Visual SLAM for humanoid robot localization and closed-loop control

Arnaud Tanguy

Collège Doctoral du Languedoc-Roussillon

THÈSE

Pour obtenir le grade de Docteur

Délivré par l'**Université de Montpellier**

Préparée au sein de l'école doctorale :
Information Structures Systèmes (I2S)

Et de l'unité de recherche :
Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier

Spécialité :
Systèmes Automatiques et Microélectroniques

Présentée par **Arnaud TANGUY**

Visual SLAM for humanoid robot localization and closed-loop control

Soutenue le 28 Novembre devant le jury composé de

M. Olivier Stasse	Directeur de Recherche	LAAS-CNRS	Rapporteur
M. David Filliat	Professeur	ENSTA ParisTech	Rapporteur
M. Phillippe Martinet	Directeur de recherche	INRIA	Examinateur
Mme. Claire Dune	Maître de conférences	Université de Toulon	Examinateur
M. Abderrahmane Kheddar	Directeur de Recherche	CNRS-UM LIRMM	Directeur de thèse
M. Andrew Comport	Chargé de Recherche	CNRS-UM I3S	Co-encadrant de thèse

Université Montpellier

Acknowledgements

I would like to thank all the people who have supported me, directly or indirectly, throughout this endeavour.

First, comes my family, parents and sisters that have always been here for me. My parents who have always pushed me to reach my best in all my endeavours, be it academic or personal. My sisters, that I have somewhat neglected of late. Life is all about choices, and I chose to pour every bit of energy that was not used in work leading up to the elaboration of this manuscript, to my passion for climbing and mountaineering. Let this writing be both an apology, and a reminder never to get so caught up in my own life as to neglect such as important thing as family, seeing the nephews grow, visiting the Albatross, and much more.

Then my colleagues from I3S, LIRMM and JRL, without whom this experience would not have been the same. Special thanks to Fernando, Myriana and Majdi, for being true friends in-and-out of the lab. Special thanks to Stéphane Caron for his constant support and helpful comments, Kevin for bearing with me[1], to Pierre Gergondet without whom the DRC would not have been possible and for the mc_rtc framework.

My climbing partners, Florian, Angelik, Marie-Doha, Antoine, Julien, Jerôme, Amandine to name but a few, deserve a special place in this acknowledgement. Without them, I would definitely have gone crazy. Or maybe I did? Let's find out this winter!

I would also like to thank my professors for the four year we've spent together. Andrew Comport for sparking my interest in SLAM and inspiring me to undertake this thesis, and for always being there to support me afterwards. My thanks to you, Abder, for your patience and capacity to let bygones be bygones, and always focus on what needs to be done. We haven't always understood each other, and that's a shame, a lot would have gone smoother if we had.

Last, but not least, I would like to thank the members of the jury for accepting to share their valuable time to evaluate this work.

[1] 1...2 robots? We need one more!

TITRE : SLAM visuel pour la localisation et la commande en boucle fermée de robots humanoïdes

RÉSUMÉ :

Cette thèse traite du problème de localisation et contrôle de robots humanoïdes par rapport à leur environnement, tel qu'observé par ses capteurs embarqués. Le SLAM visuel dense, consistant en l'estimation simultanée d'une carte 3D de l'environnement et de la position du robot dans cette carte est exploité pour étendre et robustifier les méthodes de planification contrôle multi-contact. Celles-ci consistent à établir et exploiter des contacts robot-environnement pour accomplir des tâches de locomotion et manipulation. Des incertitudes sur la posture initiale du robot, ainsi que des perturbations causées par une modélisation inadéquate des contacts, ainsi que des perturbations externes oblige à la prise en compte de l'état du robot et son environnement. Une méthode de calibration corps-complet est également proposée, afin d'obtenir une connaissance fiable de la chaîne cinématique du robot, nécessaire pour réaliser de telles tâches. Finalement, une méthode de marche basée sur de la commande prédictive de modèles est robustifiée par la prise en compte de large perturbations, permettant d'ajuster les trajectoires de pied et du centre de masse afin de garantir sa stabilité, tout en accomplissant les objectifs désirés. Les méthodes proposées sont illustrées et validées par de multiples expérimentations sur les robots humanoïdes HRP-2Kai et HRP-4.

TITLE: Visual SLAM for humanoid robot localization and closed-loop control

ABSTRACT: This thesis deals with the problem of localizing and controlling humanoid robots with respect to its environment, as observed by its on-board sensors. Dense visual *SLAM*, consisting in the simultaneous estimation of a 3D map of the environment and of the robot localization within that maps is exploited to extend and robustify multi-contact planning and control. Establishing and exploiting robot-environment contacts allows the accomplishment of both locomotion and manipulation tasks. Uncertainties in the initial robot posture, and perturbations arising from improper contact-modelling and external causes are accounted for by observing the state of the robot and its environment. A whole-body calibration method is also proposed, so that robust knowledge of the robot's kinematic structure is known, a prerequisite to all robot-environment interaction tasks. Finally, a walking method based on model predictive control is robustified by taking into account large perturbations, and adjusting the footstep and center-of-mass trajectories accordingly to guarantee stability while accomplishing desired objectives. Several experiments on an HRP-2Kai and an HRP-4 humanoid robots are presented and discussed to illustrate and validate each of the proposed methods.

Table of contents

Introduction

Human beings are extremely proficient in interacting with their environment, and can easily adapt to any situation. Locomotion and manipulation are both achieved by applying forces at contact points between the body and the environment, and we are extremely good at it. One of the best examples for this, from my personal experience, is to look at an Alpinist evolving in a mountain environment, where you will find all the most complex of scenarios.

The image above shows three representative examples: the first, is climbing down a set of dozens of ladders leading to the Glacier d'Argentière in the French Alps; the second is negotiating a narrow rock ridge-line; the third is climbing a steep overhanging wall with ice-climbing tools. To achieve any of these scenarios, one needs to establish contacts between parts of the body and suitable surfaces within the environment, figure out the best body position for stability, how much force to apply, and so on. As humans, we are able to do this seemingly intuitively, endowing us with the capability to accomplish almost any conceivable action. This impressive ability to interact with any environment has been a considerable source of inspiration in the development of humanoid robots. Not contempt with being designed to look and move like us, with significantly similar degrees of freedom and a kinematic structure mimicking ours (legs and arms attached to a waist and torso), they also attempt to replicate our main senses (vision, force sensing, proprioception, *etc*). The intuition is that the kinematic and sensory attributes of human beings are extremely well adapted to the accomplishment of any task within the environment, and that a robot with similar capabilities

could one day achieve similar feats. A legitimate question is raised : why build a robot to perform tasks that humans are already capable of?

Several applications are envisioned for humanoid robots: rescue and intervention in disaster situations (detailed hereafter); home service companions to assist frail and ageing people[2] ; collaborative workers (i.e. as cobots termed “comanoids”) in large manufacturing assembly plants[3] where wheeled and rail-ported robots cannot be used (e.g. aircrafts and shipyards); amongst other applications.

In rescue and intervention in disaster scenarios, robots are envisioned to replace humans in hazardous conditions. A strong case for humanoids was made in 2011, when a magnitude 9 earthquake and subsequent tsunami laid waste to Japan. Besides the expected devastation, a catastrophic explosion occurred at Fukushima Daiichi’s nuclear power plant, followed by massive radioactive leaks preventing workers from accessing the area to mitigate consequences of the disaster. Humans, being unable to access the area, the work had to be attempted by robots instead, in a complex environment initially designed for humans. Staircases, ladders, doors, valves, rubble and damaged floors, and so many more challenges to be expected! Not surprisingly, traditional wheeled robots performed very poorly, and none came close to successfully accomplishing any tasks.

At the time, humanoid robots were not ready yet to face such real challenges, and in an effort to encourage further development of humanoid robotics in facing such scenarios, the DARPA Robotics Challenge[4] (DRC) was created. It invited the robotics community to face eight challenging tasks back-to-back, without any safety tether for the robots, and thus substantial material risks in case of failure. Those tasks were, in order:

1. Driving a car through a simple track,
2. Getting out of the car (egress),
3. Walking to a door and opening it,
4. Walking to a valve and rotating it by 360 degrees,
5. Walking to a wall, picking up a drill and cutting out a large square window to reach objects behind it,
6. Walking to a "surprise" tasks. In practice it either consisted in opening a box and pushing a button, or unplugging and re-plugging an industrial cable,

[2] https://projetromeo.com
[3] www.comanoid.eu
[4] https://www.darpa.mil

7. Walking through, or clearing a rubble field,

8. Climbing up a set of stairs (with a handrail).

Fig. 1 DRC Tasks performed by IHMC Team [Johnson et al., 2015]

The challenge highlighted the strengths and weaknesses of humanoid robots. It demonstrated that robots with human-like morphology are well-suited for the tasks, as some of them managed to complete all the tasks. However, it also highlighted that humanoids are not yet ready to autonomously face the challenge presented with performing tasks in uncontrolled environments. The DARPA Robotics Challenge test field (Figure 1) was nowhere near being as challenging as a real environment would be. The ground was clear of debris, the obstacles far apart, overall a very controlled environment; and the teams had months of experience in trying the various tasks at hand. Still, only 3 out of 23 teams managed to complete all the tasks, and only a single team managed to perform the whole challenge without their robot falling over. Over the course of the challenge, most robots had some dramatic failure due to (i) errors in localization and perception of the environment, and (ii) lack of suitable stabilization, and (iii) human errors, or simply (iv) programming errors (bugs). A comprehensive review of what went wrong during the challenge was compiled by Atkeson [2015].

In my personal experience as part of team *AIST-NEDO* [Cisneros et al., 2015] (ranked 10^{th} out of 23 teams, with $5/8$ tasks completed), the main limiting factor across all teams was

the lack of experience in taking into account the state of the robot *w.r.t.* to its environment into the control strategies. At the perception level, such estimation was far from mastered, and most teams were relying on ad-hoc perception methods. In the case of team AIST-NEDO, the HRP-2Kai humanoid robot used during the challenge had been fitted with a 2D laser range finder (LRF), used to obtain thee dimensional measurements of the environment in front of the robot by performing a head-scan motion from a static posture and localizing itself once *w.r.t.* to that environment. Sitting amongst the non-expert audience watching the challenge live, one could hear legitimate questioning

– Why is it so slow? What is it doing?
– Oh look it's moving now... wait, why did it stop? What is it thinking?
– Oh look at that one with the cute ears?

People truly related, they got exited when the robot succeeded, felt empathy when it failed. Yet, the recurring conversation was wonder at what could be taking them so long to perform such simple looking tasks? And with reason. Imagine a human having to stop, stay still, move its head up and down to figure out where he is and what he's looking at, and then closing his eyes and performing the task blindly. Not really a natural and robust way of achieving complex tasks, is it not? This is exactly what was done during the DRC, by our team, and many others, and a very, very long time was spent doing it.

This lack of perception could also be found at control level. Most robots in the challenge fell at least once, and none had an appropriate reaction - if any - to these falls. In our team, after successfully walking over the debris field our robot simply slowly toppled over on the last step. It did not attempt to regain balance, which could easily have been achieved by using the arms to apply some momentum to the body, or simply taking a small step to recover from the perturbation. It was later determined that the failure occurred due to a mis-estimation of the step height by about 4cm. Many more examples of such failures could be cited, and mocking videos of robots failing have even been compiled[5].

In recent years, dense visual *Simultaneous Localization and Mapping* (*SLAM*) methods have gained considerable maturity. Based only on data obtained with vision sensors (such as color and depth cameras), they simultaneously track the sensor motion, and reconstruct a 3D-representation (map) of the observed environment at sensor rate (typically around 30Hz). Modern methods such as those developed by Audras et al. [2011], Meilland et Comport [2013a] and Whelan et al. [2012, 2015b] have proved themselves both in terms of robustness and precision, but haven't yet been fully exploited in the context of humanoid robot control. Yet offering both 6D-localization (position and rotation), and three dimensional maps, it

[5] Video compilation of DRC failures: https://youtu.be/g0TaYhjpOfo

has the potential to answer the fundamental question that is preventing robots from robustly interacting with the world, and will constitute the main focus of this thesis.

How can a humanoid robot be controlled with respect to its observed environment?

Objective The DARPA Robotics Challenge exposed a clear lack of expertise in (i) perceiving the environment in which the robot evolves, (ii) localizing the robot *w.r.t.* to that environment, and (iii) exploiting this information to inform control strategies. This thesis aims at exploring all three aspects by exploiting the capabilities of dense Visual SLAM algorithms to devise a complete whole-body control framework that accounts for the state of the robot in its environment. An overview of this framework is provided below (Figure 2), and its main components will be detailed throughout this thesis.

