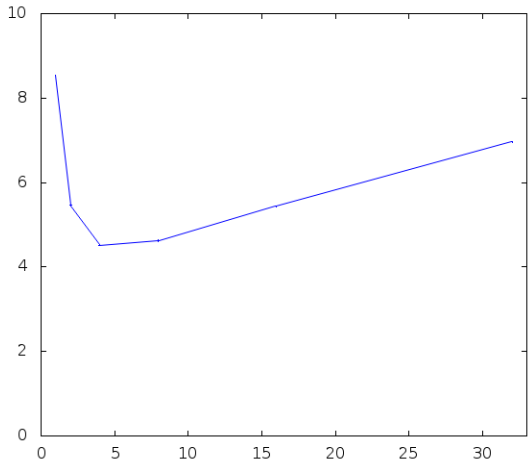
The measurements that were described above will be compared as a function of the via points density. We will focus on the three distance measurements and the accuracy. The stability will not be demonstrated since it its dependency on the specific posture is too tight, so there is no logical basis to perform a general analysis. Regarding the performance, it is increasing linearly as a function of the number of via points. As was presented in Chapter 4, the complexity of the IKP is polynomial, but relatively high-order.

In Figure 7-2, the effect of increasing $n$, number of via points on the end-effector total length (Eq. 7.6) is described. It is compared to the Cartesian distance between the initial and target points. It is clear that the total length is increasing, but its maximal length is bounded.

The length is calculated by Equation 7.6. This equation calculates the total length as the root of the quadratic equation of each coordinate ($x$, $y$ and $z$). The term of the deviation of the length of each coordinate is a summation of the deviations stem from the angular change of each joint. This deviation is defined for $x$, $y$ and $z$ as above (Eq. 7.6).

Dealing with linear joints interpolation, the second term of each deviation is zero, i.e.

Figure 7-2: The total path length of the end effector. The horizontal axis represents the number of via-points and the vertical axis represents the total path length




$$\frac{\partial X d^2\theta_i}{\partial \theta_i dt^2} = 0$$

$$\frac{\partial Y d^2\theta_i}{\partial \theta_i dt^2} = 0 \tag{7.16}$$

$$\frac{\partial Z d^2\theta_i}{\partial \theta_i dt^2} = 0$$

Dealing with cubic spline interpolation, this can be rewritten as:

$$\frac{\partial X d^2\theta_i}{\partial \theta_i dt^2} = \frac{\partial X}{\partial \theta_i}\left(\left(\frac{x_{i+1}+x_{i+2}-x_{i-1}-x_{i-2}}{6}t\right)dt\left(\frac{x_{i+1}-x_{i-1}}{2}\right)\right)$$

$$\frac{\partial Y d^2\theta_i}{\partial \theta_i dt^2} = \frac{\partial Y}{\partial \theta_i}\left(\left(\frac{y_{i+1}+y_{i+2}-y_{i-1}-y_{i-2}}{6}\right)dt\left(\frac{y_{i+1}-y_{i-1}}{2}\right)\right) \tag{7.17}$$

$$\frac{\partial Z d^2\theta_i}{\partial \theta_i dt^2} = \frac{\partial Z}{\partial \theta_i}\left(\left(\frac{z_{i+1}+z_{i+2}-z_{i-1}-z_{i-2}}{6}\right)dt\left(\frac{z_{i+1}-z_{i-1}}{2}\right)\right)$$

Dealing with Bezier Interpolation, this term is

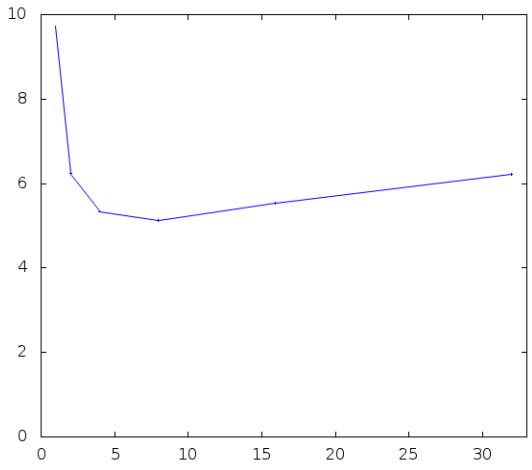
$$\frac{\partial X d^2\theta_i}{\partial \theta_i dt^2} = \frac{\partial X}{\partial \theta_i}\left(\frac{x_{i+1}-x_{x-1}}{2}\right)$$

$$\frac{\partial Y d^2\theta_i}{\partial \theta_i dt^2} = \frac{\partial Y}{\partial \theta_i}\left(\frac{y_{i+1}-y_{i-1}}{2}\right) \tag{7.18}$$

$$\frac{\partial Z d^2\theta_i}{\partial \theta_i dt^2} = \frac{\partial Z}{\partial \theta_i}\left(\frac{z_{i+1}-z_{x-1}}{2}\right)$$

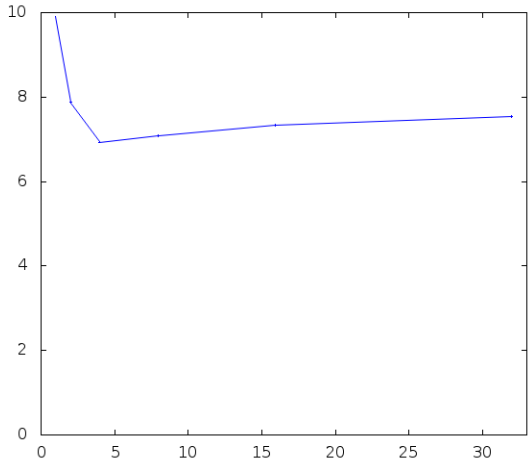
Therefore, the total length as a function of the density of the via points, using Spline interpolation is described in Figure 7-3. The horizontal-axis represents the number of via-points and the vertical-axis represents the total path length

Figure 7-3: The total path length of the end effector (Spline Interpolation)




And, the total distance that the end effector moves using Bezier interpolation is described in Figure 7-4.

Figure 7-4: The total path length of the end effector (Bezier Interpolation). The horizontal axis represents the number of via-points and the vertical axis represents the total path length




Now, the criteria of the sum of the total distance of each joint during the movement is compared as a function of the density of the via points. Figures 7-5 - 7-7 are demonstrating it in relating to linear joint, spline and Bezier interpolations, respectively.

Figure 7-5: The summation of the joints total distance (linear joints Interpolation). The horizontal axis represents the number of via-points and the vertical axis represents the total length of the joints

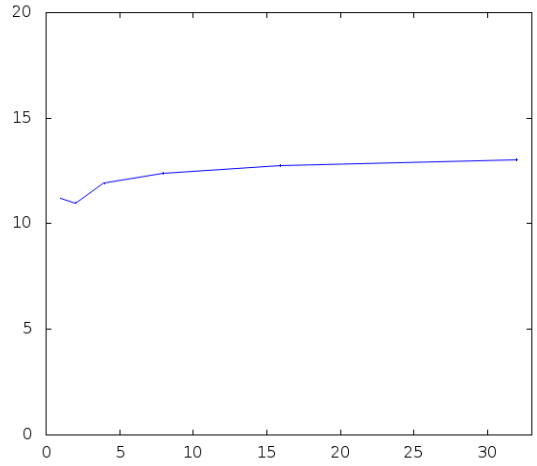
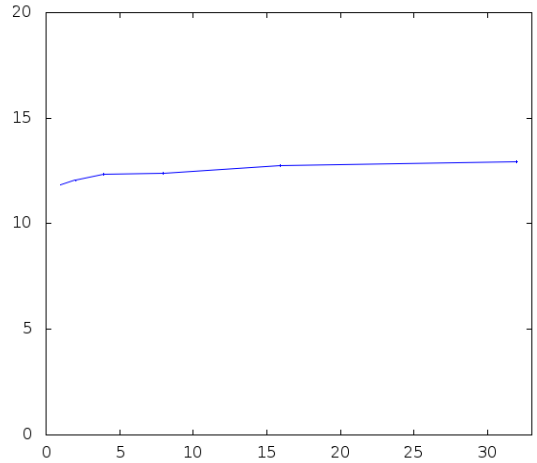



Figure 7-6: The summation of the joints total distance (Spline Interpolation). The vertical axis represents the aggregated path length of the joints




Finally, it is interesting to compare the accuracy derived from each type of interpolation, as a function of the via-points density. It is represented in Figures 7-8, 7-9 and 7-10, respectively.

The conclusion is that the both the interpolation function and both the number of via points affect the deviation of the actual path from the planned path. No matter what is the interpolation method, as much via points as we take the accuracy increases until a point where it decreases. This means that there is an optimal number of via points, in terms of accuracy.

Figure 7-7: The summation of the joints total distance (Bezier Interpolation). The horizontal axis represents the number of the interpolated via points.

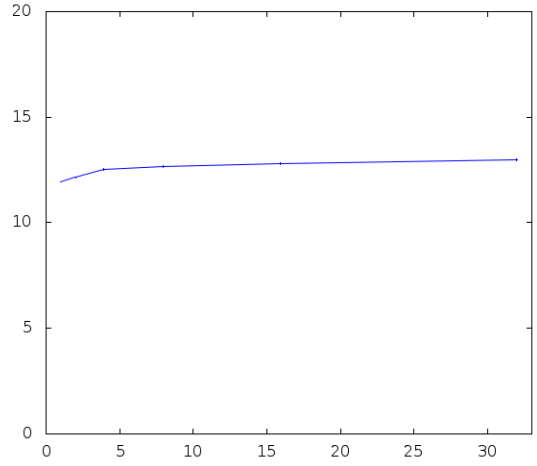
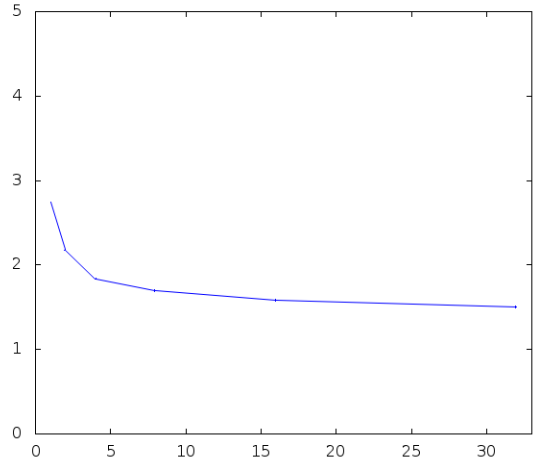



Figure 7-8: The accuracy of the path (linear joints Interpolation). The horizontal axis represents the number of the interpolated via points. The vertical axis represents the aggregated deviation of the actual path from the planned one.




## 7.4 An algorithm to find the next via-point

Assume that the robot structure is known, the inputs for motion planning are similar to those defined for the IKP problem in Chapter 4. Recall, that $\Theta$ is the vector of the current joints state. $R_0$ is the robot reference position. It is a $4 \times 4$ homogeneous coordinate matrix. Usually, this input relates the pelvis, but it might be any other reference point.

$S_p$ is the set of 3D support points. $S_p$ is defined as points on the robot (in relative to the zero point). Notice that a support point has a contact with the ground or any other surface that it can be laid on. A support surface will be presented by a single

Figure 7-9: The accuracy of the path (Spline Interpolation)

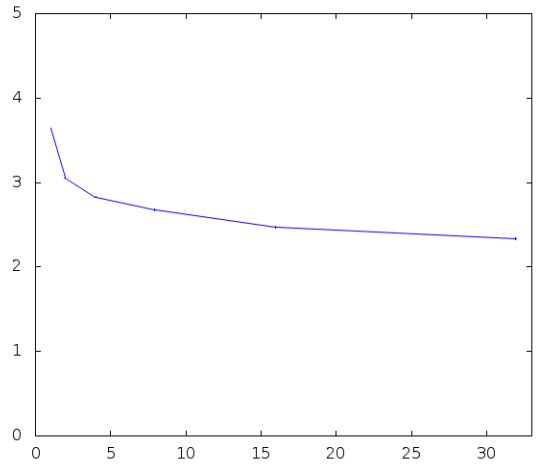
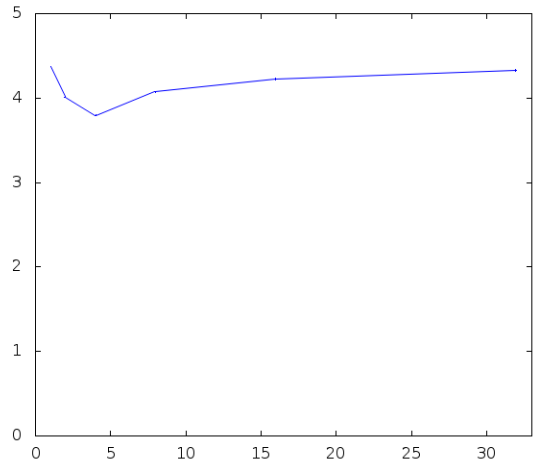



Figure 7-10: The accuracy of the path (Bezier Interpolation). The horizontal axis represents the number of the interpolated via points.




point or a set of points. $T_p$ is the set of 3D target points that the robot has to touch them but not lay on.

