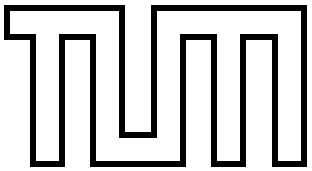


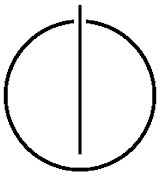
FAKULTÄT FÜR INFORMATIK

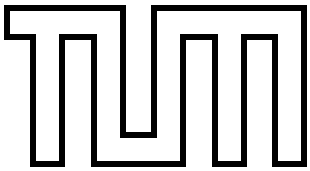
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's thesis

# Realizing Online (Self-)Collision Avoidance Based on Inequality Constraints with Hierarchical Inverse Kinematics

Karsten Knese





FAKULTÄT FÜR INFORMATIK

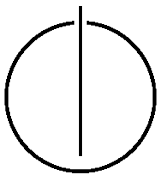
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's thesis

# Realizing Online (Self-)Collision Avoidance Based on Inequality Constraints with Hierarchical Inverse Kinematics

# Realisierung von Selbstkollisionsvermeidungen auf Basis von Ungleichheitsbedingungen mit Hierarchisch Inverser Kinematik


| | |
|---|---|
| Author: | Karsten Knese |
| Supervisor: | Prof. Dr. Daniel Cremers |
| Advisor: | Dr. Jürgen Sturm<br>[Technische Universität München] |
| | Adolfo Rodriguez Tsouroukdissian, PhD<br>[PAL Robotics S.L] |
| Date: | April 15, 2014 |



Ich versichere, dass ich diese Master's Thesis selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

I assure the single handed composition of this Master's thesis, only supported by declared resources.

München, den February 18, 2014 Karsten Knese

# Acknowledgments

# Abstract

In this Master's thesis, an approach for highly dynamic (self-) collision avoidance is presented. The collision avoidance is realized as an inequality constraint task inside the Stack-of-Tasks framework. It is fully implemented in a ROS compatible way, yet independent of any specialized hardware component.
The base of all developments introduced in this work is the integration of the Stack-of-Tasks framework inside a ROS based robot environment. Hereby, it utilizes a generic robot description file as well an hardware abstraction layer to guarantee a portability and straightforward migration to any robot satisfying those two requirements.
Furthermore, the (self-) collision avoidance is implemented for the quadratic programming solver inside the Stack-of-Tasks. Hence the name, the alignment of linear constraints in form of a stack enables an iterative solving progress with respect of given priorities to these constraints. The collision avoidance task is based on closest point pair calculation of two possibly colliding body parts. To achieve a unique solution for the closest point pair, a complete capsule decomposition for the robot model is created. The pseudo convexity of capsules shrinks the number of discontunuities and leads to a smooth transition. The velocity along the unit-vector of these two points gets limited by a threshold parametrized by the corresponding distance. The projection of the jacobian towards the collision center point gets restricted to zero in case of hitting the inequality boundary, which disables any movement towards the collision center. The calculation of the closest point pairs can be simultaneously used for self-collision avoidance as well as external collision objects captured via various sensors.
To keep the computation time of the collision matrix to a minimum, only the necessary body parts are distinguished needed to cover all possible self-collisions. As the implementation of the Stack-of-Tasks allows a dynamic "push" and "pop" of constraints from the stack, the computation time can be further reduced by only computing actively moving body parts.
All developments are developed and tested against PAL-Robotics REEM-H and REEM-C robots, however example implementations for various robot models such as Romeo or PR2 are provided. Throughout the development of this work, perspective applications like visual servoing and teleoperations got successfully instantiated and can be smoothly used given the self-collision avoidance.

# CONTENTS

**Acknowledgements** **vii**

**Abstract** **ix**

**I. Introduction and Theory** **1**

**1. Motivation** **3**
1.1. Problem Statement . . . . . 3
1.2. Overview of this Thesis . . . . . 4

**2. Related Work** **5**
2.1. Offline Motion Planner . . . . . 5
2.2. Reactive Planner / IKS . . . . . 5

**3. Stack of Task** **7**
3.1. Prerequisites . . . . . 7
3.1.1. Least Square . . . . . 7
3.1.2. Nullspace Optimization . . . . . 7
3.1.3. Quadratic Programming . . . . . 7
3.1.4. Position Controlling . . . . . 7
3.2. Stack Of Tasks . . . . . 7
3.2.1. Stack Operations, Push, Pop . . . . . 7
3.2.2. Task Definition . . . . . 7
3.2.3. Solving The Stack . . . . . 7
3.3. Stack of Tasks - Hierarchical Quadratic Programming . . . . . 7
3.3.1. Equality Constraints . . . . . 7
3.3.2. Inequality Constraints . . . . . 7