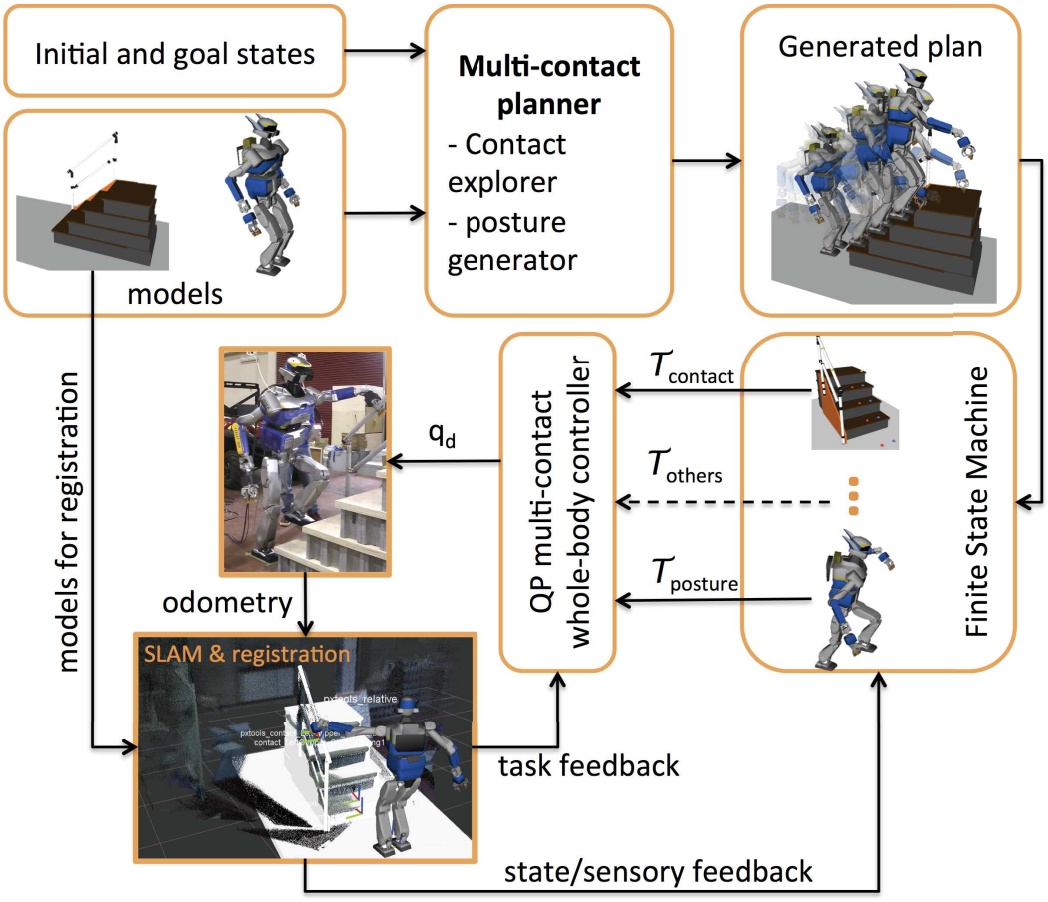


Fig. 2 Overview of the closed-loop control framework: from a desired goal and 3D models, a multi-contact planner generates a plan with feasible contacts and associated robot posture. A Finite State Machine (FSM) is generated to handle the set of constraints and tasks of an online Quadratic Programming (QP) controller that computes actuator commands $\mathbf{q}_d$ that drive a humanoid robot to accomplish desired tasks. The map obtained from *SLAM* is used to relate the 3D models with the real environment through registration, while its pose estimates are used to estimate the full state of the robot, and close the control loop.

Contributions and plan The main focus of this thesis is the formulation of a closed-loop approach to humanoid robot control, where Simultaneous Localization and Mapping is used to perceive the environment and estimate the state of the robot *w.r.t.* that environment.

- Chapter 1 provides an overview of the state-of-the-art in both humanoid robot control, and dense visual *SLAM* algorithms. The importance of taking into account the state of the robot within its environment, and how it can be incorporated in the traditional control methods for humanoid robots is further explored.

- Chapter 2 highlights the importance of kinematic-chain calibration to precisely know the location of any surface on the robot, including its RGB-D sensor. For robust operation, especially in challenging environments, such calibration procedure should not require any human intervention, nor any calibration apparatus (printed checker-board patterns, *etc*). This is particularly important if we wish to one day have humanoid robots that can intervene anywhere, and are likely to get damaged in the process. Such damages are not hypothetical, as the work of Samy et Kheddar [2015], where an HRP-4 humanoid robot was pushed to investigate strategies for reducing the impact of the fall. This experiment resulted in damages in the elbow mechanical structure, far too weak to sustain fall impacts. Both of our robots HRP-4 and HRP-2Kai also sustained damages during experiments: one caused by the rope-safety mechanism, where the rope broke the mechanical binding for the camera (Figure 2.1), the other where HRP-2Kai neck was bent as a result of a collision. In the last two examples, the damage is particularly impacting, as the transformation between the RGB-D sensor itself and the robot becomes inaccurate, which affects the localization of all of the robot links within the environment.

  A novel whole-body calibration method was envisioned, to autonomously calibrate the robot's kinematic chain online, without requiring any calibration apparatus. The proposed method relies solely on sensors widely available on any humanoid robot. Robot motion, acquired from optical joint encoders and dense visual SLAM tracking of the RGB-D sensor images is used as input to our calibration procedure.

- Chapter 3 considers how complex multi-contact plans generated offline on CAD models can be successfully executed *w.r.t.* an environment reconstructed with *SLAM*. It is shown that the initial robot state *w.r.t.* to the environment does not need to be the one considered during the offline planning phase, and that contacts can robustly be achieved with a simple task-space closed-loop control of the robot end-effectors, even under perturbations. Experiments with HRP-2Kai on complex multi-contact plans are

shown, including walking towards and interacting with an industrial valve, grasping the wheel of the DRC vehicle, and grasping a staircase handrail.

- Chapter 4 explores dynamic walking. A closed-loop model predictive controller (MPC) is formulated, and *SLAM* is exploited to regulate a Cartesian-space controller and make the robot walk to desired location. The MPC is computed from an estimated state of the robot obtained with both *SLAM* and force-torque sensor measurements, and its ability to react to strong perturbations by altering its footstep and associated center-of-mass trajectory.
- The conclusion summarizes the main contributions introduced in this manuscript, and provides perspectives for future improvements.

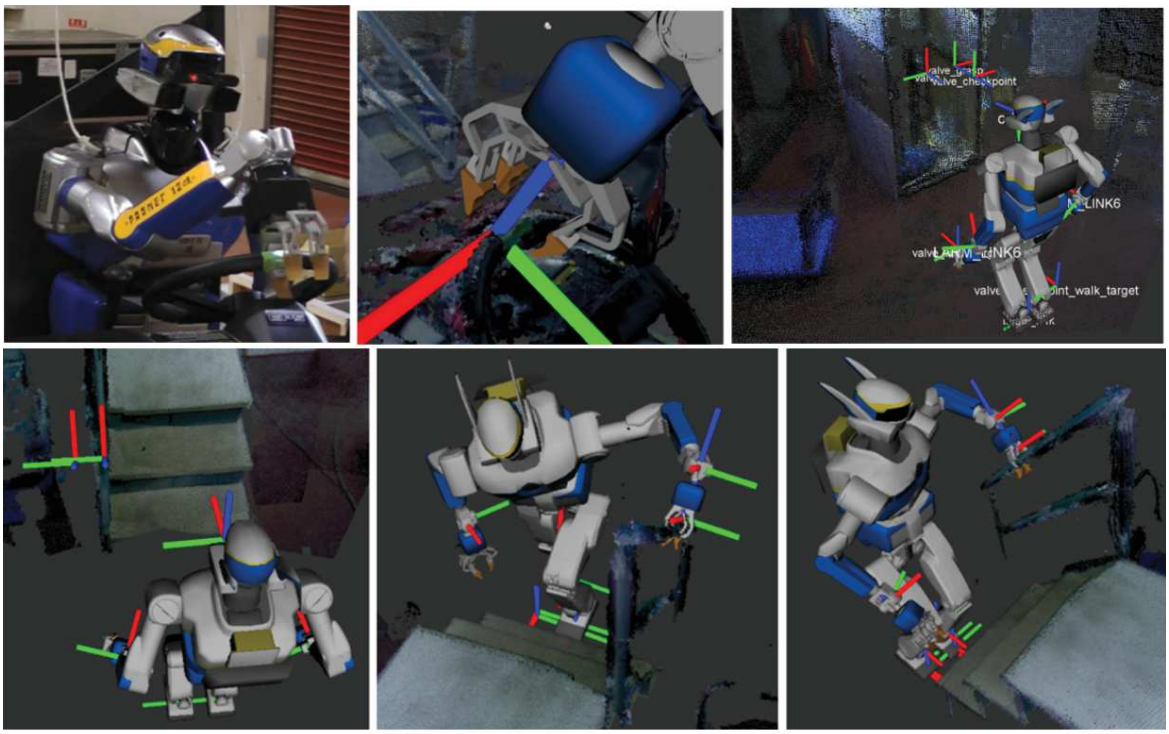


Fig. 3 Example of multi-contact control experiments with HRP-2Kai and dense visual SLAM.

The work presented in this thesis have been presented in the following peer-reviewed publications:

- **Online Eye-Robot Calibration** [SIMPAR 2018] [Tanguy et al., 2018b] covering the novel Robot-Eye calibration presented in Chapter 2.

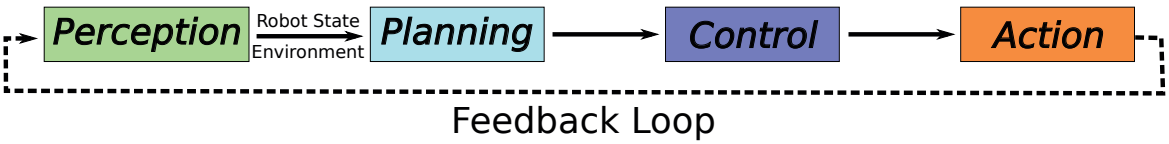
- **Closed-Loop RGB-D SLAM Multi-Contact Control for Humanoid Robots** [SII 2016] [Tanguy et al., 2016] covering the use of multi-contact plans and closed-loop control *w.r.t.* the environment using dense visual SLAM, and is reported in Chapter 3.
- **Closed-loop MPC with Dense Visual SLAM - Stability through Reactive Stepping** [ICRA 2018, *submitted*] [Tanguy et al., 2018a] covering the work on closed-loop model predictive control presented in Chapter 4. The work was carried out in close collaboration with Daniele De Simone (*Sapienza University of Rome, DIAG, Italy*).

In addition to these contributions, and in order to increase the usefulness and reach of this work, the methods developed in this thesis are made available as open-source software whenever possible. See Appendix C for details about published software.

Chapter 1

State of the art and motivations

Humanoid robots are intended to imitate human capabilities. Doing so requires *perception* to observe the environment, *planning* to decide how to best achieve a task in that environment, and *robot action* to realize the planned task. Consequently, a high-level representation of any robot controller can be depicted as



Perception Humanoid robots can in general move in all directions of space, and interact with three dimensional objects. As such, a truly autonomous robot should have the capability to observe a 3D-map of its surroundings, and know its position and orientation within that map at all time. To do so, it must rely solely on its on-board sensors. Most humanoid robots are fitted with at least (i) a vision sensor (camera, RBB-D sensor, stereo pair, LIDAR, etc), (ii) force-torque sensors, (iii) joint encoders, and (iv) inertial measurements. Consequently this thesis focuses on methods able to provide this information from the available sensors. In particular, visual *Simultaneous Localization and Mapping* (*VSLAM*) methods have reached a high degree of maturity, making them highly suited for that purpose.

Planning Humanoid robots are complex systems, with many redundant degrees of freedom, and finding how to make them interact with their environment to accomplish desired tasks is not a trivial problem. Common planning methods will be reviewed here, including *multi-contact planning* (*e.g.* finding contact surfaces and associated postures that move the robot from one configuration to another), and *model predictive control*

(MPC) (planning trajectory over a future timespan based on some simplified model of the system).

Control Planning is seen here as a higher level framework that generates reference actions to be realized by a robot, such as attempting to respect a given whole-body posture, while keeping it's center of mass (CoM) at a desired position and moving one of its end-effectors to a desired location. These are high-level commands, that need to be tracked by a local controller that generates suitable actuator commands to track these reference actions, while respecting additional constraints, such as kinematic and dynamic feasibility, limits on generated torques, *etc*. In this thesis, we consider in particular an acceleration-based *Quadratic Programming (QP)* controller.

Action Desired references generated by the controller are then followed by the system that is being controlled. This might be a humanoid robot in its real physical environment, or any kind of simulation of such a system. The important point to note here, is that this system will not necessarily perfectly achieve the desired state asked by the controller. Discrepancies can arise due to un-modelled effects, improper contact modelling, external perturbations, *etc*.

Feedback Loop The role of a feedback loop is to observe the action of the robot system, and use this knowledge to alter the control law in such a way that it achieves its desired task. In practical robot systems, feedback loops are everywhere, from high-level reasoning to low-level actuator control.

These main components will now be reviewed individually in greater detail, before considering where and how Simultaneous Localization and Mapping can be used to improve upon the existing control schemes.