The new input is $D_{max}$ which is the maximal distance between the linear path and the actual path, allowed by the accuracy requirement.

The output of the problem is $X_v$, the Cartesian coordinates of the via points that lies on the direct line between the current and target position, and located in the maximal distance from the current position, but still satisfies both the stability condition and the maximal distance requirement. If $X_v$ equals $T_p$, then the transition to the target point is feasible.

Notice that in any case that $X_v \neq T_p$ then there is a feasible transition up to $X_v$, but there might be that along the next transition, between $X_v$ to the target point,

the robot will lose its stability or will not follow the linear path or will pass through a singular point.

In each via point the constrained IKP with the stability condition (from Chapter 6) is calculated and $\Theta_t$ is its result. This is the state vector of the joints in the target or via point position.

As was mentioned above, the transition between $\Theta_0$ and $\Theta_t$ is performed by the controller as linear interpolation of the joints values. This method arises the following problems:

- Will the robot keep its stability along the transition
- Will the robot follow the linear path (or any other path) in the Cartesian space
- Is the transition feasible, i.e. does the robot pass through singular points
- Is the transition optimal in terms of Cartesian distance and angular distance

The proposed algorithm finds the point on the linear path between the current position and the target position that is located in maximal distance from the current position and is reachable by linear joint interpolation to keep stability and follow the Cartesian line within the given threshold $D_{max}$. The soundness and completeness of the algorithm are discussed. Then, the complexity is analyzed.

For simplification of the model analysis, without loss of generality, we assumed that the robot has one support point and one target point. In the general case, each couple of these parameters contributes an additional condition to the set of equations.

We developed and analyzed the equations of the transition. Then, we present an algorithm that determines the density of the via points that guarantees stable motion. The equations and analysis will utilize the Nao structure (in terms of the number of joints and the lengths of the links). The algorithm can be applied to any robot structure with any DoF, and any number of kinematic chains. The kinematic structure of the Nao robot is described in Appendix A. Here, the implementation of the algorithm is described, for a specific robot configuration.

Assume, the origin frame is located in the robot's pelvis (0,0,0), the support is in the left foot and the target has to be reached by the right palm. Then, the right palm position can be represented as follows:

$$
\begin{aligned}
&X_{rpalm}(\bar{\theta}) = -0.009\sin(\theta_{rsr}) + 0.1925\sin(\theta_{rey})\cos(\theta_{rsr}) + 0.098 \\
&Y_{rpalm}(\bar{\theta}) = -0.1925[-\sin(\theta_{rey})\cos(\theta_{rsp})\sin(\theta_{rsr}) - cos(\theta_{rey})\sin(\theta_{rsp})] \\
&+0.009\cos(\theta_{rsp})\cos(\theta_{rsr}) + 0.09\sin(\theta_{rsp}) \\
&Z_{rpalm}(\bar{\theta}) = -0.1925[\cos(\theta_{rey})\cos(\theta_{rsp}) - \sin(\theta_{rey})\sin(\theta_{rsp})\sin(\theta_{rsr})] \\
&+0.009\sin(\theta_{rsp})\cos(\theta_{rsr}) - 0.09\cos(\theta_{rsp}) + 0.075
\end{aligned}
\tag{7.19}
$$

The forward kinematics of the foot in relative to the pelvis is based on a much complex chain. We use the following variables:

$$
\begin{aligned}
x_1 &= 0.5[cos(\theta_{lhy}) + 1] \\
x_2 &= 0.5[cos(\theta_{lhy}) - 1] \\
x_3 &= \tfrac{\sin\theta_{lhp}\sin\theta_{lhy}}{\sqrt{2}} - \cos\theta_{lhp}\sin\theta_{lhr}x_1 + \cos\theta_{lhp}\cos\theta_{lhr}x_2 \\
x_4 &= \tfrac{-\cos\theta_{lhp}\sin\theta_{lhy}}{\sqrt{2}} - \sin\theta_{lhp}\sin\theta_{lhr}x_1 + \sin\theta_{lhp}\cos\theta_{lhr}x_2 \\
x_5 &= -\tfrac{\cos\theta_{lhp}\sin\theta_{lhr}\sin\theta_{lhy}+\cos\theta_{lhp}\cos\theta_{lhr}\cos\theta_{lhy}}{\sqrt{2}} - \sin\theta_{lhp}\cos\theta_{lhy} \\
x_6 &= -\tfrac{\sin\theta_{lhp}\sin\theta_{lhy}}{\sqrt{2}}(\sin\theta_{lhr} + \cos\theta_{lhr}) + \cos\theta_{lhp}\cos\theta_{lhy} \\
x_7 &= -\tfrac{\sin\theta_{lhp}\sin\theta_{lhy}}{\sqrt{2}} + \cos\theta_{lhp}\cos\theta_{lhr}x_1 - \cos\theta_{lhp}\sin\theta_{lhr}x_2 \\
x_8 &= \tfrac{\cos\theta_{lhp}\sin\theta_{lhy}}{\sqrt{2}} + \sin\theta_{lhp}\cos\theta_{lhr}x_1 - \sin\theta_{lhp}\sin\theta_{lhr}x_2
\end{aligned}
\tag{7.20}
$$

Then the forward kinematics formulas become to be:

$$
\begin{aligned}
&X_{lfoot}(\bar{\theta}) = 0.1506\left\{\cos\theta_{lap}\left[x_3\sin\theta_{lkp} + x_4\cos\theta_{lkp}\right] + \sin\theta_{lap}\left[x_3\cos\theta_{lkp} - x_4\sin\theta_{lkp}\right]\right\} \\
&-0.1\left[x_3\cos\theta_{lkp} - x_4\sin\theta_{lkp}\right] - 0.12x_3 + 0.005x_4 - 0.05 \\
&Y_{lfoot}(\bar{\theta}) = 0.1506\left\{\sin\theta_{lap}\left[x_5\cos\theta_{lkp} - x_6\sin\theta_{lkp}\right] + \cos\theta_{lap}\left[x_5\sin\theta_{lkp} + x_6\cos\theta_{lkp}\right]\right\} \\
&-0.1\left(x_5\cos\theta_{lkp} - x_6\sin\theta_{lkp}\right) + 0.005x_6 - 0.12x_5 \\
&Z_{lfoot}(\bar{\theta}) = 0.1506\left\{\cos\theta_{lap}\left[x_7\sin\theta_{lkp} + x_8\cos\theta_{lkp}\right] + \sin\theta_{lap}\left[x_7\cos\theta_{lkp} - x_8\sin\theta_{lkp}\right]\right\} \\
&-0.1\left[x_7\cos\theta_{lkp} - x_8\sin\theta_{lkp}\right] - 0.12x_7 + 0.005x_8 - 0.115
\end{aligned}
\tag{7.21}
$$

In linear joints interpolation the current value of each joint is increased proportionally along the overall motion time, by dividing the difference between the target joint value and the current joint value into infinitesimal angular portions.

Applying linear joints interpolation between two consequent points might lead to the following problems:

- Reaching unstable positions

- Reaching locations that are highly deviated from the linear Cartesian path between the two points
- Generating long paths.

The sensitivity of this condition to the changes derived by the linear joints' interpolation are actually the sensitivity to changes in $\bar{\Theta}(t)$, and it can be determined by the partial derivative of condition 6.33 to the changes of $\Theta_j(t)'s$.

Therefore, if the intersection point of the linear forward and backward motion is within the convex hull, then the stability will be kept along the interpolation. Otherwise, the motion should be divided by via points into piecewise interpolated paths. This will be repeated iterative, according to algorithm 13. The following definitions are required by the algorithm.

$C(\Theta)$ is actually 2 equations (for the x and y coordinates). Define $\Lambda$ to be the partial derivative of $C(\Theta)$ in relative to each joint $\theta_i$, i.e. $\Lambda$ is a $2Xn$ matrix.

$$\Lambda = \frac{\partial C(\Theta)}{\partial \theta_i} \tag{7.22}$$

Then, the algorithm is described in Alg. 13. In lines 1–14 four different substitutions of joints values in the partial derivative matrix $\Lambda$ are performed. The substituted values are of the joints at the initial position, the joints at the final position (as calculated by the MIKP, Alg. 1) and the differences between those values from the initial to the final position and vice versa. Lines 17–21 are the major loop that determines how many via-points are required within the interval in order to guarantee that the robot will stay stable along the interpolation.

### 7.4.1 Soundness

In principle, Algorithm 13 does not guarantee soundness. The basics of the algorithm is applied linearization to the GCoM equations and to substitute the values of $\theta_i's$ at the endpoints to test whether the mid-point is also within the convex hull of the supported polygon.

However, since the GCoM equations are polynomials of transcendental functions, they are non-linear on one hand, but they are smooth on the other hand. The mid-point is estimated by 3 different models:

- extrapolation from the start point
- extrapolation from the end point

**Algorithm 13** Calculation the most far via-point along the path that guarantees keeping stability during interpolation

**Require:** $\Theta_0$ - the joints values at the current pose.
**Require:** $\Theta_t$ - the joints values at the target.
**Require:** $CH$ - the convex hull of the supported polygon
**Require:** $\ell$ - the spatial distance between the current and target positions

1: $\Lambda_0 \leftarrow \Lambda_t < -\Lambda$
2: **for** $j = 1$ until $j = n$ **do**
3: **for** $k = 1$ until $k = n$ **do**
4: **if** $k \neq j$ **then**
5: **for** $i = 1$ until $i = 2$ **do**
6: Calculate $\Lambda_0 i, j(\theta_{0_k})$
7: Calculate $\Lambda_t i, j(\theta_{t_k})$
8: **else**
9: **for** $i = 1$ until $i = 2$ **do**
10: Calculate $\Lambda_0 i, j(\frac{\theta_{t_k} - \theta_{0_k}}{2})$
11: Caculate $\Lambda_t i, j(-\frac{\theta_{t_k} - \theta_{0_k}}{2})$
12: $i \leftarrow i + 1$
13: $k \leftarrow k + 1$
14: $j \leftarrow j + 1$
15: $i = 1$
16: **repeat**
17: **if** $\{\Lambda_0[1, i], \Lambda_0[2, i]\} \notin CH$ **then**
18: $e_1 \leftarrow True$
19: **else**
20: $e_1 \leftarrow False$
21: **if** $\{\Lambda_t[1, i], \Lambda_t[2, i]\} \notin CH$ **then**
22: $e_2 \leftarrow True$
23: **else**
24: $e_2 \leftarrow False$
25: $D = \sqrt{(\Lambda_{ts}[1, 1] - \Lambda_{0s}[1, 1])^2 + (\Lambda_{ts}[2, 2] - \Lambda_{0s}[2, 2])^2}$
26: $i \leftarrow i + 1$
27: **until** $((i > n) \vee (e1) \vee (e2) \vee (D > \ell))$
28: **if** $(i > n)$ **then**
29: return(the target point as the next via point)
30: **else**
31: $\theta_t < -\frac{\theta_0 + \theta_t}{2}$
32: goto 1

- intersection point of the two derivatives of both end points.

If one of these points is out of the convex hull, then a via point is defined. This is a repetitive process, and at each iteration $\delta\theta_i$ is smaller, so that the extrapolation range is closer to the derivation points.

### 7.4.2 Completeness

The algorithm always returns a via point such that if the path passes through it, there is a high probability that the path is feasible and the robot will keep its stability. If such a motion exist while moving directly to the end point, then the returned via point is identical to the target point.

There might be that although a direct motion is feasible, the algorithm will return another via point. That means that on one hand, the performance will be worse, but on the other hand, the accuracy of following the linear path between the two point will be better. However, if there exists any feasible path between the two points, the algorithm will return such a path.

### 7.4.3 Complexity

In each iteration there are $O(n^3)$ steps, where $n$ is the number of joints. After each iteration that computes the partial derivative of the current and target GCoM locations, there are two calculations whether the estimated via point is within the convex hull of the supported polygon. This requires $O(S_p)$, which is the number of the points of the convex hull. Since at each iteration a smaller portion of $\delta\theta_i$ is taken, then the number of iterations is $\log\max(\delta\theta_i)$. So the overall complexity is bounded by:

$$O(\log\max(\delta\theta_i)n^3 S_p) \tag{7.23}$$

Therefore, applying this algorithm in real time is feasible as soon as the frequent of calculation is not very high.

# Chapter 8

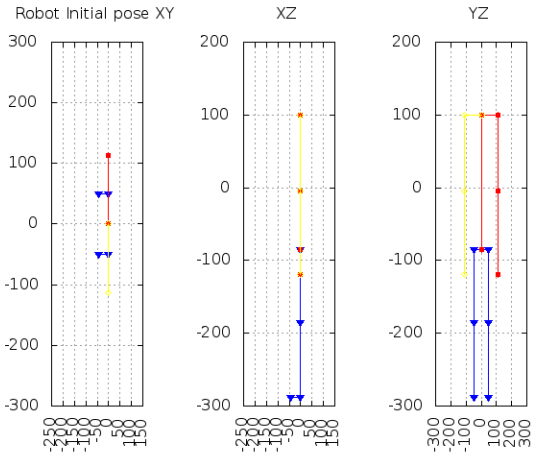
# Results

The results of embedding the static stability constraint (Eq. 6.4) in the constrained IKP solver are stable postures. In this section some simulations of the proposed method are discussed. The Nao robot model is simulated by using Maxima [66].