**4. Collision Avoidance** **9**
4.1. Task Definition . . . . . 9

4.1.1. Influence and Safety Zone . . . 9
4.1.2. Jacobian Projection . . . 9
4.2. Capsule Decomposition . . . 9
4.2.1. Strictly Convex Considerations . . . 9
4.2.2. Closest Point Calculation . . . 9

**II. Implementation** **11**

**5. ROS Controlled Robots** **13**
5.1. ROS-Control . . . 13
5.2. Dynamic Graph and Signals . . . 13
5.3. Fast Collision Library (FCL) . . . 13
5.4. Robot Hardware . . . 13

**6. Collision Avoidance Implementation** **15**
6.1. FCL Entity . . . 15
6.1.1. Dynamic Allocation of Collision Pairs . . . 15
6.1.2. Closest Points Signals . . . 15
6.2. Velocity Damping . . . 15
6.3. Joint Velocity Limits . . . 15
6.4. Joint Space Limits . . . 15
6.5. Task Weights . . . 15

**7. Experiments** **17**
7.1. TBD . . . 17

**8. Results** **19**

**Appendix** **23**

**A. Detailed Descriptions** **23**

**Bibliography** **25**

# Part I.

# Introduction and Theory

# CHAPTER 1
# MOTIVATION

## 1.1. Problem Statement

Robot technology is increasing statically as the performance of the underneath lying hardware components lead to accomplishments of highly complex software computations. Looking at state-of-the art robot dynamic systems e.g. Reem-C [4], Asimo [1], Hubo [3] and Atlas [2] indicates that Humanoid robots are capable of achieving versatile goals in appropriate time. At the time of this writing, the walking speed of Asimo is with 6 km/h the fastest known Humanoid. Videos like this expose the Kinematic capabilities of those systems with an remarkable speed.
[ picture of humanoid robots]
This follows the necessity of tackling the safety issues which rise on the highly developing speed of robots. Here, safety issues are meant in terms of Human Robot Interaction (HRI) where robots are more likely used in a human environment to fulfill or accompany tasks of human resource. Hereby, collision avoidance with the robots environment has to be on highest priority. Secondly, avoiding self-collision has to be maintained at all time. Special care has to be taken in versatile humanoid robots as they might easily collide with itself due to the redundant Degrees of Freedom (Dof) or the overlapping workspace of the attached manipulators.
However, the high speed of robots, problems and unmistakable performance issues arise at a point of considering a highly dynamic system. Modelling the environment of the robot happens usually in Task-Space rather than in Joint-Space. Peripherals like Vision systems percept the outer world of the robot in a metric unit independent of the robot kinematic chains. Obviously, this leads to a requirement of having a Inverse Kinematic Solver (IKS). Inverse Kinematic Solver can be time intensive when applied to a high number of DoFs. This reasons the numbers of computing trajectories for multiple minutes before applying them. Hence, it is not tremendously astonishing that robots equipped with two LWR III Lightweight Arms with an operating speed of 120 $\frac{Deg}{s}$ of joint speed still need approx. 4 minutes to finish a pancake [5] [footnote: This should in no way be meant as a negative aspect than more as an example of technical limitations.].
The limitation in execution speed is related to two critical aspects of motion planning.

Firstly, a valid trajectory without any collisions (self collision as well as with the robot environment) has to be found. Further, this involves robot dependent configurations such as joint limits and balancing the Center-of-Mass (CoM). Once a trajectory is found which satisfies all the constraints of not exceeding the joint limits, keep the robot balanced and does not comprise any collision, the IKS comes into place. Current solvers are mainly based on finding a pseudo inverse solution for resolving the joint state vector. This can further be used to involve a Null-Space optimization for additional constraints like an elbow field or using redundant DoF for secondary tasks.
[ short diagram of how the computation pipeline works on a high abstract level]
Full-Body Motion Control such as controlling a robot in various positions of all manipulators implies a high number of redundant joints. Furthermore, redundant manipulators imply a Null-space optimization with multiple independent constraints as well as nested constraints. Humanoid robots posses independent constraints in their two arms manipulators. The action of the right arm is mainly independent with respect to the action of the left. The only shared part of the kinematic chain would be the torso. As Full-Body Motion Control computes all DoF in one kinematic chain, this raises the need for an efficient solver. Thus, Full-Body Motion Control can only be effectively applied on a high dynamic system when the cyclic execution time of each update step is low in cost and real-time safe.