1.1 Main aspects of Humanoid Robot control

Humanoid robots are very complex systems, and controlling them is not an easy task. In this section, we will review the main aspects of humanoid robots, and what makes them so special and complex to control.

1.1.1 Humanoid robot presentation

Humanoid robots are human-size and human-shaped robots. That is, they have similar limb structure, *e.g.* two legs and two arms attached to a waist and torso respectively. These limbs

are connected by joints (actuated or passive), providing them with similar degrees of freedom as we humans have. The word *similar* has an importance here, as due to limits in mechanical joints and actuator design, their kinematic capabilities tend to be much less flexible than that provided by human joints. The humanoid robots HRP-4 [Kaneko et al., 2011] and HRP-2Kai [Kaneko et al., 2015] (standing for *Humanoid Robot Project*), both constructed by the Japanese company Kawada Industries[1] are used for most experiments performed in this thesis. Both robots have a similar kinematic structure, as depicted in Figure 1.1 and 1.2. HRP-4 has 32 actuated degrees of freedom, plus passive cable actuation in the fingers that are closed together with a single actuator. It is electrically actuated, and uses Harmonic Drive technology for transmission, which provides back-drivable high-reduction transmission with no backlash. HRP-2Kai has similar characteristics, and details can be found in Kaneko et al. [2015].

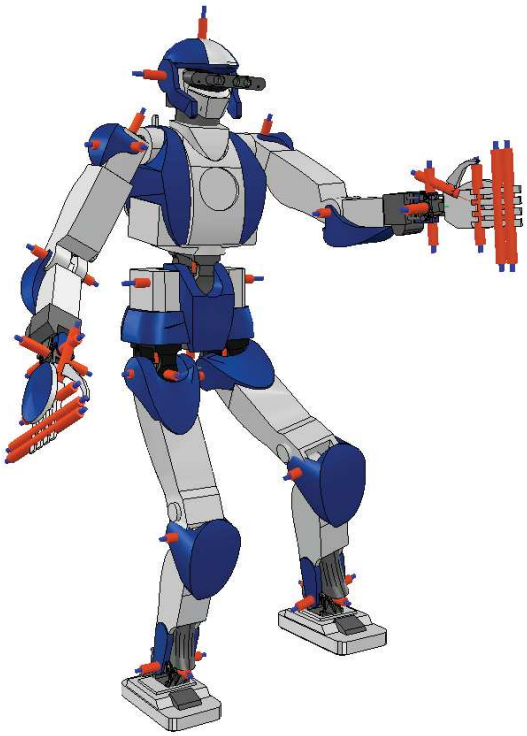


Fig. 1.1 HRP-4 kinematic structure. Orange cylinders represent joints around which robot links move. On HRP-4, all main joints are revolute joints, fingers are a passive cable mechanism.

Both HRP-4 and HRP-2Kai robots are fitted with a passive shock-absorbing rubber bush between its sole and a force-torque sensor attached under its ankle joint [Kanehira et al.,

[1] http://global.kawada.jp/

2002]. Its role is two-fold, (i) it protects the force sensor from high impacts that are likely to occur as the robot is stepping, and (ii) it provides passive compliance while walking, increasing the robustness of its interaction with the environment. However, as will be later seen in this thesis, this mechanism also increases the complexity of control, and in particular the estimation of the robot state. While this particular mechanism is specific to HRP robots, similar method have been employed by other manufacturers, such as

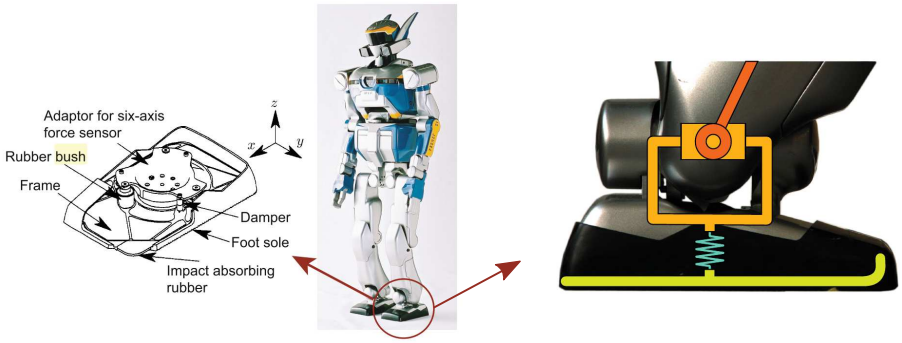


Fig. 1.2 Passive foot mechanism on HRP-2. Figures from [Kajita et al., 2014, Sec 3.2.2] and Benallegue et Lamiraux [2015]

1.1.2 Equations of Motion

No work on humanoid robotics would be complete without citing the standard equations governing the motion of a humanoid robot. The presentation here is largely inspired by the excellent tutorial by Caron [2018]. Another excellent reference is that of Wieber [2005]. The equations themselves can be found in any robotics handbooks such as Kajita et al. [2014] and Featherstone [2014] to cite but a few. They are derived from Newtonian and Lagrangian mechanisms and describe the motion of a physical system as a function of time and controls. In their most general form, they are written

$$F(\mathbf{q}(t), \dot{\mathbf{q}}(t), \ddot{\mathbf{q}}(t), \boldsymbol{u}(t), t) = 0, \tag{1.1}$$

where

- t is the time variable,
- $\mathbf{q}$ is the vector of generalized coordinates, for instance the vector of joint-angles,
- $\dot{\mathbf{q}}$ is the first time-derivative (velocity) of $\mathbf{q}$,
- $\ddot{\mathbf{q}}$ is the second time-derivative (acceleration) of $\mathbf{q}$,
- $\boldsymbol{u}$ is the vector of control inputs.

These equations provide a mapping between the *control space* (the commands that are sent to actuators, often referred to as *joint space*) and the *state space* of the robot (what we want to control in practice: the robot's position and velocity).

Case of manipulators

Manipulators are a particular type of articulated system where at least one link is fixed to the environment. The fixation is assumed to be strong enough to withstand any effort exerted on it. Under these assumptions, the equations of motion for a manipulator can be written

$$\boldsymbol{M}(\mathbf{q})\,\ddot{\mathbf{q}} + \dot{\mathbf{q}}^{\top}\boldsymbol{C}(\mathbf{q})\,\dot{\mathbf{q}} + \boldsymbol{g}(\mathbf{q}) = \boldsymbol{\tau}, \tag{1.2}$$

where, denoting by $n = \dim(\mathbf{q})$ the degree of freedom of the robot,

- $\boldsymbol{M}(\mathbf{q})$ the $n \times n$ inertia matrix,
- $\boldsymbol{C}(\mathbf{q})$ the $n \times n \times n$ Coriolis tensor,
- $\boldsymbol{g}(\mathbf{q})$ the n-dimensional gravity vector,
- $\boldsymbol{\tau}$ the n-dimensional vector of actuated torques.

This equation is typically used to represent how the system, starting from a state $(\mathbf{q}, \dot{\mathbf{q}})$, reacts under an applied torque $\boldsymbol{\tau}$ (control).

Case of humanoid robots

We now enter the heart of what makes humanoid robot control so special. Contrary to manipulators, humanoid robots are under-actuated systems, meaning that some of their degrees of freedom cannot be controlled directly in the *control space*. They can make and break contact with the environment, and through these interactions influence how their body is moved through space. A reference frame, often referred to as *floating base* and occasionally *base link*, is chosen on the humanoid body (typically its waist), which represents its configuration in space. This adds 6 unactuated degrees of freedom to fully describe the position and orientation of the robot in space. A robot with n degrees of freedom is fully described by $\mathbf{q} \in \mathbb{R}^{n+6}$, and the vector of actuated torques is still $\tau \in \mathbb{R}^n$. The position and orientation of any other link can be obtained by forward kinematics of its *joint space* configuration.

These 6 unactuated degrees of freedom are central to what makes humanoid control so complex. Its state is entirely dependent on physical interaction with the environment,

occurring through forces applied at contact points, and subject to gravity. This has several implications.

The first implies that, its state cannot be reliably obtained by only considering the control inputs. Assuming the robot-environment link to be established perfectly as expected, a forward-kinematics algorithm Featherstone [2014] along with the position and orientation of contacts can be used to obtain its configuration. Unfortunately, in general, inaccuracies arising from improper modelling of the physical interactions at contact points, along with external perturbations will cause it to drift away from it's desired control. If left unaccounted for, this drift quickly accumulates, and may lead to disastrous effects, such as the robot falling. To prevent this, it's state needs to be estimated, and carefully taken into account, which will be later reviewed (see Section 4.2).

The second has a strong implication on how humanoid robots are controlled. Since its *floating base* degrees of freedom can only be controlled through contact forces applied onto the environment, they need to be carefully modelled. A good overview of contact mechanics and their modelling is provided in Abe et al. [2007], where the problem of generating dynamically consistent motions for computer graphics avatars is considered.

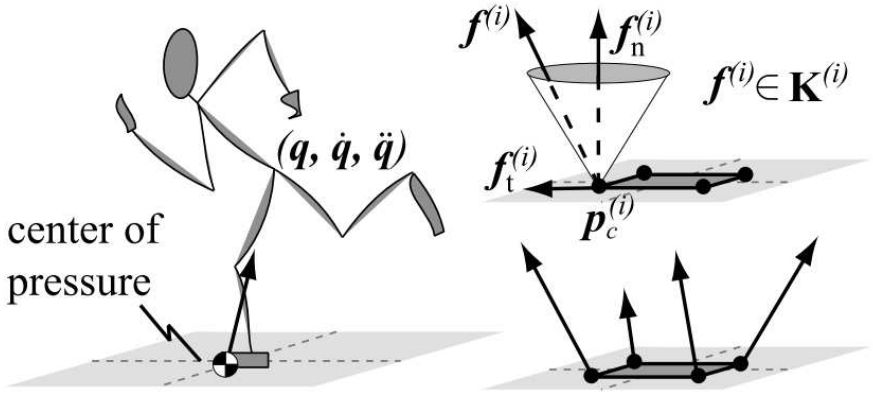


Fig. 1.3 Illustration of environment-contact interactions and constraints with friction-cones (Figure [Abe et al., 2007])

Point contacts and contact forces Contacts with the environment can be seen as constraints between each point $\mathbf{p}_{C_i}$ on the robot in contact with a corresponding point $\mathbf{p}_{D_i}$ in the environment. For non-slipping contacts, this constraint expresses kinematically : each robot-environment contact point is fixed, that is $\mathbf{p}_{C_i}(\mathbf{q}) = \mathbf{p}_{D_i}$. This also implies constraints on each

point's velocity and acceleration

$$\begin{aligned} \dot{\mathbf{p}}_{C_i} = 0 \quad &\Leftrightarrow \quad \boldsymbol{J}_{C_i}\dot{\mathbf{q}} = \mathbf{0} \\ \ddot{\mathbf{p}}_{C_i} = 0 \quad &\Leftrightarrow \quad \boldsymbol{J}_{C_i}\ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \boldsymbol{H}_C \dot{\mathbf{q}} = \mathbf{0} \end{aligned},$$

where $\boldsymbol{J}_{C_i}$ and $\boldsymbol{H}_C$ are respectively the Jacobian and Hessian of the position vector $\boldsymbol{p}_{C_i}(\mathbf{q})$.

A point contact yields a contact interaction force $\boldsymbol{f}_i$ that prevents geometric overlap between the robot and environment surfaces (assuming rigid surfaces). This interaction force is not arbitrary. To respect the non-slipping contact assumption, the contact force is restricted by the Coulomb friction model, that restricts the tangential component of the force, such that $||\boldsymbol{f}_t^i|| \leq \mu \boldsymbol{f}_n^i$, where $\mu > 0$ is a positive coefficient of friction that depends on the material properties of the surfaces in contact. Furthermore, a contact force does not pull on the body in case of separation (unilateral contact), which implies that its normal component must be positive $\boldsymbol{f}_n^i \geq 0$. These limits are commonly represented as a friction cone K^i that restricts the direction and magnitude of the contact force

$$\boldsymbol{f}^i \in K^i = \left\{ \boldsymbol{x} \quad \middle| \quad ||\boldsymbol{x}_t|| \leq \mu \boldsymbol{x}_n \right\} \tag{1.3}$$

By the principle of virtual work [Spong et Vidyasagar, 2004, p. 272], a linear map $\boldsymbol{J}_{C_i}^\top \boldsymbol{f}_i$ determines the contribution of each contact point to the joint torque. Thus, assuming the humanoid is making N point contacts $C_1, \ldots, C_N$ with the environment, their equations of motion become

$$\boldsymbol{M}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \boldsymbol{C}(\mathbf{q})\dot{\mathbf{q}} + \boldsymbol{g}(\mathbf{q}) = \boldsymbol{S}^\top \boldsymbol{\tau} + \sum_{i=1}^{N} \boldsymbol{J}_{C_i}^\top \boldsymbol{f}_i, \tag{1.4}$$

where

- $\boldsymbol{S}$ is the $n(n+6)$ matrix that selects the n actuated coordinates in $\mathbf{q}$. If the six floating base coordinates are the first entries of $\mathbf{q}$ then $\boldsymbol{S} = [6 \times n, \boldsymbol{I}_n]$. This convention will be followed throughout this thesis.
- $\boldsymbol{J}_{C_i}$ is Jacobian of the position vector (in the inertial frame) $\boldsymbol{p}_{C_i}(\mathbf{q})$ of a contact point C_i.
- $\boldsymbol{f}_i$ is the contact force exerted by the environment on the robot at the contact point C_i.