## 8.1 Raising hand

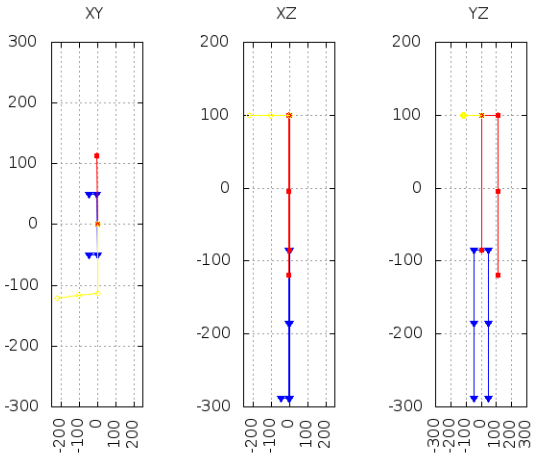
In the initial posture of this example the robot stands with both hands along its body (Fig. 8-1). In the target position the right hand has to be raised upwards. The transition between postures can be done by moving only one joint (rsp) in one chain (right arm). This transition does not affect the robot stability. Such movements are required in many grasping tasks.

Figure 8-1: The projection of the robot initial posture on three planes (XY, XZ and YZ)



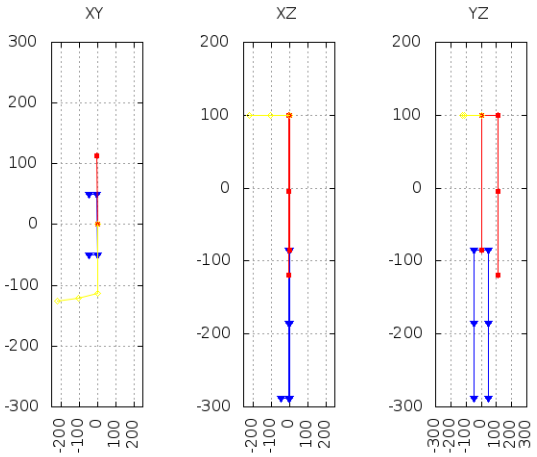
Giving a target position that the right arm is pointing forward, the resultant posture of the basic IKP Algorithm (without any constraint) [1] is presented in Figure 8-2

Figure 8-2: The IKP result for the raised arm posture




Applying the ICKP algorithm (2) with the stability condition to the same target position, the following result is obtained (Fig. 8-3):

Figure 8-3: The ICKP result for the raised arm posture




Applying the improved ICKP Algorithms 2–7 to the same target point of the raised arm, results the postures presented in Figures 8-4-4-1, correspondingly.

Concluding the results of Section 8.1, one can see that both the constrained and unconstrained IKP algorithms solve the problem. In this case the target position can be reached by changing only the arm joints. This means that in this case, the posture

Figure 8-4: Result of improved ICKP Algorithm 2 for the raised arm posture

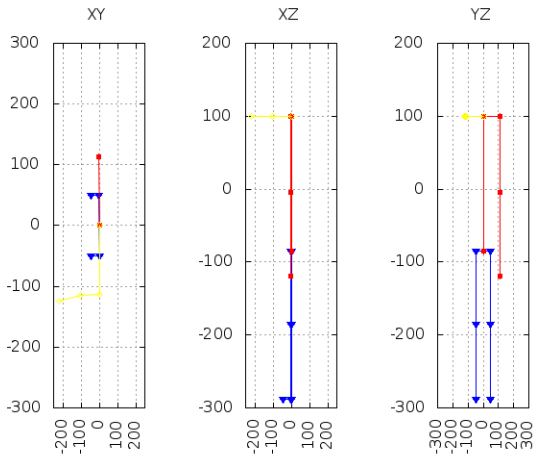
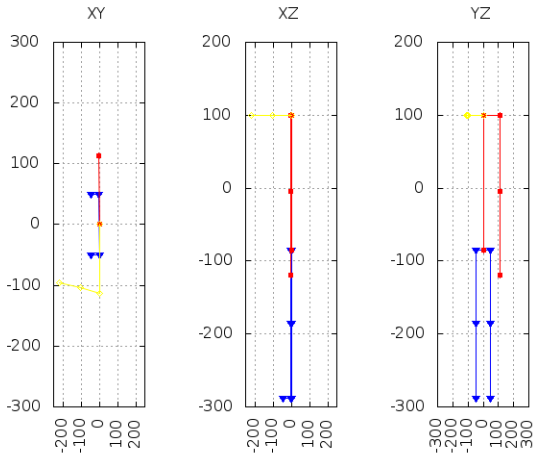



Figure 8-5: Result of improved ICKP Algorithm 3 for the raised arm posture




is not sensitive to losing the stability. The iterative process was terminated in all algorithms, after 25 iterations or as soon as the process converged.

The difference between the algorithms is in their rate of convergence, which is shown in Figures 8-8 through 8-13.

Figures 8-8 through 8-13 demonstrate that both the IKP and the ICKP with all its improvements converge to a solution. But they differ in the rate of convergence and in the accuracy of the solution.

Figure 8-14 presents the rate of change in the joints values along the iterative process in both the IKP Algorithm 1 and the ICKP Algorithm 2.

As can be learned from Figure 8-14 both the IKP and the ICKP converge into a valid solution in the case of raising an arm which involves just one kinematic chain movement. The rate of convergence is different. While the IKP converge faster, the

Figure 8-6: Result of improved ICKP Algorithm 5 for the raised arm posture

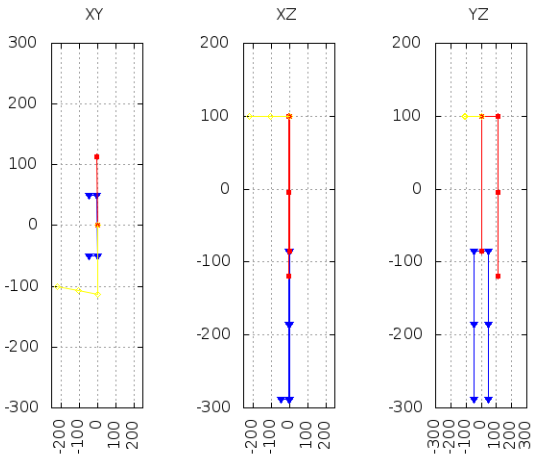



Figure 8-7: Result of improved ICKP Algorithm 7 for the raised arm posture

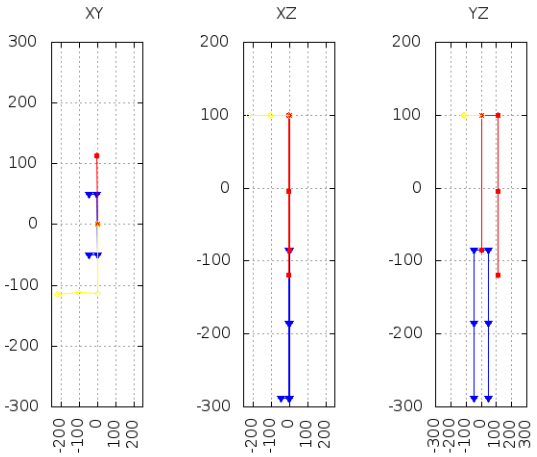



Figure 8-8: Convergence of applying the IKP Algorithm 1 to the raised right arm posture

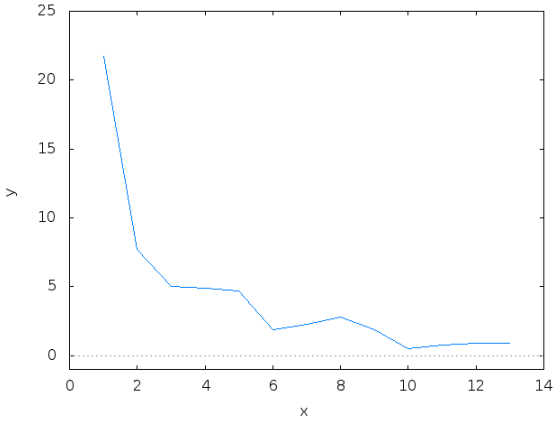


Figure 8-9: Convergence of applying the ICKP Algorithm 2 to the raised right arm posture

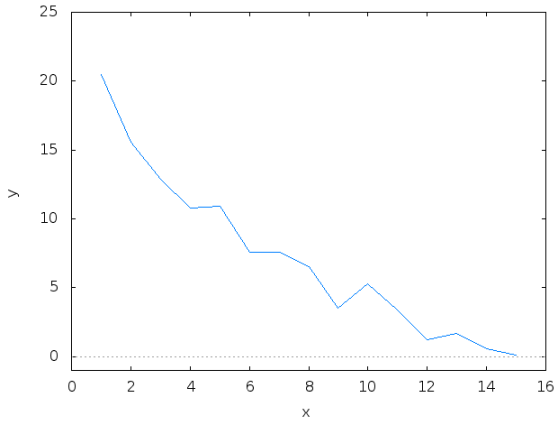



Figure 8-10: Convergence of applying the improved ICKP Algorithm 3 to the raised right arm posture

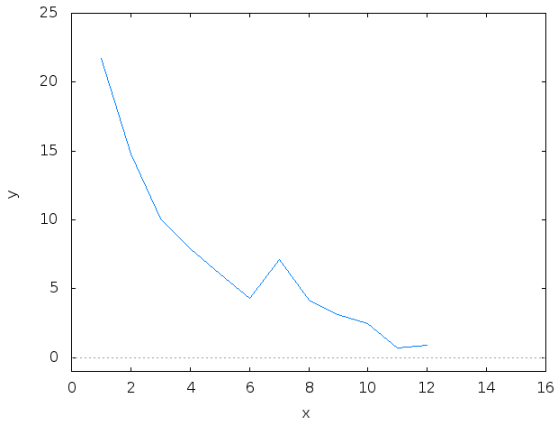



Figure 8-11: Convergence of applying the improved ICKP Algorithm 5 to the raised right arm posture

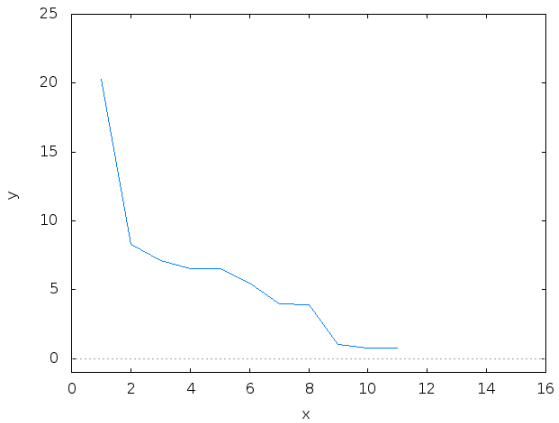


Figure 8-12: Convergence of applying the improved ICKP Algorithm 7 to the raised right arm posture

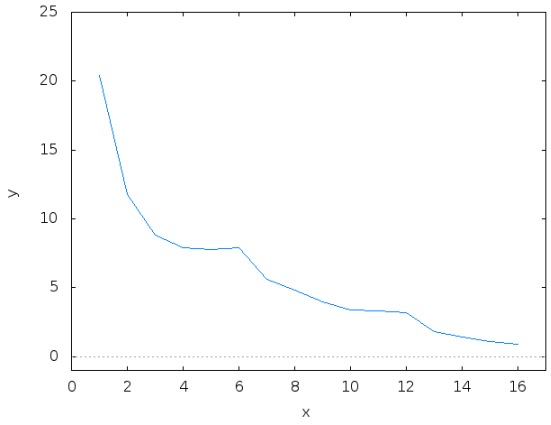
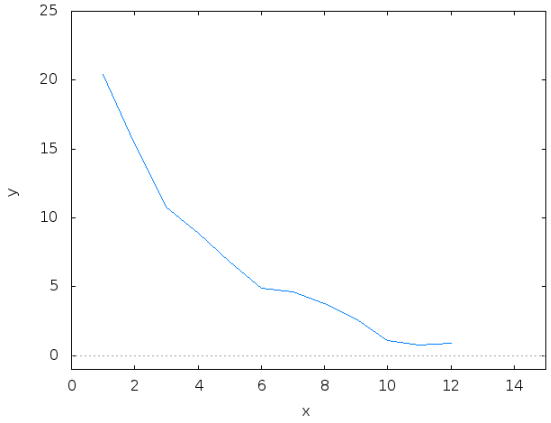



Figure 8-13: Convergence of applying the improved ICKP Algorithm 9 to the raised right arm posture




joints values oscillate more during the process.

## 8.2 Stand to Sit

In this transition the supports are the robot's foot. Originally, the robot stands stably. The constraints at the target position is that there is an additional support on the buttocks.

The robot initial posture is as defined when all joints equal zero and the robot stands. There are two supports (for the feet) and one target point, without loss of generality, the left hip.