## 1.2. Overview of this Thesis

During this work, the two requirements of finding a trajectory and solving the Inverse Kinematic will be explored. The main focus hereby lies on the self-collision avoidance, which on the same hand includes the collision avoidance with dynamic obstacles as it will be shown to be the same algorithm. Part 1 of this work explains all the theoretical details which are necessary for understanding the concepts implemented in Part 2. By introducing a new way of solving Inverse Kinematic based on Hierarchical Programming and stacked tasks, time consuming components such as the collision checking and trajectory finding can be integrated directly into the solver. Thus, Chapter ?? is introducing the Stack-of-Tasks (SoT), which serves as the underlying software framework throughout this work for computing the inverse kinematics with respect to all constraints. Based on the capabilities of the SoT such as dealing with dynamic equality and inequality constraints, Chapter ?? exploits those techniques for formulating constraints such as joint limits, balancing Center-of-Mass and distance computation. Joint limits and distance computation are of major importance for fulfilling the collision avoidance task. Implementation details and practical experiments on the robot are depicted in Part 2. Experimental results on the robot expose further developments such as gain tuning which cannot be fully covered by theory as mechanical issues are just fairly modelled in the mathematical formulas. The results of this work were preceded by the integration and migration of the SoT as a pure autonomic c++ software framework into the Robot Operating System (ROS). This integration process was necessary to evaluate the results on ROS-enabled robots such as the ones from PAL-Robotics and make it comparable as well as competitive to other robots such as PR2.

# CHAPTER 2

# RELATED WORK

## 2.1. Offline Motion Planner

## 2.2. Reactive Planner / IKS

# CHAPTER 3
# STACK OF TASK

## 3.1. Prerequisites

### 3.1.1. Least Square

### 3.1.2. Nullspace Optimization

### 3.1.3. Quadratic Programming

### 3.1.4. Position Controlling

## 3.2. Stack Of Tasks

### 3.2.1. Stack Operations, Push, Pop

### 3.2.2. Task Definition

### 3.2.3. Solving The Stack

## 3.3. Stack of Tasks - Hierarchical Quadratic Programming

### 3.3.1. Equality Constraints

### 3.3.2. Inequality Constraints

# CHAPTER 4
# COLLISION AVOIDANCE

## 4.1. Task Definition

### 4.1.1. Influence and Safety Zone

### 4.1.2. Jacobian Projection

## 4.2. Capsule Decomposition

### 4.2.1. Strictly Convex Considerations

### 4.2.2. Closest Point Calculation

# Part II.

# Implementation Details

# CHAPTER 5
# ROS CONTROLLED ROBOTS

## 5.1. ROS-Control

## 5.2. Dynamic Graph and Signals

## 5.3. Fast Collision Library (FCL)

## 5.4. Robot Hardware

# CHAPTER 6
# COLLISION AVOIDANCE IMPLEMENTATION

## 6.1. FCL Entity

### 6.1.1. Dynamic Allocation of Collision Pairs

### 6.1.2. Closest Points Signals

## 6.2. Velocity Damping

## 6.3. Joint Velocity Limits

## 6.4. Joint Space Limits

## 6.5. Task Weights

# CHAPTER 7
# EXPERIMENTS

## 7.1. TBD

# CHAPTER 8

# RESULTS

# Appendix

# APPENDIX A

# DETAILED DESCRIPTIONS

Here come the details that are not supposed to be in the regular text.

# BIBLIOGRAPHY


[1] Honda Asimo. Asimo specification. http://asimo.honda.com/asimo-specs/, 2014.

[2] Darpa. Darpa' atlas robot unveiled. http://www.darpa.mil/NewsEvents/Releases/2013/07/11.aspx, 2013.

[3] Korean Advanced Institute for Science and Technology. Hubo specification. http://ohzlab.kaist.ac.kr/p_hubo2p, 2014.

[4] Pal Robotics S.L. Reem-c specification. http://ohzlab.kaist.ac.kr/p_hubo2p, 2014.

[5] Technische Unversität München WillowGarage. Tum rosie and pr2 james make pancakes together. http://www.willowgarage.com/blog/2010/10/21/tum-rosie-and-pr2-james-make-pancakes-together?page=5, 2010.