Contact points C_i are taken at the interface between robot links and the environment. In the case of surface contacts, when a robot's surface is in full contact with an environment surface (such as a foot on the floor), it is sufficient to take C_i's at the vertices of the contact polygon [Caron et al., 2015].

Surface contacts and contact wrenches A similar representation can be obtained for surface contacts (such as a flat foot on the floor). Kinematically, this is expressed by the equality

$$^{D}\boldsymbol{T}_{C}(\mathbf{q}) = \boldsymbol{I}_{4},$$

where $^{D}\boldsymbol{T}_{C}$ is the transform matrix between a robot's frame with origin at C and an inertial frame with origin at D [4]. It binds velocities and accelerations as well by

$$\begin{array}{rcl} \frac{\partial}{\partial t}{}^{D}\boldsymbol{T}_{C} = \mathbf{0}_{4\times 4} & \Leftrightarrow & \boldsymbol{J}_{C}\dot{\mathbf{q}} = \mathbf{0} \\ \frac{\partial}{\partial t^{2}}{}^{D}\boldsymbol{T}_{C} = \mathbf{0}_{4\times 4} & \Leftrightarrow & \ddot{\mathbf{q}} + \dot{\mathbf{q}}^{\top}\boldsymbol{H}_{C}\dot{\mathbf{q}} = \mathbf{0} \end{array} \tag{1.5}$$

where $\boldsymbol{J}_{C}$ is the manipulator Jacobian of the contact frame rooted at C, obtained by stacking

- the rotational Jacobian of the contacting link, *i.e.*, the matrix J^{rot} such that the rotational velocity ω of the link satisfies $\omega = \boldsymbol{J}^{\text{rot}}\dot{\mathbf{q}}$
- the linear Jacobian $\boldsymbol{J}_{C}$, *i.e.*, the matrix such that the linear velocity $\dot{\mathbf{p}}_{C}$ satisfies $\dot{\mathbf{p}}_{C} = \boldsymbol{J}_{C}\dot{\mathbf{q}}$

Under frame contact constraints, the equations of motion become

$$\boldsymbol{M}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{q}}^{\top}\boldsymbol{C}(\mathbf{q})\dot{\mathbf{q}} + \boldsymbol{g}(\mathbf{q}) = \boldsymbol{S}^{\top}\boldsymbol{\tau} + \sum_{i=1}^{N}\boldsymbol{J}_{C_i}^{\top}\boldsymbol{\omega}_{C_i}, \tag{1.6}$$

where $\boldsymbol{\omega}_{C_i} = \begin{bmatrix} \boldsymbol{\tau}_{C_i} \\ \boldsymbol{f}_{i} \end{bmatrix}$ is the contact wrench exerted by the environment on the robot at the link i to which C_i belongs. See [Featherstone, 2014, Chapter 2] for an excellent introduction to wrenches and, more generally, to spatial vector algebra.

1.1.3 Dynamic Equilibrium

Conservation of momentum dictates that the total sum of contact forces equals the total change in linear and angular momentum. In the absence of contact forces, it is impossible for an active body to control the location of its center of mass (CoM).

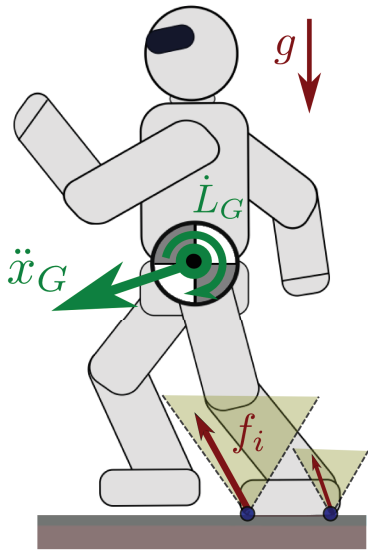


Fig. 1.4 Illustration of the interaction between forces acting on a humanoid robot and its linear and angular momentum (Figure [Caron, 2018])

For a given link i, we write

- m_i and G_i its mass and center of mass (CoM), respectively
- R_i its orientation matrix in the absolute frame;
- ω_i its angular velocity in the link frame;
- $\mathbb{I}_i$ its inertia matrix in the link frame.
- m is the total mass of the robot.

The linear momentum $\boldsymbol{P}_G$ and angular momentum L_G of the system, taken at the CoM G, are defined by:

$$\boldsymbol{P}_G = \frac{1}{m} \sum_{\text{link } k} m_i \dot{\mathbf{p}}_{G_i} \tag{1.7}$$

$$\boldsymbol{L}_G = \sum_{\text{link } i} m_i \vec{GG}_i \times \dot{\mathbf{p}}_{G_i} + \boldsymbol{R}_i \mathbb{I}_i^l \omega_i^l \tag{1.8}$$

The fundamental principle of dynamics states that the dynamic wrench of the robot is equal to the total wrench of forces acting on the system, that is

$$\begin{bmatrix} \dot{\boldsymbol{P}}_G \\ \dot{\boldsymbol{L}}_G \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}^g \\ \mathbf{0} \end{bmatrix} + \sum_{\text{contact } i} \begin{bmatrix} \boldsymbol{f}_i^c \\ \vec{GC}_i \times \boldsymbol{f}_i^c \end{bmatrix} \tag{1.9}$$

where $\boldsymbol{f}_g$ denotes the gravity force and $\boldsymbol{f}_i^c$ is the contact force exerted by the environment on the robot at C_i . This equation is also called dynamic balance or the dynamic equilibrium of the system. It can correspond to the six unactuated components in the equations of motion of the system (robot + environment) [Wieber, 2005].

1.1.4 Quadratic Programming Control

The equations of motion (1.6) along with the dynamic equilibrium (1.9) constrain how the humanoid robot moves dynamically under contact constraints, and how joint torques are affected by contact forces. Controlling the robot now consists in finding a suitable configuration for the actuated degrees of freedom $\boldsymbol{q}^{n-6}$ that achieves a desired behaviour of the robot under those physical constraints (such as moving an end-effector to a specific Cartesian position, while maintaining balance). To do so, it is common to solve a quadratic programming optimization problem [Abe et al., 2007; Bouyarmane et Kheddar, 2011b], *etc*. After giving a brief review of quadratic programming, we will see why it is so well-suited for the control of humanoid robots, and how it can be exploited to do so.

A quadratic program can be written in standard form as the following minimization problem

$$\begin{array}{lll} \text{minimize} & \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x} & \\ \text{subject to} & \boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{h} & \text{inequatlity constraints} \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} & \text{equality constraints} \end{array} \tag{1.10}$$

Here, $\boldsymbol{x}$ is the vector of optimization variables $x_1, \ldots, x_n$. The matrix $\boldsymbol{Q}$ and vector $\boldsymbol{c}$ are used to define any quadratic objective function on these variables, while the matrix-vector couples $(\boldsymbol{G}, \boldsymbol{h})$ and $(\boldsymbol{A}, \boldsymbol{b})$ are used to define inequality and equality constraints, respectively. Vector inequalities apply coordinate by coordinate. In the case where $\boldsymbol{Q}$ is positive definite, the problem is a special case of the more general field of convex optimization, and can be solved efficiently. This enables its use to formulate a dynamically consistent whole-body controller in the context of humanoid robots, or simulated avatars.

We hereby consider the special case of an acceleration-based QP controller, in which the minimization vector $\boldsymbol{x}$ represents the desired joint accelerations $\ddot{\mathbf{q}}$ that cause the robot to move according to a set of quadratic cost functions $\boldsymbol{g}^{(i)}$, while respecting (1.6,1.9). Quadratic objectives regulate the values of kinematic quantities $\boldsymbol{x}_K(\mathbf{q})$ by choosing their accelerations $\ddot{\boldsymbol{x}}_K(\mathbf{q})$ at each time step. The value of each objective $\boldsymbol{g}^{(i)}$ measures the difference between the current $\ddot{\mathbf{q}}$ and the desired $\ddot{\mathbf{q}}_d$ acceleration

$$\boldsymbol{g}^{(i)} = || \ddot{\mathbf{q}} - \ddot{\mathbf{q}}_d || = || \boldsymbol{J}^{(i)} \ddot{\mathbf{q}} + \dot{\boldsymbol{J}}^{(i)} \dot{\mathbf{q}} - \ddot{\mathbf{q}}_d^{(i)} ||, \tag{1.11}$$

where the Jacobian $\boldsymbol{J}^{(i)}$ describes the linear relationship between joint velocities and the velocities of regulated kinematic quantities

$$\dot{\boldsymbol{x}}_K = \boldsymbol{J}^{(i)}\,\dot{\mathbf{q}}$$

Specificities of the QP formulation used in this thesis Throughout this thesis, the weighted quadratic programming controller formulation of Bouyarmane et Kheddar [2011a] will be used. Further details and task definitions can be found in Vaillant et al. [2016] along with Joris Vaillant's thesis. The overall whole-body QP controller can be summed up as follows:

$$\begin{aligned}
&\mathbf{z} = \underset{\mathbf{z}}{\operatorname{argmin}} \sum_{i=1}^{N} w_i \|\mathcal{T}_i(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}})\|^2 + w_\lambda \|\boldsymbol{\lambda}\|^2 \\
&\text{subject to:} \\
&1)\ \text{dynamic constraints } (1.6, 1.9) \\
&2)\ \text{sustained contact positions } (1.5, 1.3) \\
&3)\ \text{joint limits} \\
&4)\ \text{non-desired collision avoidance constraints} \\
&5)\ \text{self-collision avoidance constraints}
\end{aligned} \tag{1.12}$$

The QP decision vector is $\mathbf{z} = (\ddot{\mathbf{q}}, \boldsymbol{\lambda})$, where $\ddot{\mathbf{q}}$ gathers the linear and angular acceleration of floating-base coordinates and the generalized joint velocities. Meanwhile, $\boldsymbol{\lambda}$ denotes the vector of conic coordinates of linearised Coulomb friction cones, such that the contact forces $\mathbf{f}$ are equal to $\mathbf{S}_f\boldsymbol{\lambda}$ with $\mathbf{S}_f$ the span matrix of cone generators. To actuate a position-controlled humanoid robot, such as HRP-4, these desired joint accelerations can then be integrated twice to obtain $\mathbf{q}_D$. This desired position is then tracked by a *proportional-derivative (PD)* controller on joint encoder values.

Here $\mathcal{T}_i$ denotes the residual (difference between actual and desired values) for task i. Each task is given a relative importance by its weight w_i. In case of conflicting objective the overall motion might not be the expected one, as some of the tasks will fail to converge to their final objective. Another approach consists in using a strict hierarchy with equality or inequality constraints, then solved with a hierarchical QP [Escande et al., 2014; Mansard et al., 2009]. Constraints however always take precedence over task objectives, and will be respected. Tasks commonly used include

- A *posture* task $\mathcal{T}_{\text{posture}}$ that steers the joint configuration from its current value $\mathbf{q}^{n-6}$ to a desired configuration $\mathbf{q}_d^{n-6}$.

- A *CoM* task $\mathcal{T}_{\text{CoM}}$ that steers the robot CoM towards a desired position.
- A *surface transform* $\mathcal{T}_{\text{surface}}$ task that steers a robot surface towards another one.
- other tasks $\mathcal{T}_{\text{others}}$

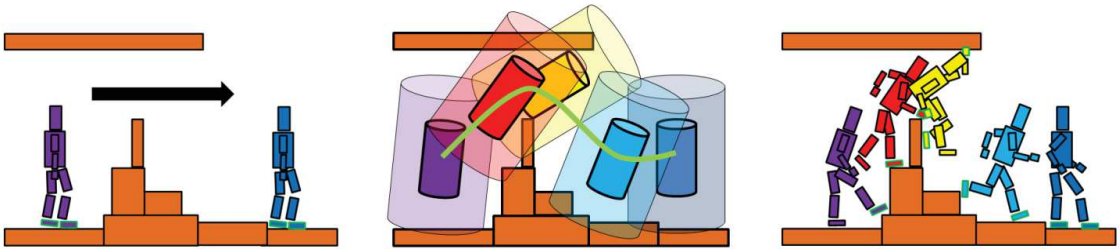
In what follows, tasks are often defined as set-point objective tasks; defined by their associated task-error $\boldsymbol{\varepsilon}_i$ so that $\mathcal{T}_i = \mathbf{K}_{p_i}\boldsymbol{\varepsilon}_i + \mathbf{K}_{v_i}\dot{\boldsymbol{\varepsilon}}_i + \ddot{\boldsymbol{\varepsilon}}_i$ for some stiffness and damping matrices $\mathbf{K}_{p_i}$ and $\mathbf{K}_{v_i}$, respectively.