The result of applying the IKP algorithm without any constraints is shown in Figure 8-16.

Figure 8-14: The joints' correction along the iterative process of the IKP Algorithm 1 (left) and the ICKP Algorithm 2 (right)

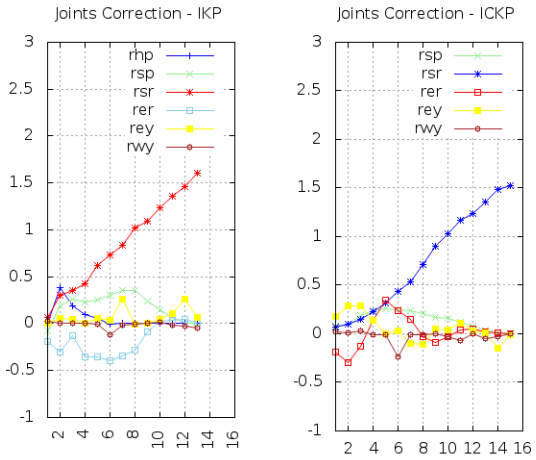
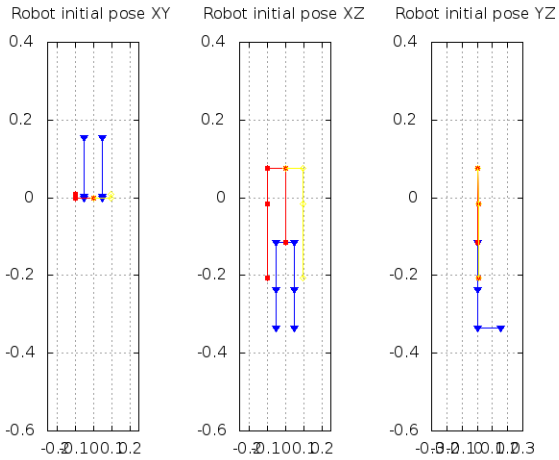



Figure 8-15: The robot initial posture (projected in three planes)




The aggregated error of each iteration is calculated as follows:

$$\theta_j = \theta_j + (\bar{J_j}^T(\bar{\theta})\bar{J}(\bar{\theta}))^{-1}\bar{J_j}^T(\bar{\theta})\bar{L} \tag{8.1}$$

The process converged after 8 iterations, as showed in Figure 8-17. The changes of the joints values along the iterations are shown in Figure 8-18. Values are in radians. The horizontal axis represents the iteration number and the vertical axis represents the aggregated change of the joints values in radians. (Aggregation of all the values that were changed, in all kinematic chains)

The changes of the joints values along the iterations are shown in Figure 8-18. Notice that joints which were not modified, are not presented.

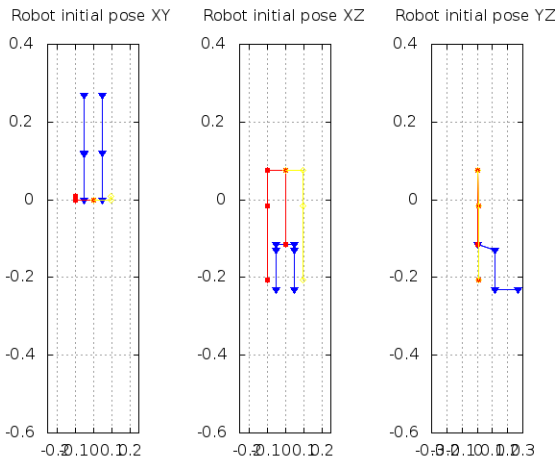



Figure 8-16: *The result of the IKP without constraints, stand-to-sit.*

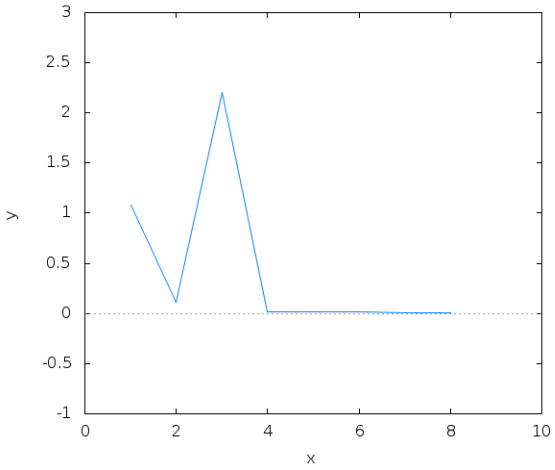



Figure 8-17: *The aggregated error of each iteration, stand-to-sit.*

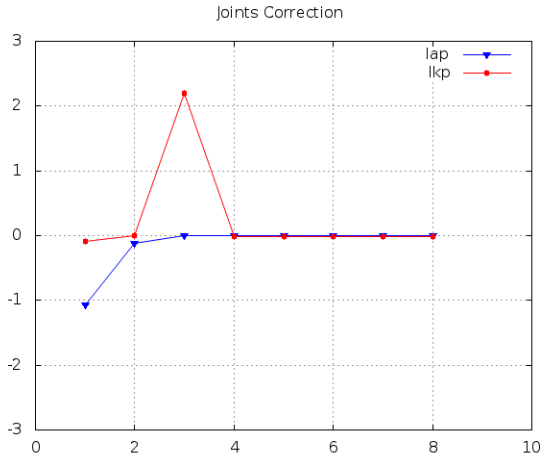



Figure 8-18: *Joints value modification in each iteration, stand-to-sit.*

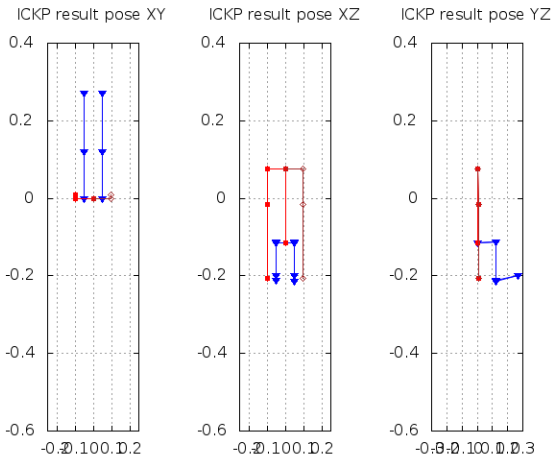



Figure 8-19: *The sitting posture after ICKP calculation with the GCoM constraint*

In Figure 8.2, the posture achieved by applying the ICKP (Algorithm 2) to the same initial posture, support and target conditions is demonstrated. In this case the IKP is solved under the stability condition.

The process converged within 13 iterations as shown in Figure 8-20. The change of the joints values after each iteration are shown in Figure 8-21. It is clear that the solution is converged symmetrically for the left and right sides. Also, It is well demonstrated that the process is not strictly converged. There are some points of divergence, but the process corrects itself and converges again. It might be that the divergence will be repetitive. In that case, the algorithm can be modified, so that if there are few repetitive divergence steps, the initial values of the process are changed to a random numbers within a relevant range. This case was not described in the algorithm.

An interesting aspect of the process is that joints that are in kinematic chains that are not required for the transition are not changed, although they might affect the CoM calculations. The process might change joints values that are not necessary for the transition, in a case they are in the same kinematic chain as those joints that should be changed.

The change in the GCoM location along the convergence process, in relative to the convex hull is described in Figure 8-22.

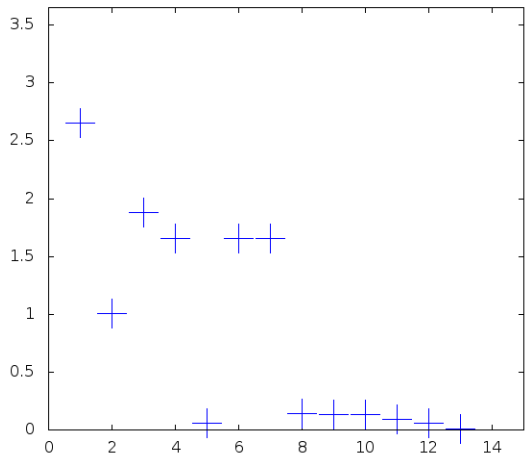



Figure 8-20: *The aggregated error of each iteration of the ICKP for the transition between stand to sit.*

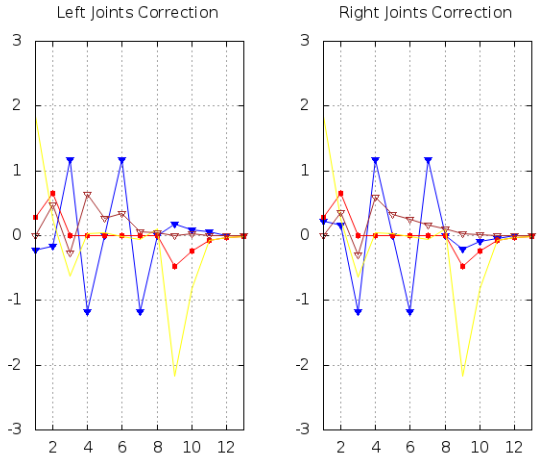



Figure 8-21: *The joints changes in each iteration.*

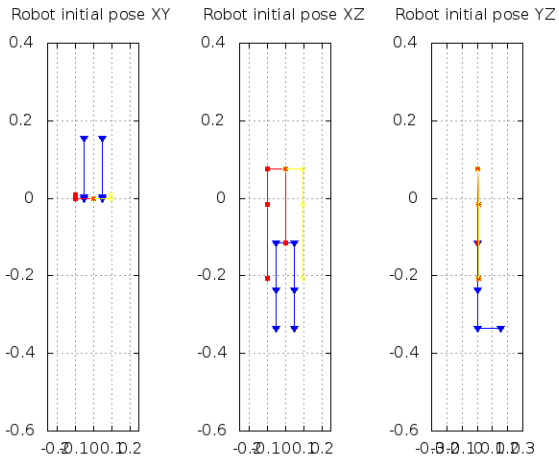



Figure 8-22: *The GCoM location along the process*

## 8.3 Summary

Figures 8-2 through 8-7 demonstrate that while solving the Inverse Kinematic Problem for postures that require moving only few joints belong to a unique kinematic chain that does not affect much the robot stability (e.g. left arm or right arm in sitting posture), the results of the unconstrained solver, the constrained solver and the improved constrained solver are quite similar. Their rate of convergence is different with advantage of the unconstrained solver.

While dealing with postures that require moving joints that affect the robot stability, there is a significant advantage of the constrained IKP solvers in finding a good solution. Also, the aggregated error extinguishes faster.

# Chapter 9

# Conclusions and Future Work

## 9.1 Conclusions

Robot motion planning is a complex task which has to be treated with a rich and modular toolbox that can adjust to each problem the most relevant and efficient solver.

The difference between off-line motion planning and real-time motion planning should be distinguished carefully. Dealing with humanoids, real-time planning is more common and it mainly uses some feedback control strategies to roughly follow a planned paths. Offline planning might be very useful for relatively static environments, such as production halls, or abandoned sites that were previously mapped, like those existing in hazardous sites (e.g. the nuclear facilities in Japan after the earthquake). In these cases the robot motion should and can be planned precisely by using motion planning tools that can achieve accurate trajectory. These planners usually have to account for the positioning uncertainty, and considering it in the planning of the next step.

This research had supplied a motion planning toolbox that includes:

- IKP efficient solver for high DoF devices (Chapters 4 and 5)
- The IKP can be solved under external constraints that are not necessarily kinematics (Chapter 6)
- The stability constraint which is crucial in biped motion, was formalized to be assimilated in the Constrained IKP solver (Chapter 7)
- Some interpolation methods were compared in respect to their differences in the total distance and the accuracy

- The predicted inaccuracy of a motion was formulated as a function of the current posture and the target
- Finally, an algorithm to determine the next via point was developed. Using this algorithm can minimize the number of calculations from one hand, while keeping the robot stability and the accuracy of the path from the other hand.

### 9.1.1 The constrained IKP solver

The IKP problem is actually a set of polynomial of transcendental functions that has to be solved. This is usually an under constrained problem with few solutions. The order of the polynomial depends on the number of joints in the longest kinematic chain of the support- target conditions. Each couple of support and a target point constitutes an additional kinematic constraint. This set of nonlinear equations can be solved by using linearization methods if the nature of the equations is known in advanced. Usually, these functions are oscillating several times in the relevant interval. Finding the suspected zones of oscillations is equivalent of finding the solution proximity on the solutions' envelope.

We proposed a set of improvements to the basic IKP algorithm, that each of them increases the probability of finding the "solution proximity" by a different approach. The first one finds a set of initial joints values that is in already in the proximity of a solution, and it can linearly converge into the required solution. The second one, solves the equations in a multi stage process that each time solves the most significant equations of that stage. The third improvement reduces the number of variables by choosing a set of the most dominant ones for the current iteration. This strategy reduces the DoF of the sub problem to be solved at each phase. Each algorithm represents a family of solutions. The last algorithm proposes how to combine between these strategies so as to choose the optimal one for the current iteration of the problem.