Such quadratic programming controllers are by nature local. They express feasible motions of the whole-body that respect contact constraints (zero velocity at contact points, and forces within the friction cone) and dynamic equations of motion, while trying to achieve specified tasks. As such, a QP controller is rarely used alone, and its role is commonly to track higher level actions, by generating feasible whole-body motions. In the next section, we will explore how multi-contact planning can be used in conjunction with QP control to achieve complex tasks.

1.1.5 Multi-Contact Planning

Multi-contact planning unifies locomotion and manipulation in a single framework: moving an end-effector towards or away from a contact enables locomotion, applying forces on fixed contacts allows control of the robot's floating base, while moving by keeping contact with a mobile part of the environment allows for manipulation. A general multi-contact planning framework includes two main components: a contact explorer, and a posture generator. The first finds suitable contacts surfaces in the environment that the robot can use to move towards its objective. The second, given these available contacts generates statically-stable postures that respect robot constraints (such as joint and torque limits). By repeatedly establishing and breaking contacts, the robot is then able to move to its objective.

Since early work in Bretl [2006], two promiment multi-contact planning approaches have emerged. In Bouyarmane et al. [2012]; Escande et Kheddar [2009] focuses on physical accuracy, respecting torque limits, collision avoidance, physical plausibility, equilibrium, *etc*. The planner generates a sequence of contacts, along with statically-stable postures that respect the physical constraints imposed. While this formulation allows impressive results, such as the vertical ladder climbing with HRP-2 achieved in Vaillant et al. [2016], it comes at the cost of computational efficiency, which is prohibitive for online planning. Furthermore, practical experience in a wide variety of scenarios (such as DRC tasks) has shown that the plans obtained were in practice rarely applicable without significant manual-tuning.



(a) Initial request (b) Reachability planning (c) Contact sequence

Fig. 1.5 Illustration of multi-contact planning [Carpentier et al., 2016]

Another recent approach is that of Carpentier et al. [2016], where the focus was instead placed on computational accuracy, at the expense of physical accuracy. In the approach, a guide trajectory is first computed by considering a simplified model of the robot, with it's body (waist and torso) represented as a cylinder while it's end-effector reachability is represented with a larger cylinder. This allows to compute efficiently a guide CoM trajectory, along which contacts are sampled [Tonneau et al., 2018], and suitable postures generated (see Figure 1.5). The whole planning process runs at interactive rate, allowing its use with Model Predictive Control (detailed in Section 4.2).

1.1.6 Model Precitive Control

Model Predictive Control (MPC), sometimes called *Preview Controller*, is a widely used optimization method that computes future states of a dynamical system under a set of costs and constraints. In essence, it enables to look for physically feasible actions that will drive the robot from one state to another. For this reason, Model Predictive Control has been widely used to model dynamic walking of humanoid robots. With a suitable model of its dynamics, an MPC can be formulated to compute trajectories of the robot's CoM that are compatible with its footstep placement (in the sense that the robot does not fall). Being able to predict and influence the dynamics of a whole-body humanoid over a large time could be considered as the Holy Grail of humanoid robotics: it would unify planning and control into the same framework, and provide with the ability to plan whole-body dynamic trajectories. Unfortunately, the dynamics of a humanoid robot are very complex, and computing the evolution of such a system in real-time is a challenging problem. Chretien et al. [2016] attempted to improve the speed of whole-body dynamic computation (Equation 1.6) so that it can be itself used a whole-body preview controller. While substancial improvements in computational speed was achieved thanks to efficient GPGPU programming, we are still a long way from achieving it.

Instead, it is common to rely on simplified models to describe the dynamics of humanoid robots in restricted scenarios. For instance, in the case of walking, it is common to represent the robot as a Linear Inverted Pendulum (LIP) [Kajita et al., 2014], where the whole robot body is assimilated to its center of mass, and the swingfoot leg is considered massless. Multiple MPC formulations have been proposed to control such a system [Herdt et al., 2010; Naveau et al., 2017]. These methods all share some common point. The role of the MPC is to compute a CoM trajectory (or equivalently Zero-Moment Point trajectory) going from one footstep to the next, without causing the robot to fall. The main criterion states that the robot is stable if the Zero-Moment Point (ZMP) remains within the convex hull of the support area. If this criterion is not met, the robot can no longer keep balance with its current contacts, and either new suitable contacts need to be established, or it will fall. In addition to computing CoM trajectories, some MPC formulations also include footstep location [Herdt et al., 2010; Scianca et al., 2016].

1.2 Foundations of RGB-D Pose Estimation

Simultaneous Localization and Mapping (*SLAM*) consists in the simultaneous estimation of the state of a robot equipped with on-board sensors, and the construction of a model (the map) of the environment that the sensors are perceiving. In simple case, the robot state is described by its pose (position and orientation), but other quantities such as velocity, uncertainties, *etc* may be included as well. The map is a representation of the environment in which the robot operates (objects, landmarks, surfaces, *etc*). The advantage conferred by the map are two-fold, (i) it can be exploited to support other tasks and (ii) it limits the drift that would be introduced by dead-reckoning by exploiting *loop-closures* (*e.g.* relocating *w.r.t.* to previously known places in the environment).

A comprehensive review of the development of *SLAM* from an early age can be found in two successive surveys [Bailey et Durrant-Whyte, 2006; Durrant-Whyte et Bailey, 2006]. These cover mainly the period from 1986-2004, which saw the introduction of the main probabilistic approaches for *SLAM*, including approaches based on Extended Kalman Filters (EKF-SLAM), Rao-Blackwellized particle filters (FastSLAM [Montemerlo et al., 2002]), and maximum likelihood estimation. It also introduced the challenges connected to robust and efficient data association. A broader review, including the current state of SLAM, and perspectives towards *semantic-SLAM*, where the map embeds higher-level reasoning (objects, situations, *etc*) can be found in Cadena et al. [2016]. The reader wishing to understand the theoretical roots of SLAM arising in probabilistic formulations, and how the realization

that localization is best achieved by simultaneously tracking the pose and building a map is referred to these reviews.

This thesis focuses on the field of *visual* SLAM, and more specifically on *dense visual* SLAM, that is the problem of tracking a vision sensor (camera, LIDAR, *etc*) from all available measurements, and building a dense three dimensional map. A broad review of the main approaches from 2010-2017 can be found in Taketomi et al. [2017].

With the introduction of low-cost RGB-D sensors providing both color and depth information (Asus Xtion, Microsoft Kinect, *etc*), and along with the computational improvements most notably introduced by GPGPU parallel computing, tracking the camera motion based on whole color and depth images and building a dense map that aims at keeping all of the observed information has become popular. In the next section, the main components of forming such kind of pose estimation pipeline will be briefly reviewed, before showing how a complete SLAM can be built from it by introducing the two most common approaches: the keyframe-based method pioneered by Comport et al. [2007], which serves as the basis to many subsequent methods, such as Audras et al. [2011]; Kerl et al. [2013]; Meilland et Comport [2013a], *etc*; and the voxel-based method introduced in Newcombe et al. [2011a].

1.2.1 General overview of pose estimation

The view registration problem, *e.g.* the problem of estimating the 6D pose that related two sets of measurements $\mathbf{M}$ and $\mathbf{M}^*$ obtained from the same scene at different viewpoints, has been widely studied in the field of computer vision. In the case of visual *SLAM*, measurements are obtained at high frequency (typically over 30Hz), and it is assumed that the viewpoints do not vary much between successive measurements, in which case the problem can be solved by a *local pose estimation* framework (detailed hereafter). When the viewpoints are far apart, the local method cannot be reliably used, and the need for *global pose estimation* methods arises. Those will not be considered here, and the reader is referred to the thesis of Ireta [2018] for detailed information.

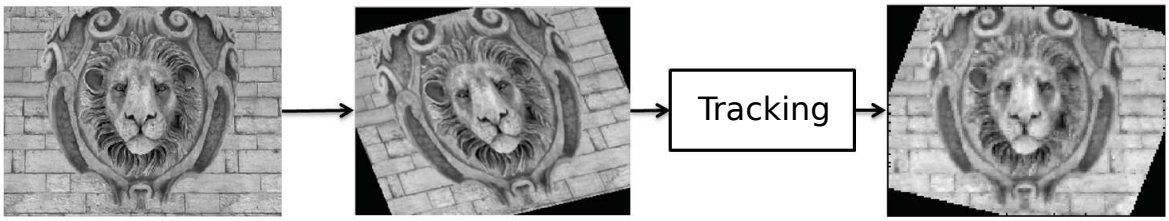


Fig. 1.6 View registration problem. Two measurements $\mathbf{M}$ and $\mathbf{M}^*$ are taken from the same scene. The measurement $\mathbf{M}$ is obtained by moving the sensor by $\mathbf{T}(\mathbf{x})$. If the acquisition model is known, it can be re-projected onto the viewpoint M^* through the warping function ω. The view-registration pipeline aims at estimating the value of $\mathbf{T}(\mathbf{x})$ from the two measurements $(\mathbf{M}, \mathbf{M}^*)$.

The main idea behind solving the *local pose estimation* problem is to find the pose $\mathbf{T}(\mathbf{x})$ that minimizes a re-projection error between both sets of measurements. This implies that the sensor's projection model, that transforms a point P_i in the environment onto a measurement M_i is known. In the case of cameras, the *pinhole camera model* is commonly used. In its most general form, this is often expressed as the following non-linear minimization problem

$$\mathbf{T}(\mathbf{x}) = \underset{\mathbf{T}(\mathbf{x})}{argmin} \sum_{i=1}^{N} ||\mathbf{M}_i^* - \omega(\mathbf{T}(\mathbf{x}), \mathbf{M}_i)||^2 \in \mathbb{SE}(3) \tag{1.13}$$

where $\omega(\mathbf{T}(\mathbf{x}), \mathbf{M}_i)$ is the warping function that transforms a set of measurements $\mathbf{M}_i$ onto a set $\mathbf{M}^w$ by the transformation $\mathbf{T}(\mathbf{x})$. N is the number of measurements that are both in M and M^*. The superscript $*$ will be used throughout this thesis to identify the set of measurements that were obtained first (reference dataset). $\mathbf{T}(\mathbf{x})$ is the homogeneous transformation matrix, which can be decomposed into rotational and translational components.

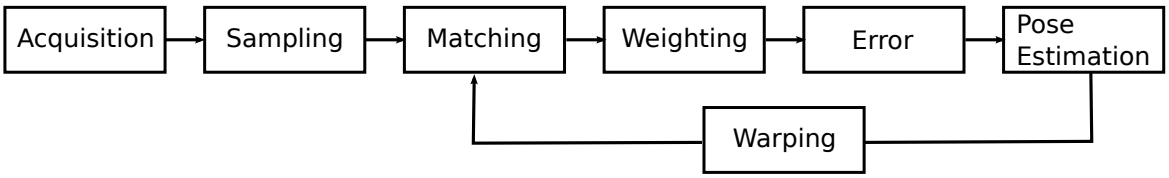
$$\mathbf{T}(\mathbf{x}) = (\mathbf{R}(\mathbf{x}), \mathbf{t}(\mathbf{x})) \in \mathbb{SE}(3)$$

, $\mathbf{x} = [\upsilon_x, \upsilon_y, \upsilon_z, \omega_x, \omega_y, \omega_z] \in \mathfrak{se}(3)$ is the lie-algebra parametrization of the pose. The relationship between both is given by the exponential map as

$$\mathbf{T}(\mathbf{x}) = e^{[\mathbf{x}]_\wedge} \in \mathbb{SE}(3)$$

where $[\cdot]_\wedge$ is the twist matrix operator.

The main components for solving the minimization problem of (1.23) are as follows.



Acquisition and sampling is the process by which a set of points P_i in the environment are observed by the sensor, and transformed into the corresponding measurement $\mathbf{M}$. For a camera, 3D points are projected onto the image plane through the pinhole camera model. If depth measurements are available, this operation is reversible, and the position of the 3D point can be obtained from the measurements. This property is exploited in RGB-D *SLAM* to formulate the warping function.

Warping consists in transforming a measurement $\mathbf{M}$ into a measurement $\mathbf{M}^{\mathbf{w}} = \omega(\mathbf{M}, \mathbf{T}(\mathbf{x}))$, where $\mathbf{T}(\mathbf{x})$ is the solution to the IRLS problem of Equation 1.13. When a solution is found, the measurements in $\mathbf{M}^w$ coincide with those in $\mathbf{M}$, that is the solution that brings one set of measurement with its reference has been determined.

Matching refers to the problem of finding correspondences between $\mathbf{M}^w$ and $\mathbf{M}^*$. For local pose estimation, it is common to select the nearest-neighbour, as the viewpoints are assumed to be close. The problem is much harder in case of global registration.

Weighting is the process by which the influence of outliers is reduced. The importance of measurements suspected to be outliers is reduced by setting a correspondingly low weight, while valid points are considered with a high weight. The weighting is typically determined by robust M-Estimators [Huber, 2011].

Error The error between measurements $\mathbf{E}(\mathbf{T}(\mathbf{x})) = [\mathbf{e}_1\ \mathbf{e}_2\ \cdots \mathbf{e}_n]^\top$, with $\mathbf{e}_i = \mathbf{M}_i^* - \omega(\mathbf{T}(\mathbf{x}), \mathbf{M}_i)$ is a function whose norm monotonically reduces as the estimated pose $\mathbf{T}(\mathbf{x})$ becomes closer to the transformation undertaken by the sensor. The formulation of the cost function depends on the data acquired. In the case of RGB images, the photometric error, that is the difference in intensity between neighbouring pixels is usually considered, while depth images are typically treated by Iterative Closest Point (reviewed extensively later). No matter the formulation, its gradient is used to drive the minimization in a direction that reduces the overall error, that is at each step, the estimate gets closer to the pose to be estimated. +

Pose estimation Equation 1.23 is commonly solved using *non-linear iteratively re-weighted least squares* (IRLS) which is an effective algorithm for solving L_p norm problems and is well-suited in that context. A general detailed explanation is provided in Appendix A. The error $\mathbf{e}(\mathbf{T}(\mathbf{x}))$ is iteratively minimized using a robust Gauss-Newton approach with M-Estimators. Each iteration computes a transformation $\mathbf{x}$ that brings the measurement M^w in closer alignment with $\mathbf{M}^*$ as

$$\mathbf{x} = -(\mathbf{J}^\top \mathbf{W}\mathbf{J})^{-1}\mathbf{J}^\top \mathbf{W}\mathbf{e}(\mathbf{T}(\mathbf{x})) \in \mathbb{R}^6 \tag{1.14}$$

where $\mathbf{J}$ represents the stacked Jacobian matrices obtained by derivating the stacked error functions $\mathbf{e}(\mathbf{T}(\mathbf{x}))$. The Jacobian is obtained by partial derivation of the error function as

$$\mathbf{J} = \frac{\partial \mathbf{E}(\mathbf{T}(\mathbf{x}))}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{e}_1}{\partial \mathbf{x}_1} & \cdots & \frac{\partial \mathbf{e}_1}{\partial \mathbf{x}_6} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{e}_n}{\partial \mathbf{x}_1} & \cdots & \frac{\partial \mathbf{e}_n}{\partial \mathbf{x}_6} \end{bmatrix} \in \mathbb{R}^{n\times 6} \tag{1.15}$$

The weight matrix $\mathbf{W}$ contains the stacked weights ρ_i associated with each set of coordinates obtained by M-estimation [Huber, 2011] as

$$\mathbf{W} = \begin{bmatrix} \rho_1 & 0 & \cdots & 0 \\ 0 & \rho_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \rho_n \end{bmatrix} \in \mathbb{R}^{n\times n} \tag{1.16}$$

The solution is then updated as

$$\mathbf{T}(\mathbf{x}) = \hat{\mathbf{T}} e^{[\mathbf{x}]_\wedge}$$

, where $\hat{\mathbf{T}}$ is the solution obtained at the previous iteration.

1.2.2 Obtaining RGB-D images

In the previous paragraph, the general framework for local pose estimation has been presented. Let's now consider the special case of estimation from RGB-D sensors, which will be used throughout this thesis. RGB-D sensor is considered in a large sense here, as any sensor that is able to obtain both a color image, and a depthmap.

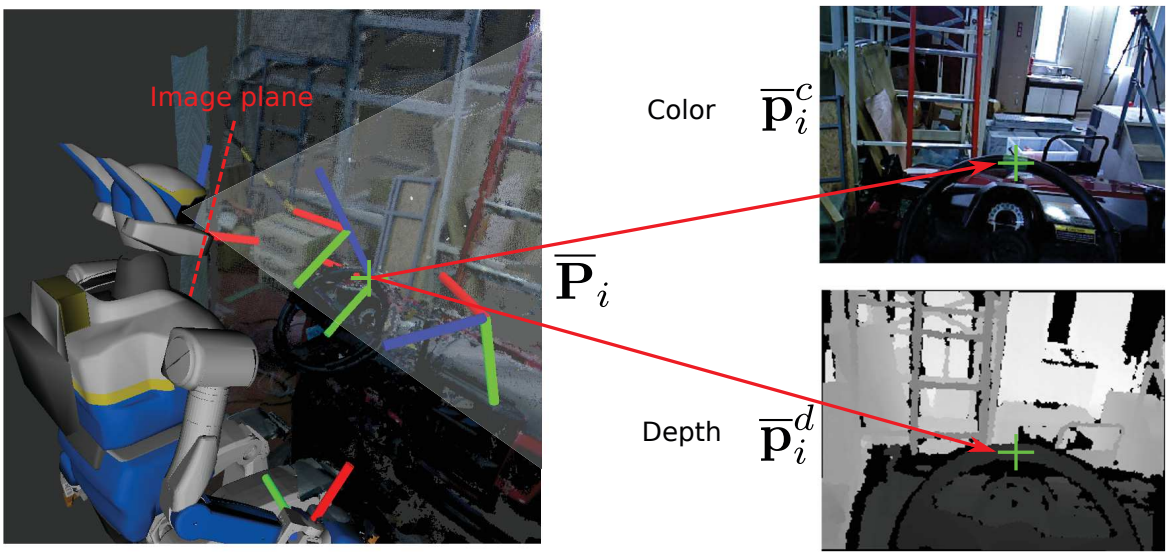


Fig. 1.7 A 3D point $\mathbf{P}_i$ can be projected onto an RGB image forming a point with pixel coordinates $\overline{\mathbf{p}}_i^c \in \mathbb{R}^3$ of intensity $\mathbf{I}_i$, and onto a depthmap with coordinates $\overline{\mathbf{p}}_i^d \in \mathbb{R}^3$ with depth Z_i. The information contained in color and depth images, along with the camera projection model allows to revert the process and obtain 3D points from the images.

RGB images An image captured by a camera sensor encodes information about the light reflected off each observed 3D point, such as its intensity, color, or other properties. Each ray that reaches the camera lens is projected towards the lens focal point. Crossing the path of these light rays, on the focal plane, a sensitive sensor (such as a CCD array) converts properties of the light (intensity, color) into electrical signals, that can be digitalized.

Geometrically, image formation is commonly modeled with a pinhole camera model (Figure 1.8). Each visible 3D point $\mathbf{P}_i = [X_i\ Y_i\ Z_i]^\top$ is projected onto the image plane at the pixel coordinates $\overline{\mathbf{p}}_i = [u_i, v_i, 1]^\top$ as follows.

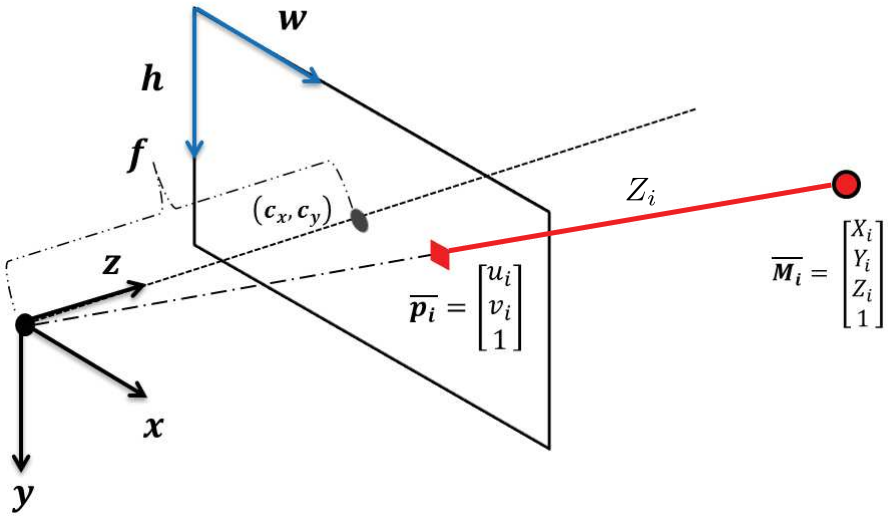


Fig. 1.8 Pinhole Camera Model - The coordinates of a 3D point $\mathbf{M}_i$ are projected onto a 2D frame with pixel coordinates $\mathbf{p}_i$. The model of a pinhole camera is employed, where (c_x, c_y) are coordinates (in pixels) usually placed at the center of the $w \times h$ frame and f is the focal distance; the skew factor s_θ of a pixel is the angle between the image axis **h** and **w**. Figure modified from [Ireta, 2018, p59]

$$Z_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} fw & fs_\theta & c_x \\ 0 & fh & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \tag{1.17}$$

where f is the focal distance, s_θ is the skew angle of a pixel (which is usually considered equal to 0), w and h are the width and height of the image, respectively and c_x, c_y are the coordinates of the center of the image. These constitute the intrinsic parameters of the camera. For simplicity, (1.17) can be written as

$$Z_i \overline{\mathbf{p}_i} = \mathbf{K} \Pi_3 \overline{\mathbf{P}_i}$$

Note that the pinhole model is an approximation of the image formation process, but does not model all of the physical properties of image formation (it does not include, for example, geometric distortions or blurring of unfocused objects caused by lenses and finite sized apertures). It is however a sufficient model for most RGB-D sensors, including the Asus Xtion that will be used in practice in this thesis.

It is interesting to note here that the information contained in the image (pixel coordinates (u_i, v_i) and intensity I_i) is not sufficient to uniquely invert the model and obtain a unique corresponding 3D point. Instead, a line passing through the focal point and the pixel

p_i is obtained, and the 3D point can be anywhere on that line. Uniquely obtaining the corresponding 3D point requires knowing its depth, represented by Z_i. Let's now consider how this depth information can be acquired and stored.

Depthmaps For RGB-D pose estimation, it is interesting to know the depth of a point in the RGB image, such that a corresponding 3D point can be uniquely defined. Thus, a convenient representation for depth information is that of a depthmap, which is an image storing for each pixel coordinate the distance Z_i from the focal plane to the observed 3D point. Since Z_i is known, it becomes possible to convert pixel coordinates into a unique 3D point, and conversely using the pinhole model of Equation 1.17.

Obtaining the depth information Z_i has been achieved with several technologies and methods. In recent years, infrared-sensors exploiting structured-light, such as the Asus Xtion, Microsoft Kinect, or Intel RealSense have become popular. The depth information is inferred by observing the deformation of a regular pattern of infrared light caused by the scene's geometry. Time-of-flight (ToF) sensors are also common, and estimate distances by measuring the time that the light needs to travel from the light source to the scene and back to the camera. When no sensor providing depth information is available, it can be estimated from RGB images. When two RGB cameras are available, this is referred to as stereo-vision, where correspondences between each pixel in the images are found, and used to geometrically estimate the depth by triangulation, if the baseline between both cameras is known. When a single camera is available, estimating the depth information from motion remains possible, but up to a scale factor (*e.g.* the metric depth cannot be known), which makes monocular SLAM methods [Engel et al., 2014; Mur-Artal et al., 2015] possible.

Note that pixel coordinates $\mathbf{p}_i^d$ in the depthmap and those in the RGB image $\mathbf{p}_i^c$ do not necessarily correspond, as they may acquired by different sensors that do not share the same optical center (such as the RGB and IR receptors). However, given a known transformation between both optical frames, it is always possible to reproject the depthmap from the corresponding RGB viewpoint to form the new depth image such that $\mathbf{p}_i^d = \mathbf{p}_i^c$, which, as will be seen in the next section is necessary for RGB-D pose estimation. From now on, unless explicitly required, pixel coordinates will be denoted as $\mathbf{p}_i$ in both the depthmap and RGB images, and refer to the same corresponding 3D point.

1.2.3 Pose Estimation from RGB-D images

When frame-to-frame RGB-D registration is performed, the frame-rate of acquisitions allows to maintain sufficient overlap between consecutive images to formulate a direct error function,

comparing (i) photometric information from RGB images and (ii) geometric information from depth images [Tykkälä et al., 2011].

The advantages *w.r.t.* to early methods that were considering only keypoints in RGB images are many. First, the need to design robust keypoint descriptors, that can be used to robustly find correspondences between images acquired with different viewpoints is removed. Second, none of the acquired information is discarded, leading to a more robust formulation. Finally, considering both photometric and geometric information together provides great advantages in terms of robustness. When a scene is lacking textures, but has plenty of geometry, the geometric cost constrains the pose estimate, and conversely, when little geometry is present but a lot of texture is available, the pose is also well constrained. When the scene is rich in both texture and geometry, the situation then becomes ideal and the pose will be constrained by both. Thanks to this, in most cases the pose estimation problem will almost always be well constrained. Specific failure cases still exist, such as when neither geometry nor texture is available, or when observing geometry from far away with little texture. In practice, these situation are quite rare, making such a tracking formulation well-suited for robotics applications.

Let's now consider the formulation of these two cost functions, and how they can be exploited to obtain the motion of an RGB-D sensor.