The simulations can demonstrate that although the set of equations is quite complex, and non-linear to a high degree, the process converges to a solution within a reasonable number of iterations.

The convergence of the constrained IKP depends on the type of constraints which should be characterized before solving the problem. In a case that the additional constraints geometric, they might behave similar as the kinematic constraints. Anyhow, there might be a case where the additional constraints convolute with the kinematic conditions. In that case the finding a solution is quite difficult.

The simulations that were performed hint that the constrained IKP solver con-

verged into a solution with / without additional constraints. But, in the case of additional non kinematic constraints, the convergence process is slower. The reason might be the rate of convergence of the non kinematic constraints.

Relating especially the stability constraints, their use is crucial to achieve a stable IKP solution. Stability constraint can be formulated in different ways, especially it depends whether a static or dynamic stability is required. In off-line planning there is no logic to use the dynamic stability constraint, but it is possible. Anyway, the stability constraints must be represented as a function of the kinematic variables $\Theta$, which is not necessary the case in some feedback control methodologies.

The constrained IKP solver is general and supports different types of constraints, among them are some optimization criteria that are common in robotics.

The rate of convergence of the IKP is faster than of the constrained IKP. The reason is that the IKP solved just kinematic equations, that are all polynomials bounded by a certain degree and have same type of variables coupling. Adding the GCoM constraint, which is another type of constraint that actually translates dynamic consideration into geometric terms, changes the convergence process. Divergence might occur not necessarily because of strong coupling, but because of different trends of the solved equations.

Another conclusion is that while the IKP solver never changes the value of joints in chains that are not directly affect the position towards the target, the constrained IKP cannot assures it.

One can conclude that if the transition between two positions does not require big change of the GCoM in relative to the convex hull of the supported polygon, then the IKP is more efficient. But in any case that the change in the GCoM position is large or if the GCoM is close to the border of the convex hull, then the constrained IKP solver with the stability constraint is necessary.

It seems that there is no difference in the accuracy of reaching the target position between the two solvers.

The IKP in humanoid should be solved under the stability constraint. The results demonstrate that the constrained solution is useful for whole body motions in humanoids, as well as for partial body transitions. This general method has good performance and it can be implemented in robot controllers that require real-time responsive.

### 9.1.2 Interpolation during Motion

The resultant Cartesian path of different types of interpolations differs in the accuracy of following the planned path and in the total length of the end effector. As smooth as the interpolation, as deviates the generated path from the planned one. There is a strong correlation between the inaccuracy and the total length of the end effector.

The transition between two postures is a complementary task for robot motion planning which is a discrete task, or more precisely, piecewise continuous task. While feedback control approach keeps stability and generates unconstrained path, interpolating between those points can follow a given path but does not account for stability.

As the gap between two consequent discrete positions increases the runtime performance is improved, but from the other hand, the robot stability and the motion optimality might be reduced.

Chapter 7 discussed how the gap between two consequent positions can be estimated to reduce the probability of losing stability and to minimize the number of IKP calculations, i.e. maximize the intervals between the interpolated positions. This is an off-line task.

Adopting a suitable interpolation strategy might improve the stability of the robot from one hand and from the other hand, it might improve the run-time performance.

As soon as the GCoM of both the initial and the target points are located far from the edges of the convex hull, and the derivatives of the stability constraint in relative to each joint does not exceed these edges, then any interpolation method is satisfied. The difference between the interpolations is crucial in the case that at least one of the positions causes the GCoM to be close to the convex hull borders. Then, using the spline interpolation is recommended, since it generates fewer deviations in the GCoM position during transition.

Also, different interpolations generate different trajectories. Using linear joint interpolation does not guarantee following a certain path, unless the density of the via points is high.

In a case that the GCoM is near the borders of the convex hull of the supported polygon, the constrained IKP with the stability constraint should be applied frequently, which means reduction in performance, or alternatively should be planned in off-line mode.

This research should be extended to examine more interpolation methods and to conclude what is the best method for different stability and feasibility conditions.

## 9.2 Future Work

The rate of convergence of the IKP is faster than of the constrained IKP. The reason is that the IKP solved just kinematic equations, that are all polynomials bounded by a certain degree and have same type of variables coupling. Adding the GCoM constraint, which is another type of constraint that actually translates dynamic consideration into geometric terms, changes the convergence process. Divergence might occur not necessarily because of strong coupling, but because of different trends of the solved equations.

Another conclusion is that while the IKP solver never changes the value of joints in chains that are not directly affect the position towards the target, the constrained IKP cannot assures it.

One can conclude that if the transition between two positions does not require big change of the GCoM in relative to the convex hull of the supported polygon, then the IKP is more efficient. But in any case that the change in the GCoM position is large or if the GCoM is close to the border of the convex hull, then the constrained IKP solver with the stability constraint is necessary.

It seems that there is no difference in the accuracy of reaching the target position between the two solvers. The IKP in humanoid should be solved under the stability constraint. The results demonstrate that the constrained solution is useful for whole body motions in humanoids, as well as for partial body transitions. This general method has good performance and it can be implemented in robot controllers that require real-time responsive. The next development of this method is to analyze its stability sensitivity and improve it, so that the robot will stay stable in the interpolation between two consequent stable postures.

# Appendix A

# The D-H of the Nao robot

We provide a set of kinematic chains and associated D-H matrices that describe the Aldebaran Nao robot. These were constructed based on Aldebaran information [25] about the Nao robot with 26 DoF, and 5 kinematic chains. The zero-point of the robot is its pelvis. The chains are in relative to this zero-point.

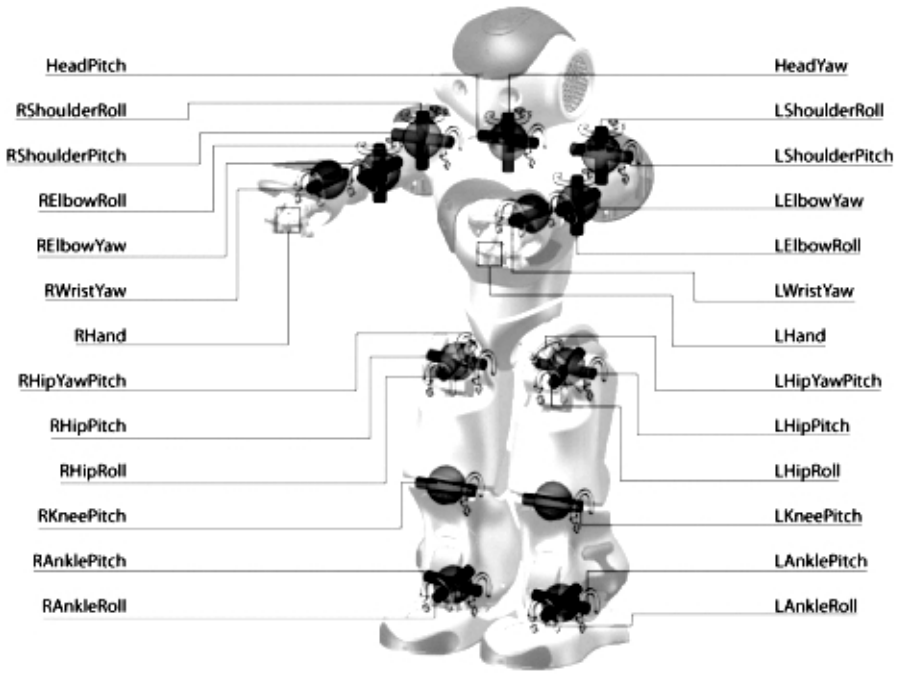



Figure A-1: *The kinematics structure of the Nao robot [Aldebaran Ltd.]*

Table A.1: D-H of the Head chain

| Joint | $a$ | $\alpha$ | $d$ | $\theta$ |
|---|---|---|---|---|
| Base | $A(0, 0, NeckOffsetZ)$ | | | |
| HeadYaw | 0 | 0 | 0 | $\theta_{1,1}$ |
| HeadPitch | 0 | $-\frac{\pi}{2}$ | 0 | $\theta_{1,2} - \frac{\pi}{2}$ |
| Rotation | $R_x(\frac{\pi}{2})R_y(\frac{\pi}{2})$ | | | |

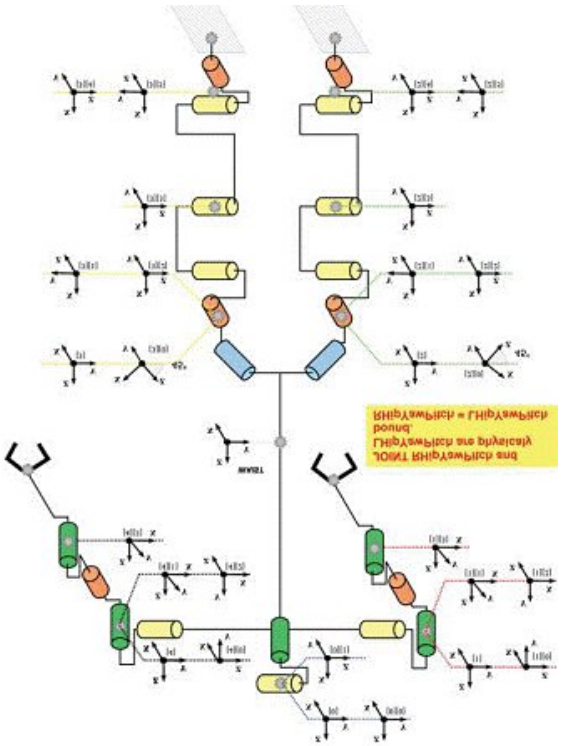


Figure A-2: *Sketch of the Nao's joints [Aldebaran Ltd.]*

Table A.2: D-H of the left leg chain

| Joint | $a$ | $\alpha$ | $d$ | $\theta$ |
|---|---|---|---|---|
| Base | $A(0, HipOffsetY, -HipOffsetZ)$ | | | |
| LHipYawPitch (lhy) | 0 | $-\frac{3\pi}{4}$ | 0 | $\theta_{4,1} - \frac{\pi}{2}$ |
| LHipRoll (lhr) | 0 | $-\frac{\pi}{2}$ | 0 | $\theta_{4,2} + \frac{\pi}{4}$ |
| LHipPitch (lhp) | 0 | $\frac{\pi}{2}$ | 0 | $\theta_{4,3}$ |
| LKneePitch (lkp) | -ThighLength | 0 | 0 | $\theta_{4,4}$ |
| LAnklePitch (lap) | -TibiaLength | 0 | 0 | $\theta_{4,5}$ |
| LAnkleRoll (lar) | 0 | $-\frac{\pi}{2}$ | 0 | $\theta_{4,6}$ |
| Rotation | $R_z(\pi)R_y(-\frac{\pi}{2})$ | | | |
| End effector | $A(0, 0, -FootHeight)$ | | | |

Table A.3: D-H of the right leg chain

| Joint | $a$ | $\alpha$ | $d$ | $\theta$ |
|---|---|---|---|---|
| Base | $A(0, -HipOffsetY, -HipOffsetZ)$ | | | |
| RHipYawPitch (rhy) | 0 | $-\frac{\pi}{4}$ | 0 | $\theta_{5,1} - \frac{\pi}{2}$ |
| RHipRoll (rhr) | 0 | $-\frac{\pi}{2}$ | 0 | $\theta_{5,2} - \frac{\pi}{4}$ |
| RHipPitch (rhp) | 0 | $\frac{\pi}{2}$ | 0 | $\theta_{5,3}$ |
| RKneePitch (rkp) | -ThighLength | 0 | 0 | $\theta_{5,4}$ |
| RAnklePitch (rap) | -TibiaLength | 0 | 0 | $\theta_{5,5}$ |
| RAnkleRoll (rar) | 0 | $-\frac{\pi}{2}$ | 0 | $\theta_{5,6}$ |
| Rotation | $R_z(\pi)R_y(-\frac{\pi}{2})$ | | | |
| End effector | $A(0, 0, -FootHeight)$ | | | |

Table A.4: D-H of the left arm chain

| Joint | $a$ | $\alpha$ | $d$ | $\theta$ |
|---|---|---|---|---|
| Base | $A(0, ShoulderOffsetY + ElbowOffsetY, ShoulderOffsetZ)$ | | | |
| LShoulderPitch (lsp) | 0 | $-\frac{\pi}{2}$ | 0 | $\theta_{2,1}$ |
| LShoulderRoll (lsr) | 0 | $\frac{\pi}{2}$ | 0 | $\theta_{2,2} - \frac{\pi}{2}$ |
| LElbowYaw (ley) | 0 | $-\frac{\pi}{2}$ | UpperArmLength | $-\theta_{2,3}$ |
| LElbowRoll (ler) | 0 | $\frac{\pi}{2}$ | 0 | $\theta_{2,4}$ |
| Rotation | $R_z(\frac{\pi}{2})$ | | | |
| End effector | $A(HandOffsetX + LowerArmLength, 0, 0)$ | | | |