Photometric Error

An error function based on photometric measurements can now be defined [Comport et al., 2007]. The brightness of each pixel $\mathbf{p}_i$ is denoted as $\mathbf{I}_i$. As shown in Figure 1.9, a reference image represented by $\mathcal{I}^* = [\mathbf{I}_1^*, \mathbf{I}_2^*, \dots, \mathbf{I}_{mn}^*]$ and a current image by $\mathcal{I} = [\mathbf{I}_1, \mathbf{I}_2, \dots, \mathbf{I}_{mn}]$ are established. A corresponding pointcloud $\mathbf{P}^*$ can is obtained through the pinhole model (Equation 1.17). A new image $\mathcal{I}^w$ can be synthesized between two positions by projecting the reference poincloud $\mathbf{P}^*$ into the current image $\mathcal{I}$ and then by linearly interpolating the intensity values on a regular array such as:

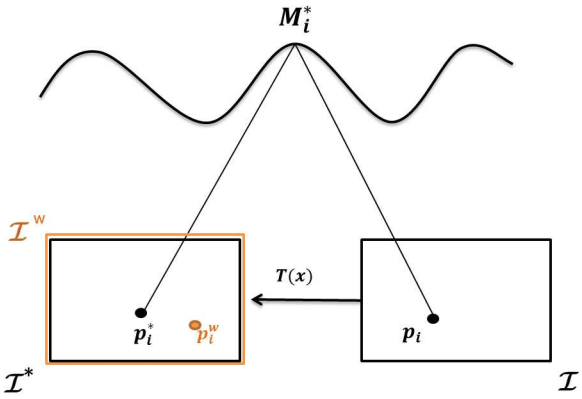


Fig. 1.9 For the intensity-based minimization of the error function e_I, a new image $\mathscr{I}^w$ is synthesized from $\mathscr{I}$ based on the pose $\mathbf{T}(\mathbf{x})$. $\mathscr{I}^*$ is the reference image (Figure [Ireta, 2018]).

$$\mathscr{I}^w = \mathscr{I}\left(\omega(\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x}), \mathbf{P}^*)\right) \tag{1.18}$$

where the geometric warping function $\omega(\cdot)$ obtain the warped pixels such as:

$$\overline{\mathbf{p}_i}^w = \frac{\mathbf{K}\Pi_3\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x})\overline{\mathbf{M}_i}^*}{\mathbf{e}_3^\top\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x})\mathbf{M}_i^*} = \begin{bmatrix} u_i^w \\ v_i^w \\ 1 \end{bmatrix} = \begin{bmatrix} c_x + f_x X_i^w/Z_i^w \\ c_y + f_y X_i^w/Z_i^w \\ 1 \end{bmatrix} \in \mathbb{R}^3 \tag{1.19}$$

where $\mathbf{K}$ is the intrinsic calibration matrix and $\Pi_3 = [\mathbb{I}_{3\times 3}, \mathbf{0}_3]$ projects the 4×4 pose matrix onto the 3×4 space and $\mathbf{e}_3^\top = [0\ 0\ 1]$ extracts the depth component of a 3D point. Generally, the warped points $\mathbf{p}_i^w$ do not correspond to an exact pixel coordinate and their intensity $\mathbf{I}(\mathbf{p}_i^w)$ is interpolated from the surrounding pixels (typically with bilinear interpolation).

A photometric cost function, based on the difference of intensities between neighbouring pixels can then be defined as

$$\mathbf{e}_{I,i} = \mathbf{I}^*(\mathbf{p}_i^*) - \mathbf{I}(\mathbf{p}_i^w) \tag{1.20}$$

and used as the cost-function in Equation 1.23 to estimate the pose. A photometric Jacobian can be obtained by partially deriving 1.20 at each pixel as:

$$\mathbf{J}_I = \nabla \mathbf{I}_i^w \frac{\partial \mathbf{p}_i^w}{\partial \mathbf{P}_i^w} \frac{\partial \mathbf{P}_i^w}{\partial \mathbf{T}(\mathbf{x})} = \begin{bmatrix} \frac{\nabla I_{x_i}^w f_x}{Z_i^*} \\ \frac{\nabla I_{y_i}^w f_y}{Z_i^*} \\ \frac{-\nabla I_{y_i}^w Y_i^* f_y - \nabla I_{x_i}^w X_i^* f_x}{Z_i^{*2}} \\ -\nabla I_{y_i}^w f_y - \frac{Y_i^* \left(\nabla I_{x_i}^w X_i^* f_x + \nabla I_{y_i}^w Y_i^* f_y\right)}{Z_i^{*2}} \\ \nabla I_{x_i}^w f_x - \frac{X_i^* \left(\nabla I_{x_i}^w X_i^* f_x + \nabla I_{y_i}^w Y_i^* f_y\right)}{Z_i^{*2}} \\ \frac{\nabla I_{y_i}^w X_i^* f_y - \nabla I_{x_i}^w Y_i^* f_y}{Z_i^*} \end{bmatrix}^\top \in \mathbb{R}^{1\times 6} \tag{1.21}$$

where $\nabla \mathbf{I}_i^w = [\nabla I_{x_i}^w, \nabla I_{y_i}^w]$ is the gradient of the image.

Geometric Error

Consider a reference $\mathscr{D}^*$ and a current $\mathscr{D}$ depthmap, corresponding respectively to the RGB image $\mathscr{I}^*$ and $\mathscr{I}$ (that is, each pixel represents the same 3D point). The depthmap contains the metric distance $Z_k \in \mathbb{R}^+$ for each pixel, and a corresponding 3D euclidean point $\mathbf{P}_i$ can be computed according to the pinhole model (Equation 1.17). Let's consider the pointcloud associated to the depthmaps as measurements $\mathbf{M}^* = [\mathbf{P}_1^*, \mathbf{P}_2^*, \ldots, \mathbf{P}_{mn}^*]$ and $\mathbf{M} = [\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_{mn}]$. Similarly to the photometric error, an euclidean distance between the two pointclouds is considered, such that it is minimal when the clouds are brought into alignment by the estimated pose. A choice must be made here between transforming the current cloud onto the reference, or the opposite. Both have advantages, the first is more computationally efficient as some expensive computations such as normal estimation can be performed only once on the reference dataset; the second is more computationally intensive but more accurate, as in the context of *SLAM*, the reference does not necessarily correspond to a raw RGB-D measurement, but can be improved over time by integrating data from the frames being tracked.

$$\mathbf{e}_{G_i} = \mathbf{M}_i - f\left(\mathbf{T}(\mathbf{x}), \overline{\mathbf{M}_i^*}\right) \in \mathbb{R} \tag{1.22}$$

This problem is commonly referred to as *Iterative Closest Point* algorithm, and was first introduced by Besl et McKay [1992]. The algorithm is reviewed in details in Appendix B in the more general case of aligning two clouds of points, and also used extensively in Chapter 3. Several metrics have been proposed for the distance, the most common being point-to-point

(distance between a point $\mathbf{M}_i$ and its nearest neighbour $\mathbf{M}_i^*$ in $\mathbf{M}^*$, point-to-plane (distance between a point $\mathbf{M}_i$ and it's projection on the tangent plane at $\mathbf{M}_i^*$) and plane-to-plane (distance between both tangent planes), as shown in Figure 1.10. Point-to-plane and plane-to-plane method tend to have a larger convergence radius, that is they are more robust to larger change of viewpoint. See Appendix B for details about their formulation, and how nearest neighbour points can be obtained.

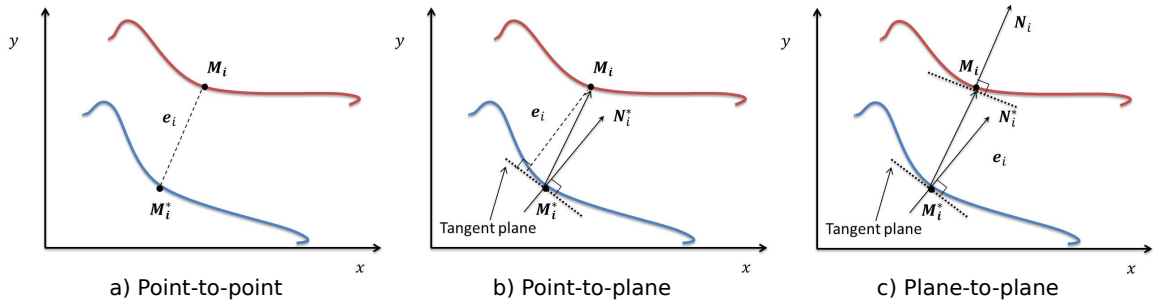


Fig. 1.10 Common error metrics for geometric pose estimation with Iterative Closest Point (Figure [Ireta, 2018])

1.3 Keyframe-based Dense Visual SLAM

In the previous section, we have seen the general method to dense tracking of RGB-D images. Let's now consider how it can be extended beyond tracking to build large-scale maps of the environment, and localize within that map. In this section, we focus on keyframe-based approaches, pioneered by Comport et al. [2007], and since then widely adopted [Engel et al., 2014; Kerl et al., 2013; Meilland et Comport, 2013a; Mur-Artal et al., 2015], *etc*. The main *SLAM* method used throughout this thesis is that of the multi-keyframe proposed in [Meilland et Comport, 2013a], that has since been transferred to the industry via the startup PIXMAP. This method was selected for its demonstrated performance, and its robustness to outliers with M-Estimators, to motion-blur [Meilland et al., 2013b], and it's ability to obtain and exploit high-dynamic range images [Meilland et Comport, 2013a]. Additionally, super-resolution maps [Meilland et Comport, 2013b] can be generated (that is maps at a higher resolution than that of the sensor itself).

1.3.1 Overview of the keyframe-based formulation

The main idea behind the keyframe-based formulation is that instead of tracking successive RGB-D frames, which amounts to pure dead-reckoning odometry, a reference frame $\mathscr{K}_i = (\mathbf{I}_i^*, \mathbf{D}_i^*)$ is selected. The next $k \in \mathbb{R}^+$ frames, denoted as $\mathscr{F}_k = (\mathbf{I}^k, \mathbf{D}^k)$. These k frames

are then tracked *w.r.t.* to that reference keyframe using the photometric and geometric errors described previously, until robust tracking can no longer be achieved by keeping that reference frame. In the studied method, a threshold on the median-absolute-deviation of the error is used. Keyframes are continuously improved by re-projecting the current frame onto the reference keyframe, and fusing their data (*e.g.* filling holes in the depthmap, improving its accuracy, generating super-resolution textures, *etc*). When tracking can no longer be achieved *w.r.t.* to the $i-th$ keyframe, a new one $\mathcal{K}_{i+1} = (\mathbf{I}_k^*, \mathbf{D}_k^*)$ is selected by taking the last successfully tracked frame as the new reference. For better performance, the n closest keyframes can be used to predict the current keyframe (see Meilland et Comport [2013a]). A keyframe-graph is updated, with the new keyframe as a node, and linked to the previous keyframe node by an edge that stores the estimated transformation $\hat{\mathbf{T}}$ between both keyframes. The local pose estimation framework can then continue to be used *w.r.t.* to the new keyframe, and it's pose *w.r.t.* to the initial keyframe can still be determined thanks to the keyframe graph.

1.3.2 Tracking

For simplicity of notation, the superscript k will be omitted from the following equations. The motion of each frame k *w.r.t.* to the keyframe is estimated as

$$\mathbf{e}_i(\mathbf{x}) = \begin{bmatrix} \alpha \left(\mathbf{I}_i^*(\mathbf{P}_i^*) - \mathbf{I}_i \left(w(\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x}), \overline{\mathbf{P}}_i^*) \right) \right) \\ \mathbf{N}_i^{*\top} \left(\mathbf{P}_i^* - \Pi \widehat{\mathbf{T}}\mathbf{T}(\mathbf{x}) \overline{\mathbf{P}}_i \right) \end{bmatrix} \in \mathbb{R}^2, \tag{1.23}$$

where the first row of equation (1.23) is the photometric term relating the reference and current images ($\mathbf{I}^*$ and $\mathbf{I}$) and the second row is a point-to-plane ICP error with projective data association. The surface normals $\mathbf{N}_i^*$ are computed for the reference image from the time integrated set of 3D points $\mathbf{P}^*$. Π projects to non-homogeneous coordinates (see equation 1.17). The function $w(\cdot)$ is the warping function that transforms the current image to the reference, based on the current pose estimate. $\widehat{\mathbf{T}} \in \mathbb{SE}(3)$ represents the latest pose estimate, and $\mathbf{T}(\mathbf{x})$ the current pose increment; α is the weight which defines the relative uncertainty between depth and image measurements.