Table A.5: D-H of the right arm chain

| Joint | $a$ | $\alpha$ | $d$ | $\theta$ |
|---|---|---|---|---|
| Base | $A(0, -ShoulderOffsetY - ElbowOffsetY, ShoulderOffsetZ)$ | | | |
| RShoulderPitch (rsp) | 0 | $-\frac{\pi}{2}$ | 0 | $\theta_{3,1}$ |
| RShoulderRoll (rsr) | 0 | $\frac{\pi}{2}$ | 0 | $\theta_{3,2} + \frac{\pi}{2}$ |
| RElbowYaw (rey) | 0 | $-\frac{\pi}{2}$ | -UpperArmLength | $\theta_{3,3}$ |
| RElbowRoll (rer) | 0 | $\frac{\pi}{2}$ | 0 | $\theta_{3,4}$ |
| Rotation | $R_z(\frac{\pi}{2})$ | | | |
| End effector | $A(-HandOffsetX - LowerArmLength, 0, 0)$ | | | |

# Appendix B

# An example kinematic constraint

Define the joints names according to the following table (Table B):

| Joint's Name | Joint's Description |
|---|---|
| lar | Left Ankle Roll |
| rar | Right Ankle Roll |
| lap | Left Ankle Pitch |
| rap | Right Ankle Pitch |
| lkp | Left Knee Pitch |
| rkp | Right Knee Pitch |
| lhy | Left Hip Yaw |
| rhy | Right Hip Yaw |
| lhr | Left Hip Roll |
| rhr | Right Hip Roll |
| lhp | Left Hip Pitch |
| rhp | Right Hip Pitch |
| lsp | Left Shoulder Pitch |
| rsp | Right Shoulder Pitch |
| lsr | Left Shoulder Roll |
| rsr | Right Shoulder Roll |
| ley | Left Elbow Yaw |
| rey | Right Elbow Yaw |
| ler | Left Elbow Roll |
| rer | Right Elbow Roll |
| lwy | Left Wrist Yaw |
| rwy | Right Wrist Yaw |
| hp | Head Pitch |
| hy | Head Yaw |

Table B.1: Joints name abbreviation

Then, define the following variables:

$$x_1 = \frac{\sin(lhp)\sin(lhy)}{\sqrt{2}}$$

$$x_2 = \frac{\cos(lhp)\sin(lhy)}{\sqrt{2}}$$

$$x_3 = x_1 \ \sin(lhr)$$

$$x_4 = x_2 \ \sin(lhr)$$

$$x_5 = x_1 \ \cos(lhr)$$

$$x_6 = x_2 \ \cos(lhr)$$

$$x_7 = \cos(lhr) \ \sin(lhy)/\sqrt{(2)}$$

$$x_8 = \sin(lhr) \ \sin(lhy)/\sqrt{(2)}$$

$$x_9 = \cos(lhp) \ \sin(lhr)$$

$$x_{10} = \cos(lhp) \ \cos(lhr)$$

$$x_{11} = 0.5 \ \cos(lhy) + 0.5$$

$$x_{12} = 0.5 \ \cos(lhy) - 0.5$$

$$x_{13} = \sin(lhp) \ \sin(lhr)$$

$$x_{14} = \sin(lhp) \ \cos(lhr)$$

$$x_{15} = \cos(lap) \ \cos(lar)$$

$$x_{16} = \sin(lhp) \ \cos(lhy)$$

$$x_{17} = \cos(lhp) \ \cos(lhy)$$

$$x_{18} = \sin(lap)\ \cos(lar)$$

$$x_{19} = \cos(rey)\ \cos(rsp)$$

$$x_{20} = x_1 - x_9\ x_{11} + x_{10}\ x_{12}$$

$$x_{21} = -x_2 - x_{13}\ x_{11} + x_{14}\ x_{12}$$

$$x_{22} = -x_4 - x_6 - x_{16}$$

$$x_{23} = -x_3 - x_5 + x_{17}$$

$$x_{24} = x_{20}\ \sin(lkp) + x_{21}\ \cos(lkp)$$

$$x_{25} = x_{20}\ \cos(lkp) - x_{21}\ \sin(lkp)$$

$$x_{26} = x_{22}\ \cos(lkp) - x_{23}\ \sin(lkp)$$

$$x_{27} = x_{22}\ \sin(lkp) + x_{23}\ \cos(lkp)$$

$$x_{28} = \cos(lap)\ x_{24} + \sin(lap)\ x_{25}$$

$$x_{29} = -x_1 + x_{10}\ x_{11} - x_9\ x_{12}$$

$$x_{30} = x_2 + x_{14}\ x_{11} - x_{13}\ x_{12}$$

$$x_{31} = x_7 - x_8$$

$$x_{32} = x_{29}\ \sin(lkp) + x_{30}\ \cos(lkp)$$

$$x_{33} = x_{29}\ \cos(lkp) - x_{30}\ \sin(lkp)$$

$$x_{34} = \cos(lhr)\ x_{11} + \sin(lhr)\ x_{12}$$

$$x_{35} = \sin(lap)\ x_{26} + \cos(lap)\ x_{27}$$

$$x_{36} = \sin(lhr)\ x_{11} + \cos(lhr)\ x_{12}$$

$$x_{37} = \cos(lap)\ x_{32} + \sin(lap)\ x_{33}$$

$$x_{38} = \cos(lap)\ x_{24} + \sin(lap)\ x_{25}$$

$$x_{39} = x_{15}\ x_{26} - x_{18}\ x_{27} - \sin(lar)\ x_{31}$$

$$x_{40} = -x_{18}\ x_{24} + x_{15}\ x_{25} - \sin(lar)\ x_{34}$$

$$x_{41} = -x_{18}\ x_{32} + x_{15}\ x_{33} - \sin(lar)\ x_{36}$$

$$x_{42} = x_{41}\ x_{35} - x_{37}\ x_{39}$$

$$x_{43} = -x_{20}\ \sin(lkp) + x_{30}\ \cos(lkp)$$

$$x_{44} = -x_{18}\ x_{43} + x_{15}\ x_{43} - \sin(lar)\ x_{36}$$

$$x_{45} = -\sin(lap)\ \sin(lar)\ x_{43} + \cos(lap)\ \sin(lar)\ x_{43} + \cos(lar)\ x_{36}$$

$$x_{46} = \cos(lap)\ \sin(lar)\ x_{26} - \sin(lap)\ \sin(lar)\ x_{27} + \cos(lar)\ x_{31}$$

$$x_{47} = x_{45}\ x_{39} - x_{44}\ x_{46}$$

$$x_{48} = (\cos(lap) + \sin(lap))\ x_{43}$$

$$x_{49} = x_{20}\ \sin(lkp) + x_{21}\ \cos(lkp)$$

$$x_{50} = -\sin(lap)\ \sin(lar)\ x_{49} + \cos(lap)\ \sin(lar)\ x_{25} + \cos(lar)\ \ x_{34}$$

$$x_{51} = -x_{18}\ x_{49} + x_{15}\ x_{25} - \sin(lar)\ x_{34}$$

$$x_{52} = -\sin(lap)\ x_{24} + \cos(lap)\ x_{25}$$

$$x_{53} = \sin(lar)\ x_{52} + \cos(lar)\ x_{34}$$

$$x_{54} = -\sin(lap)\ x_{32} + \cos(lap)\ x_{33}$$

$$x_{55} = \sin(lar)\ x_{54} + \cos(lar)\ x_{36}$$

$$x_{56} = x_{55}\ x_{39} - x_{41}\ x_{46}$$

$$x_{57} = x_{37}\ x_{46} - x_{55}\ x_{35}$$

$$x_{58} = x_{28}\ x_{56} + x_{40}\ x_{57} + x_{53}\ x_{42}$$

$$x_{59} = 0.1506\ x_{37} - 0.1\ x_{33} - 0.12\ x_{29} + 0.005\ x_{30}$$

If the target of the $x$ coordinate of the right palm is $X_t$ then the kinematic constraint of the $x$ coordinate is:

$$
\begin{aligned}
X_t = &((x_{28}x_{39} - x_{40}x_{35})(-0.1925(x_{19} - \sin(rey)\sin(rsp)\sin(rsr)) + 0.009\sin(rsp)\cos(rsr) \\
&-0.09\cos(rsp) + 0.075))/(x_{28}((-\sin(lap)\sin(lar)x_{32} + \cos(lap)\sin(lar)x_{33} + \cos(lar)x_{36})x_{39} \\
&-x_{41}(\cos(lap)\sin(lar)x_{26} - \sin(lap)\sin(lar)x_{27} + \cos(lar)x_{31})) + x_{40}(x_{37}(\cos(lap)\sin(lar)x_{26} \\
&-\sin(lap)\sin(lar)x_{27} + \cos(lar)x_{31}) - (-\sin(lap)\sin(lar)x_{32} + \cos(lap)\sin(lar)x_{33} \\
&+\cos(lar)x_{36})x_{35}) + (-\sin(lap)\sin(lar)x_{24} + \cos(lap)\sin(lar)x_{25} + \cos(lar)x_{34})x_{42}) \\
&+((x_{37}x_{40} - x_{41}x_{38})(-0.1925(-\sin(rey)\cos(rsp)\sin(rsr) - \cos(rey)\sin(rsp)) \\
&+0.009\cos(rsp)\cos(rsr) + 0.09\sin(rsp)))/(x_{38}((-\sin(lap)\sin(lar)x_{32} + \cos(lap)\sin(lar)x_{33} \\
&+\cos(lar)x_{36})x_{39} - x_{41}(\cos(lap)\sin(lar)x_{26} - \sin(lap)\sin(lar)x_{27} + \cos(lar)x_{31})) \\
&+x_{40}(1.0x_{37}(\cos(lap)\sin(lar)x_{26} - \sin(lap)\sin(lar)x_{27} + \cos(lar)x_{31}) \\
&-(-\sin(lap)\sin(lar)x_{32} + \cos(lap)\sin(lar)x_{33} + \cos(lar)x_{36})x_{35}) + (-\sin(lap)\sin(lar)x_{24} \\
&+\cos(lap)\sin(lar)x_{25} + \cos(lar)x_{34})x_{42}) + (x_{42}(-0.009\sin(rsr) + 0.1925\sin(rey)\cos(rsr) + 0.098)) \\
&/(x_{38}((-\sin(lap)\sin(lar)x_{32} + \cos(lap)\sin(lar)x_{33} + \cos(lar)x_{36})(x_{15}((-x_4 - x_6 - x_{16}) \\
&\cos(lkp) - x_{23}\sin(lkp)) - x_18x_{27} - \sin(lar)x_{31}) - x_{41}(\cos(lap)\sin(lar)x_{26} \\
&-\sin(lap)\sin(lar)x_{27} + \cos(lar)x_{31})) + x_{40}(x_{37}(\cos(lap)\sin(lar)x_{26} - \sin(lap)\sin(lar)x_{27} \\
&+\cos(lar)x_{31}) - (-\sin(lap)\sin(lar)x_{32} + \cos(lap)\sin(lar)x_{33} + \cos(lar)x_{36})x_{35}) \\
&+(-\sin(lap)\sin(lar)x_{24} + \cos(lap)\sin(lar)x_{25} + \cos(lar)x_{34})x_{42}) \\
&+((-x_{38}((x_{15}x_{26} - x_18x_{27} - \sin(lar)x_{31})(0.15063x_{37} - 0.1((-x_1 + x_10x_1 - x9x_{12})\cos(lkp) \\
&-x_30\sin(lkp)) - 0.12x_{29} + 0.005x_30 - 0.115) - x_{41}(0.15063x_{35} - 0.1x_{26} + 0.005x_{23} - 0.12x_{22})) \\
&+x_{40}(x_{35}(0.15063x_{37} - 0.1x_{33} - 0.12x_{29} + 0.005x_30 - 0.115) - x_{37}(0.15063x_{35} \\
&-0.1((-x4 - -x6 - x_16)\cos(lkp) - x_{23}\sin(lkp)) + 0.005x_{23} - 0.12x_{22})) \\
&-(0.15063x_{38} - 0.1x_{25} - 0.12x_{20} + 0.005x_{21} - 0.05)x_{42})) \\
&/(x_{38}((-\sin(lap)\sin(lar)x_{32} + \cos(lap)\sin(lar)x_{33} + cos(lar)x_{36})x_{39} \\
&-x_{41}(\cos(lap)\sin(lar)x_{26} - \sin(lap)\sin(lar)x_{27} + \cos(lar)x_{31})) \\
&+x_{40}(x_{37}(\cos(lap)\sin(lar)x_{26} - \sin(lap)\sin(lar)x_{27} + \cos(lar)x_{31}) \\
&-(-\sin(lap)\sin(lar)x_{32} + \cos(lap)\sin(lar)x_{33} + \cos(lar)x_{36})x_{35}) + (-\sin(lap)\sin(lar)x_{24} \\
&+\cos(lap)\sin(lar)x_{25} + \cos(lar)x_{34})x_{42})
\end{aligned}
$$

The target of $y$ coordinate of the right palm is as follows:

$$
\begin{aligned}
Y_t = &((x_{40}x_{46} - (-\sin(lap)\sin(lar)x_{24} + \cos(lap)\sin(lar)x_{25} \\
&+ \cos(lar)x_{34})x_{39)}(-0.192(x_{19} - \sin(rey)\sin(rsp)\sin(rsr)) + 0.009\sin(rsp)\cos(rsr) \\
&-0.09\cos(rsp) + 0.075))/(x_{28}x_{47} + x_{40}(x_{48}x_{46} - x_{45}x_{35}) \\
&+x_{50}(x_{44}x_{35} - x_{48}(x_{15}(x_{22} - x_{23}\sin(lkp)) - x_{18}x_{27} - \sin(lar)x_{31}))) \\
&+((x_{44}x_{50} - (-\sin(lap)\sin(lar)x_{43} + \cos(lap)\sin(lar)x_{43} \\
&+ \cos(lar)x_{36})x_{51})(-0.192(-\sin(rey)\cos(rsp)\sin(rsr) - \cos(rey)\sin(rsp)) \\
&+0.009\cos(rsp)\cos(rsr) + 0.09\sin(rsp))) \\
&/(x_{28}(x_{45}x_{39} - x_{44}(\cos(lap)\sin(lar)(x_{22} - x_{23}\sin(lkp)) \\
&- \sin(lap)\sin(lar)x_{27} + \cos(lar)x_{31})) + x_{51}(x_{48}(\cos(lap)\sin(lar)(x_{22}\cos(lkp) - x_{23}\sin(lkp)) \\
&- \sin(lap)\sin(lar)x_{27} + \cos(lar)x_{31}) - x_{45}x_{35}) + x_{50}(x_{44}x_{35} - x_{48}x_{39})) \\
&+(x_{47}(-0.009\sin(rsr) + 0.192\sin(rey)\cos(rsr) + 0.098)) \\
&/((\cos(lap)x_{24} + \sin(lap)x_{25})((-\sin(lap)\sin(lar)x_{43} + \cos(lap)\sin(lar)x_{43} + \cos(lar)x_{36})x_{39} - x_{44}x_{46}) \\
&+x_{51}(x_{48}(\cos(lap)\sin(lar)x_{26} - \sin(lap)\sin(lar)x_{27} + \cos(lar)x_{31}) - x_{45}x_{35}) + x_{50}(x_{44}x_{35} - x_{48}x_{39})) \\
&+((-x_{51}(x_{46}(0.151x_{48} - 0.1x_{43} - 0.12 - x_{20} + 0.005x30 - 0.115) \\
&-x_{45}(0.151x_{35} - 0.1x_{26} + 0.005x_{23} - 0.12x_{22})) + x_{50}((x15x_{26} - x18(x_{22}\sin(lkp) \\
&+(-x_3 - x_5 + x_{17})\cos(lkp)) - \sin(lar)x_{31})(0.151x_{48} - 0.1x_{43} - 0.12 - x_{20} + 0.005x_{30} - 0.115) \\
&-x_{44}(0.151x_{35} - 0.1x_{26} + 0.005(-x_{23}) - 0.12x_{22})) + (0.151(\cos(lap)x_{49} + \sin(lap)x_{25}) \\
&-0.1x_{25} - 0.12x_{20} + 0.005x21 - 0.05) - x_{47})) \\
&/((\cos(lap)x_{24} + \sin(lap)x_{25})(x_{46}x_{39} - (-x_{18}x_{43} + x_{15}x_{43} - \sin(lar)x_{36}) \\
&(\cos(lap)\sin(lar)x_{26} - \sin(lap)\sin(lar)(x_{22}\sin(lkp) + x_{23}\cos(lkp)) + \cos(lar)x_{31})) \\
&+x_{51}(x_{48}x_{46} - x_{45}x_{35}) + x_{50}(x_{44}x_{35} - x_{48}x_{39}))
\end{aligned}
$$

And finally, the target of $z$ coordinate of the right palm is as follows:

$$
\begin{aligned}
Z_t = &((x_{53}x_{35} - x_{28}x_{46})(-0.1925(x_{19} - \sin(rey)\sin(rsp)\sin(rsr)) + 0.009\sin(rsp)\cos(rsr) \\
&-0.09\cos(rsp) + 0.075))/x_{58} \\
&+((x_{55}x_{28} - x_{37}x_{53})(-0.1925(-\sin(rey)\cos(rsp)\sin(rsr) - \cos(rey)\sin(rsp)) + 0.009\cos(rsp) \\
&\cos(rsr) + 0.09\sin(rsp)))/x_{58} \\
&+((x_{37}(\cos(lap)\sin(lar)(x_{22}\cos(lkp) - (-\sin(lhp)x_8 - \sin(lhp)x_7 + x_{17})\sin(lkp)) \\
&- \sin(lap)\sin(lar)x_{27} + \cos(lar)x_{31}) - x_{55}x_{35}) \\
&(-0.009\sin(rsr) + 0.1925\sin(rey)\cos(rsr) + 0.098))/x_{58} \\
&+((x_{28}(x_{46}x_{59} - x_{55}(0.1506x_{35} - 0.1x_{26} + 0.005x_{23} - 0.12x_{22})) \\
&-x_{53}(x_{35}x_{59} - x_{37}(0.1506x_{35} - 0.1x_{26} + 0.005x_{23} - 0.12x_{22})) \\
&-(0.1506x_{28} - 0.1x_{25} - 0.12x_{20} + 0.005x_{21} - 0.05)x_{57}))/x_{58}
\end{aligned}
$$

# Appendix C

# The Interpolated Path Measurement Applied to the Nao Robot

Based on Aldebaran information about the Nao robot with 26 DoF, and 5 kinematic chains. The zero-point of the robot is its pelvis. The chains are in relative to this zero-point. Without lose of generality, the analyzed kinematics compounds of two chains: the right palm and the left foot. The root end-effector is in the right palm and support point is in the left foot.

Apply Equation 7.1 to the right palm of the Nao, as was described in Eq. 7.4, we get:

$$\begin{aligned} D_{rpalm-lfoot} = & [0.029 \sin(\theta_{rey}) \sin(\theta_{rsp}) \sin(\theta_{rsr}) - 0.002 \sin(\theta_{rsr}) \\ & +0.001 \sin(\theta_{rsp}) \cos(\theta_{rsr}) + 0.038 \sin(\theta_{rey}) \cos(\theta_{rsr}) + 0.035 \cos(\theta_{rey}) \\ & -0.029 \cos(\theta_{rey}) \cos(\theta_{rsp}) - 0.013 \cos(\theta_{rsp}) + 0.060]^{0.5} \end{aligned} \tag{C.1}$$

Applying function 7.6 to the kinematic chain beginning at the left foot and ending at the right palm, we get:

$$L_{palm} = \int^{t} \sqrt{f_{rpalm-lfoot}(\bar{\theta})} dt \tag{C.2}$$

Now, the following variables are defined,

$$\begin{aligned} x_9 &= \cos(\theta_{rey}) \sin(\theta_{rsp}) + \sin(\theta_{rey}) \cos(\theta_{rsp}) \\ x_{10} &= \sin(\theta_{rey}) \sin(\theta_{rsp}) \cos(\theta_{rsr}) \\ x_{11} &= \sin(\theta_{rsp}) \sin(\theta_{rsr}) - \cos(\theta_{rsp}) \cos(\theta_{rsr}) - \sin(\theta_{rsp}) \\ x_{12} &= \sin(\theta_{rey}) \sin(\theta_{rsr}) + \cos(\theta_{rey}) \cos(\theta_{rsr}) \end{aligned} \tag{C.3}$$

Then, Then, the continuous function of the path length becomes to be:

$$L = \int^{t} \sqrt{(dx^2 + dy^2 + dz^2)}dt = \int^{t} \sqrt{\frac{\partial X^2}{\partial \bar{\theta}}\frac{\partial \bar{\theta}}{\partial t} + \frac{\partial Y^2}{\partial \bar{\theta}}\frac{\partial \bar{\theta}}{\partial t} + \frac{\partial Z^2}{\partial \bar{\theta}}\frac{\partial \bar{\theta}}{\partial t}}dt \tag{C.4}$$

The integration of the function will be applied numerically.

$$\begin{aligned} &f_{rpalm-lfoot}(\bar{\theta}) = 0.0289\left[x_9\left(1 + \sin\left(\theta_{rsr}\right)\right) + x_{10}\right] - 0.0013x_{11} - 0.0377x_{12} \\ &-0.0018\cos\left(\theta_{rsr}\right) - 0.0346\sin\left(\theta_{rey}\right) \end{aligned} \tag{C.5}$$

In the case of spatial distance interpolation, the value of $\theta_j\left(t\right)$ along the path is quite complex and can be represented as:

$\theta_j\left(t\right) = a_j t + \theta_0$

Applying this function to the kinematic chain from the left foot to the right palm, we get:

$$L_{palm-lfoot} = \int^{t} \sqrt{f_{palm}(\bar{\theta})}dt \tag{C.6}$$

Again, this value will be calculated numerically.

# Appendix D

# Bibliography


[1] http://www.easyvigour.net.nz/pilates/hcballbeg.htm.

[2] J. Angeles. On the numerical solution of the inverse kinematic problem. *The International Journal of Robotics Research*, 4(2):21–37, 1985.

[3] M. Avriel. *Nonlinear Programming: Analysis and Methods*. Dover Books on Computer Science Series. Dover Publications, 2003.

[4] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483, dec 1996.

[5] C. Belta, V. Isler, and G. J. Pappas. Discrete abstractions for robot motion planning and control in polygonal environments. *IEEE Transactions on Robotics*, 21:864–874, 2004.

[6] C. Belta and V. Kumar. Motion generation for formations of robots: A geometric approach. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1245–1250 vol.2, 2001.

[7] D. Bertram, J. Kuffner, R. Dillmann, and T. Asfour. An integrated approach to inverse kinematics and path planning for redundant manipulators. In *IEEE International Conference on Robotics and Automation*, pages 1874–1879, May 2006.

[8] S. R. Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. *IEEE Journal of Robotics and Automation*, 2004.

[9] F. Chapelle and P. Bidaud. A closed form for inverse kinematics approximation of general 6r manipulators using genetic programming. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, volume 4, pages 3364–3369, 2001.

[10] R. I. Charles F., P.-P. J. Sloan, and M. F. Cohen. Artist-directed inverse-kinematics using radial basis function interpolation. *Computer Graphics Forum*, 20(3):239–250, 2001.

[11] H. Choset. *Principles of Robot Motion: Theory, Algorithms, and Implementation*. A Bradford book. Prentice Hall, 2005.

[12] J. Clark and M. Cutkosky. Stability measure comparison for the design of a dynamic running robot. In M. Tokhi, G. Virk, and M. Hossain, editors, *Climbing and Walking Robots*, pages 261–268. Springer Berlin Heidelberg, 2006.

[13] E. V. Cuevas, D. Zaldívar, and R. Rojas. Incremental fuzzy control for a biped robot balance. *International Conference on Robotics and Applications*, 2005.

[14] B. Daya, S. Khawandi, and M. Akoum. Applying neural network architecture for inverse kinematics problem in robotics. *Journal of Software Engineering and Applications*, 3(3):230–239, 2010.

[15] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*. Computational Principles of Mobile Robotics. Cambridge University Press, 2010.

[16] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4):551–559, 1983.

[17] Z. Fu, W. Yang, and Z. Yang. Solution of inverse kinematics for 6r robot manipulators with offset wrist based on geometric algebra. *Journal of mechanisms and robotics*, 5(3):310081–310087, August 2013.

[18] J. Q. Gan, E. Oyama, E. M. Rosales, and H. Hu. A complete analytical solution to the inverse kinematics of the pioneer 2 robotic arm. *Robotica*, 23(1):123–129, Jan 2005.

[19] M. J. Gielniak, C. K. Liu, and A. L. Thomaz. Stylized motion generalization through adaptation of velocity profiles. In C. A. Avizzano and E. Ruffaldi, editors, *IEEE International Workshop on Robots and Human Interactive Communications*, pages 304–309. IEEE, 2010.

[20] A. Goldenberg, B. Benhabib, and R. Fenton. A complete generalized solution to the inverse kinematics of robots. *IEEE Journal of Robotics and Automation*, 1(1):14–20, Mar 1985.

[21] C. Gosselin and J. Angeles. A global performance index for the kinematic optimization of robotic manipulators. *Journal of Mechanical Design*, 113(3):220–226, 1991.

[22] A. Goswami. Foot rotation indicator (FRI) point: A new gait planning tool to evaluate postural stability of biped robots. *In IEEE International Conference on Robotics and Automation*, pages 47–52, 1999.

[23] A. Goswami. Biped locomotion: Stability, analysis and control. *International Journal on Smart Sensing and Intelligent Systems*, 1(1), March 2008.

[24] A. Goswami and V. Kallem. Rate of change of angular momentum and balance maintenance of biped robots. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 3785–3790, 2004.

[25] D. Gouaillier and P. Blazevic. A mechatronic platform, the aldebaran robotics humanoid robot. In *IEEE Industrial Electronics, IECON 2006-32nd Annual Conference on*, pages 4049–4053. IEEE, 2006.

[26] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović. Style-based inverse kinematics. In *ACM Special Interest Group on Computer Graphics*, SIGGRAPH '04, pages 522–531, NY, USA, 2004.

[27] T. Horsch and B. JÃ$\frac{1}{4}$ttler. Cartesian spline interpolation for industrial robots. *Computer-Aided Design*, 30(3):217 – 224, 1998. <ce:title>Motion Design and Kinematics</ce:title>.

[28] A. J. Ijspeert. Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21(4):642 – 653, 2008.

[29] Z. Jin and Q. Ge. Constrained motion interpolation for planar open kinematic chains. *Mechanism and Machine Theory*, 45(11):1721 – 1732, 2010.

[30] J.J.Craig. *Introduction to Robotics*. Pearson Education, 3rd edition, 2005.

[31] H. S. Jo and N. Mir-Nasiri. Stability control of minimalist bipedal robot in single support phase. *International Symposium on Robotics and Intelligent Sensors*, 41(0):113 – 119, 2012.

[32] K. Jolly, R. S. Kumar, and R. Vijayakumar. A bezier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits. *Robotics and Autonomous Systems*, 57(1):23 – 33, 2009.

[33] J.W.Grizzle, C. Chevallereau, A. Ames, and R. Sinnet. 3D bipedal robotic walking models, feedback control and open problems. In *Symposium on Nonlinear Control Systems*, Sept. 2010.

[34] S. Kajita, T. Yamaura, and A. Kobayashi. Dynamic walking control of a biped robot along a potential energy conserving orbit. *IEEE Transactions on Robotics and Automation*, 8(4):431–438, 1992.

[35] I. Kato. Development of Wabot1. *Biomechanism2*, pages 173–214, 1973. Tokyo.

[36] K. Kazerounian. On the numerical inverse kinematics of robotic manipulators. *Journal of Mechanical Design*, 109, 1987.

[37] W. Khalil and E. Dombre. *Modeling, Identification and Control of Robots*. Kogan Page Science paper edition. Elsevier Science, 2004.

[38] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *IEEE International Conference on Robotics and Automation.*, volume 2, pages 500–505, Mar 1985.

[39] R. Kőker. A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization. *Journal of Information Science*, 222:528–543, feb 2013.

[40] J.-H. Kim, J.-Y. Kim, and J.-H. Oh. Adjustment of home posture of biped humanoid robot using sensory feedback control. *Journal of Intelligent and Robotic Systems*, 51:421–438, April 2008.

[41] K. Kira and L. A. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI'92, pages 129–134. AAAI Press, 1992.

[42] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artif. Intell.*, 97(1-2):273–324, dec 1997.

[43] T. Komura, H. Leung, S. Kudoh, and J. Kuffner. A feedback controller for biped humanoids that can counteract large perturbations during gait. In *Proceedings*

*of the 2005 IEEE International Conference on Robotics and Automation*, pages 1989–1995, 2005.

[44] H. Kurniawati, Y. Du, D. Hsu, and W. Lee. *Motion Planning under Uncertainty for Robotic Tasks with Long Time Horizons*. Springer Tracts in Advanced Robotics. Springer Berlin Heidelberg, 2011.

[45] P. Last, J. Hesselbach, and N. Plitea. An extended inverse kinematic model of the hexa-parallel-robot for calibration purposes. In *IEEE International Conference on Mechatronics and Automation*, volume 3, pages 1294–1299, 2005.

[46] J. Laumond, S. Sekhavat, and F. Lamiraux. *Guidelines in nonholonomic motion planning for mobile robots*, volume 229 of *Lecture Notes in Control and Information Sciences*. Springer Berlin Heidelberg, 1998.

[47] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[48] J. H. Lee and S. N. Yang. Shape preserving and shape control with interpolating bézier curves. *Journal of Computational and Applied Mathematics*, 28(0):269 – 280, 1989.

[49] S. Lindemann, I. Hussein, and S. LaValle. Real time feedback control for nonholonomic mobile robots with obstacles. In *45th IEEE Conference on Decision and Control*, pages 2406–2411, Dec 2006.

[50] A. Loría, E. Panteley, and H. Nijmeijer. A remark on passivity-based and discontinuous control of uncertain nonlinear systems. *Automatica*, 37(9):1481–1487, 2001.

[51] V. Lugade, V. Lin, and L.-S. Chou. Center of mass and base of support interaction during gait. *Gait and Posture*, 33(3):406–411, 2011.

[52] E. Marder and R. L. Calabrese. Principles of rhythmic motor pattern generation. *Physiological reviews*, 76(3):687–717, 1996.

[53] E. Masehian and D. Sedighizadeh. Classic and heuristic approaches in robot motion planning – a chronological review. In *Proc. World Academy of Science, Engineering and Technology*, pages 101–106, 2007.

[54] L. Mateos, K. Zhou, and M. Vincze. Towards efficient pipe maintenance: Dewalop in-pipe robot stability controller. In *International Conference on Mechatronics and Automation (ICMA)*, pages 1–6, Aug 2012.

[55] M. Meredith and S. Maddock. Real-time inverse kinematics: The return of the jacobian. Department of Computer Science Research Memorandum CS-04-06, University of Sheffield, 2005.

[56] A. C. Nearchou. Solving the inverse kinematics problem of redundant robots operating in complex environments via a modified genetic algorithm. *Mechanism and Machine Theory*, 33(3):273 – 292, 1998.

[57] R. Paul. *Robot Manipulators: Mathematics, Programming, and Control : the Computer Control of Robot Manipulators*. Artificial Intelligence Series. MIT Press, 1981.

[58] D. L. Pieper. The kinematics of manipulators under computer control. Technical Report STAN-CS-68-116, Stanford University (Stanford,CA US), 1968.

[59] S. K. T. K. H. A. N. K. e. a. Q. Huang, K. Yokoi. Planning walking patterns for a biped robot. In *IEEE Trans. on Robotics and Automation*, pages 280–289, 2001.

[60] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[61] M. Riley and C. G. Atkeson. Methods for motion generation and interaction with a humanoid robot: Case studies of dancing and catching. In *Proceedings 2000 Workshop on Interactive Robotics and Entertainment, Robotics Institute, Carnegie Mellon University*, pages 35–42, 2000.

[62] A. Robotics. Naoware documentation. Technical report, 2009.

[63] M. Ruchanurucks, S. Nakaoka, S. Kudoh, and K. Ikeuchi. Humanoid robot motion generation with sequential physical constraints. In *Proceedings 2006 IEEE International Conference on Robotics and Automation*, pages 2649–2654, 2006.

[64] T. Rudny. Universal inverse kinematics problem solver. In *Proceedings of the 16th International Conference on Systems Science*. 2007.

[65] P. Sardain and G. Bessonnet. Forces acting on a biped robot. center of pressure zero moment point. *IEEE Transactions on Systems, Man, and Cybernetics Part A*, pages 630–637, Sept. 2004.

[66] W. Schelter. *Maxima Manual*. SourceForge, 2004.

[67] L. Sciavicco, B. Siciliano, and B. Sciavicco. *Modelling and Control of Robot Manipulators*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2nd edition, 2000.

[68] J. Selig. *Geometric Fundamentals of Robotics*. Monographs in Computer Science. Springer London, Limited, 2007.

[69] L. Sentis. Compliant control of whole-body multi-contact behaviors in humanoid robots. In K. Harada, E. Yoshida, and K. Yokoi, editors, *Motion Planning for Humanoid Robots*, pages 29–66. Springer London, 2010.

[70] L. Sentis, J. Park, and O. Khatib. Compliant control of multicontact and center-of-mass behaviors in humanoid robots. *IEEE Transactions on Robotics*, 26(3):483–501, June 2010.

[71] Y. Shen, K. Huper, and F. Silva Leite. Smooth interpolation of orientation by rolling and wrapping for robot motion planning. In *Proceedings 2006 IEEE International Conference on Robotics and Automation*, pages 113–118, 2006.

[72] B. Sheng, M. Huaqing, and C. Qiang. Review of humanoid robot feedback control gait planning. *Computer Engineering and Applications*, 47(7):30, 2011.

[73] K. Shoemake. Animating rotation with quaternion curves. *ACM Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH) Computer Graphics*, 19(3):245–254, jul 1985.

[74] B. Siciliano and O. Khatib. *Handbook of Robotics*. Gale virtual reference library. Springer, 2008.

[75] G. K. Singh and J. Claassens. An analytical solution for the inverse kinematics of a redundant 7DoF manipulator with link offsets. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 2976–2982. IEEE, 2010.

[76] R. Smith, M. Self, and P. Cheeseman. *Autonomous Robot Vehicles*, chapter Estimating Uncertain Spatial Relationships in Robotics, pages 167–193. Springer-Verlag Inc., NY, USA, 1990.

[77] S. Spong M.W., Hutchinson and V. M. *Robot Modeling and Control*. John Wiley and Sons, 2006.

[78] B. Stephens and C. Atkeson. Dynamic balance force control for compliant humanoid robots. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1248–1255, Oct 2010.

[79] S. A. Stuvel. Stride space: Humanoid walking animation interpolation using 3D delaunay databases. Technical report, Utrecht University, 2010.

[80] R. Tedrake. LQR-trees: Feedback motion planning on sparse randomized trees. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.

[81] G. Tevatia and S. Schaal. Inverse kinematics for humanoid robots. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 294–299, 2000.

[82] U. Thomas, B. Finkemeyer, T. Króger, and F. M. Wahl. Error-tolerant execution of complex robot tasks based on skill primitives. In *International Conference on Robotics and Automation (ICRA)*, pages 3069–3075, 2003.

[83] A. Timcenko and P. K. Allen. Modeling dynamic uncertainty in robot motions. *International Conference on Robotics and Automation (ICRA)*, May 1993.

[84] D. Tolani, A. Goswami, and N. I. Badler. Real-time inverse kinematics techniques for anthropomorphic limbs. *Journal of Graphical Models and Image Processing*, 62(5):353–388, sep 2000.

[85] P. Vadakepat and D. Goswami. Biped locomotion: Stability, analysis and control. In *International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS)*, Palmerston North, New Zealand, 2007.

[86] P. Viviani and T. Flash. Minimum-jerk, two-thirds power law, and isochrony: converging approaches to movement planning. *Journal of Experimental Psychology: Human Perception and Performance*, 21(1):32, 1995.

[87] M. Vukobratovic and B. Borovac. Zero-moment point - thirty five years of its life. *International Journal of Humanoid Robotics*, 1(1):157–173, 2004.

[88] M. Vukobratovic, B. Borovac, D. Surla, and D. Stokic. *Biped locomotion: dynamics, stability, control and application*. Scientific Fundamentals of Robotics. Springer, 1990.

[89] M. Vukobratovic and D. Juricic. Contribution to the synthesis of biped gait. *IEEE Transactions on Biomedical Engineering*, BME-16(1):1–6, Jan 1969.

[90] A.-T. Wael A. Micro-robot management. *MATLAB - A Fundamental Tool for Scientific Computing and Engineering Applications*, 3, 2012.

[91] L. Wang and C. Chen. A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, 7(4):489–499, Aug 1991.

[92] L. Wang and C. Chen. On the numerical kinematic analysis of general parallel robotic manipulators. *IEEE Transactions on Robotics and Automation*, 9(3):272–285, 1993.

[93] L. Weiss, A. Sanderson, and C. P. Neuman. Dynamic sensor-based control of robots with visual feedback. *IEEE Journal of Robotics and Automation*, 3(5):404–417, 1987.

[94] X. Wu, L. Ma, Z. Chen, and Y. Gao. A 12-Dof analytic inverse kinematics solver for human motion control. *Journal of Information and Computational Science*, 1:137–141, 2004.

[95] T. L. Xuyang Wang and P. Zhang. Study on state transition method applied to motion planning for a humanoid robot. *International Journal of Advanced Robotic Systems*, 5:145–150, 2008.

[96] S. Yazdekhasti, F. Sheikholeslam, and M. Ghayour. Stability analysis of biped robot with direct control of zero moment point. In *The 2nd International Conference on Computer and Automation Engineering (ICCAE)*, volume 2, pages 528–532, Feb 2010.

[97] H. Yue, W. Chen, W. Chen, and X. Wu. Spline-interpolation based pvt algorithm and application in a bionic cockroach robot. In *11th International Conference on Control Automation Robotics Vision (ICARCV)*, pages 1742–1747, Dec 2010.

[98] N. A. J. Z. Y, Abdoon Al-Shibaany. Design and implementation of a real-time intelligent controller for a differential drive mobile robot. *International Journal of Scientific & Engineering Research*, 4(12):1556, 2013.

[99] L. Zhang, S. Bi, and D. Liu. Dynamic leg motion generation of humanoid robot based on human motion capture. In *Intelligent Robotics and Applications*, volume 5314 of *Lecture Notes in Computer Science*, pages 83–92. Springer Berlin Heidelberg, 2008.

[100] J. Zhao and N. Badler. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics*, 13:313–336, 1994.

[101] F. Zhou, B. Song, and G. Tian. Bézier curve based smooth path planning for mobile robot. *Journal of Information and Computational Science*, 8(12):2441–24450, 2011.

# Hebrew Abstract