This nonlinear error is iteratively minimized using the Gauss-Newton method (see equation 1.3.2)

$$\mathbf{x} = -(\mathbf{J}^\top \mathbf{W} \mathbf{J})^{-1} \mathbf{J}^\top \mathbf{W} \mathbf{e}(\mathbf{T}(\mathbf{x})) \in \mathbb{R}^6$$

, where $\mathbf{J}$ contains the stacked Jacobian matrices of the errors of equation (1.23), and $\mathbf{e}$ is the stacked error vector, and $\mathbf{W}$ is the robust weighting matrix. The pose estimate $\widehat{\mathbf{T}}$ is finally

updated using a homogeneous update until convergence as $\widehat{\mathbf{T}} \leftarrow \widehat{\mathbf{T}}\mathbf{T}(\mathbf{x})$. This estimation process is highly local, the viewpoint on the current image needs to remain close to that of the reference frame, and may be subject to local minima.

1.3.3 Keyframe-Graph

All keyframes $\mathscr{K}_0, \ldots, \mathscr{K}_N$, where N is the number of keyframes are stored in a graph. Its nodes contain the refined keyframes $(\mathbf{I}_i^*, \mathbf{D}_i^*)$, and its edges are composed of the estimated rigid transformation between the keyframes. This provides a global representation of the observed environment, where each keyframe stores a specific viewpoint, and the transformation between all viewpoints is known. This way of storing the map information is very efficient, as keyframes can be directly used for the tracking process, and continuously improved over time as new data is observed.

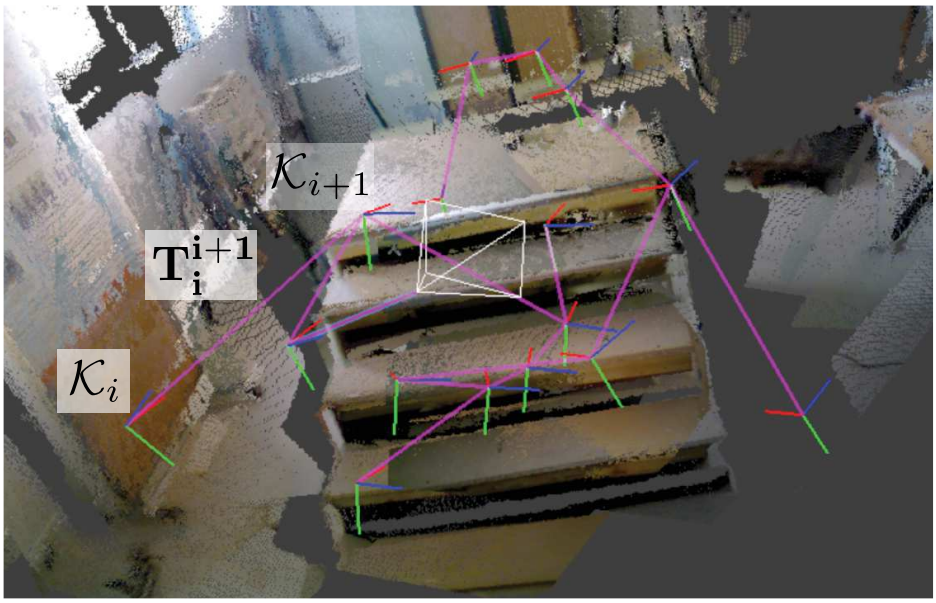


Fig. 1.11 Visualization of the keyframe graph generated from a handheld Asus Xtion RGB-D sensor while observing a mockup staircase of Airbus' final demonstration for the COMANOID project. Each node represents a keyframe $\mathscr{K}_i$, and are related by a transformation $\mathbf{T}_i^{i+1}$ estimated by dense RGB-D tracking (purple lines). The keyframe poses are continuously optimized by exploiting loop-closures (dotted black).

Furthermore, the graph offers many more advantages. It is always possible to go back to a 3D representation by re-projecting each pixel of the keyframes into a common reference frame through the pinhole camera model and the transformations between keyframes (see Section 3.1.2). But one of the main advantages of the keyframe-graph, is that it makes it possible to create a topological map of the environment, where places previously visited can

be recognised, and correctly linked by edges in the graph. The recognition of previously visited places is a problem known as *loop-closure*, and in the case of a keyframe-graph representation, drift detected by loop-closure is rather easy to correct. It is common to use a 2-stage approach for efficiency. First, keypoint descriptors are used to compare the keyframes (here fern descriptors are used), and find likely candidates. Then the tracking method can be used to estimate the relative pose between each of the keyframes. If one keyframe can be successfully re-projected onto an other, a loop closure is formed. Notice that by finding the transformation between the current keyframe and a previously observed one, we have formed a closed-loop in the graph, and the observed drift accumulated within that loop. Thanks to this, the drift can now be corrected with a bundle-adjustment method, where the transformations between each keyframe in the loop are optimized together to correct the drift.

1.3.4 Volumetric SLAM

An alternative volumetric approach to *SLAM* has been concurrently developed to keyframe-based appraoches. Contrary to keyframe-based approaches, that do not store an explicit representation of the 3D structure of the environment, volumetric *SLAM* approaches aim to maintain a continuous representation of the observed 3D surfaces. In the original algorithm, KinectFusion [Newcombe et al., 2011b], a physical space of $3m^3$ is subdivided into a fixed preallocated grid of 512^3 voxels. A variant of the *Trucated Signed Distance Function* [Curless et Levoy, 1996] (specifying a relative distance to the actual surface) is used to integrate and efficiently store every depth measurement into the voxel grid. Camera tracking is achieved with Generalized ICP Segal et al. [2009], similarly to Section 1.2.3. The approach has since then been extended to handle moving volumes Roth et Vona [2009], which allows dense volumetric modelling over an extended area by virtually translating the volumetric model as the sensor moves. This led to the formulation of Kintinous [Whelan et al., 2012], reaching similar capabilities to keyframe-based *SLAM*, and the added ability to reconstruct a 3D mesh of the environment online. Loop-closures and the ability to generate super-resolution maps were later added in Whelan et al. [2013] and Whelan et al. [2015a], using a pose graph and volumetric fusion.

Both volumetric and keyframe-based methods have reach a similar set of features, and their tracking and mapping performance are on par with each other. Volumetric methods have the slight advantage of building an explicit continuous representation of the surfaces in the environment, at the cost of increased computational power. Recently, this representation has been leverage to reach some impressive extensions to dynamic scenes have recently been presented [Newcombe et al., 2015] by exploiting the volumetric representation.

1.3.5 SLAM without a pose graph

Recently, an alternative approach that does not rely on a pose graph has been proposed [Whelan et al., 2015b], and demonstrated tracking and mapping results on-par with the aforementioned state-of-the art approaches. It attempts to move away from the classical pose-graph formulation, and adopt a map-centric approach. The model is segmented into two regions, an active region corresponding to the most recently added measurements, and a an inactive region corresponding to older measurements. Loop-closures are achieved by attempting to register active regions onto inactive ones. If this is successful, a loop-closure is formed, and the model is non-rigidly deformed to reflect this registration. Tracking is achieved with photometric and geometric pose estimation between the current measurement and the active part of the model. Additionally, a surfel-based representation is used in-stead of the traditional pixel-based representation. This allows to store higher-level information (colors, normals, uncertainty, timestamps, *etc*) that can used to inform tracking, registration, and other algorithm such as segmentation. For example, Dense Planar SLAM [Salas-Moreno et al., 2014] exploits a surfel-based representation for online segmentation of planes in the map.

1.4 Towards *SLAM* in humanoid robotics

For the purpose of robustly localizing the sensor *w.r.t.* to its environment, and exploiting map information, both volumetric and keyframe-based approaches provide essentially the same capabilities. Fully exploiting the robot localisation *w.r.t.* to the map, and information provided in the map to inform planning and control methods has the potential to change the face of humanoid robotics, and solve the recurring problems caused by open-loop model-based control. At the time of undertaking the work presented in this thesis, few works had considered its use for both planning and closed-loop control. This is mainly due to the fact that the most significant contributions and available implementation of robust dense visual *SLAM* have only been recently introduced in the years 2007-2013.

Early work mostly considered improving the performance of *SLAM* by exploiting planning information about the robot motion. In Stasse et al. [2006], the output of a walking pattern generator was exploited to both initialize a monocular *SLAM*, and improve its performance with an EKF formulation. Recently, a similar endeavour has been achieved to robustify ElasticFusion [Scona et al., 2017] with a low-drift kinematic-inertial estimator, that provides motion-prior and improves robustness to lack of features, and illumination changes. It's used was demonstrated with a closed-loop Cartesian-regulated walking controller.

The need to exploit the map is obvious. In Stasse et al. [2009] a method was developed for stepping over obstacles, and in Baudouin et al. [2011] a more generic apprach to replanning in the presence of obstacles was presented, but the position of obstacles was obtained using a MOCAP system. Walking over rough terrain was demonstrated in Fallon et al. [2015], and constitutes one of the first applications, apart from those presented in this thesis, where a dense map obtained from dense visual *SLAM* is exploited for online planning. Here, the Kintinous algorithm is used to obtain the map, and footsteps are segmented by detecting large enough planar sections within the map and localization is obtained with a kinematic-inertial state estimator [Fallon et al., 2014]. In this work the map is used to plan the next footsteps, but no attempt is made to observe, and react to perturbations by modifying the footstep plan. This was later proposed in Feng et al. [2016], where the center-of-mass (CoM) and center-of-pressure (CoP) are estimated using EFK with IMU measurements (without *SLAM*) [Xinjilefu et al., 2015] and model-predictive control is exploited to recompute footsteps online. It made the case that reactive footstep planning was far more important and flexible in the context of fast-walking humanoid robots than traditional postural stabilization methods, and that it considerably lowers the requirements on precise CoP and CoM tracking. This makes a strong-case for the exploitation of *SLAM* with model-predictive walking controllers. In Chapter 4, we propose a similar method based on a simple SLAM-based estimator, and demonstrate that is can achieve similar performance while enabling the use of its map to inform collision avoidance, and walking objectives.

Recent developemnents in multi-contact planning, give hope that the community can one day reach successful online planning [Carpentier et al., 2016] in any environment, with no human intervention. But for now, existing methods remain model-based, and an explicit representation of the environment needs to be provided (often in the form of CAD models). This greatly restrict the ability of robots to explore previously unmodeled environments. Preliminary work has started to emerge, attempting to exploit the environment observed by vision sensors. In Brossette et al. [2013], flat horizontal surfaces are extracted in the form of a convex polygonal surface from a pointcloud obtained from a single RGB-D sensor image (providing color and depth information), and a multi-contact plan that accounts for collisions by exploiting the pointcloud's convex hull, is generated using a modified version of Escande et Kheddar [2009]. The reported planning-time for an experiment climbing up a 3-step staircase made up of random furniture pieces is 98.4s for a version of the plan using only the robot feet, and 122.3s using the upper limbs as well. While far from real-time, this work shows that existing planning methods can be extended to take into account observed environments, by devising a vision algorithm that can provide high-level information about the environment (suitable surfaces, convex hulls for collisions, ...). This observation is one

of our main motivations for exploiting the capabilities of *SLAM*, whose map can be efficiently exploited to provide suitable information to the planner. Our own approach is presented in Chapter 3 where complex multi-contact plans generated offline are exploited to control the robot *w.r.t.* to the map.

This concludes the review of the state-of-the-art of humanoid robotics and *SLAM*, and why it has been deemed important to further explore the use of *SLAM* to improve multi-contact planning and control of humanoid robot through the use of both localization and map information. Let's now consider the first problem, that of calibration, which is a prerequisites to the localization of any of the robot limbs *w.r.t.* to the map.

Chapter 2

Eye-Robot Autonomous Calibration

In the previous chapter, the importance of exploiting localization *w.r.t.* to *SLAM* map was outlined. As will be seen in this chapter, a prerequisite for localizing any of the robot bodies within that map is the precise knowledge of its kinematic tree geometric parameters, including the position of the vision sensors with respect to the body. In this chapter, we present a novel approach that extends the well known Eye-Hand calibration to the online whole-body calibration of the kinematic tree geometric parameters, which we similarely name Eye-Robot calibration. Only an on-board RGB-D sensor and joint encoders are required. Online calibration allows to estimate the state of the kinematic tree at any time and thus account for inaccurate models, passive joints, mechanical wear, unexpected damages (see Figure 2.1), *etc*. One major challenge in achieving such an online self-calibration procedure, with the available sensors, is that the observability of the calibrated parameters cannot always be guaranteed. In this work, we determine the effect of joint degrees of freedom on observability. From this, we propose a novel Eye-Robot calibration method that determines the geometric transformations between joints. Conditions on joint motion are further used to improve upon existing kinematic tree parameters when observability is incomplete. In practice a dense SLAM algorithm is used for online pose estimation and the results are demonstrated with an HRP-4 humanoid robot.

2.1 Introduction

Robotic systems inherently require some sort of calibration procedure to determine parameters that are needed to perform state-estimation, planning and control. One of the most fundamental set of parameters is kinematics, which relates sensors to actuators through the geometric configuration of the robot. Whilst this model is most generally provided by the manufacturer thanks to precision machining, some parameters will change over time. This

Fig. 2.1 During a stair-climbing experiment, HRP-4 fell on it's supporting rope, which damaged the mechanism holding the Asus Xtion in place. This illustrates the importance of having an autonomous online calibration method for robust operation in real envionments.

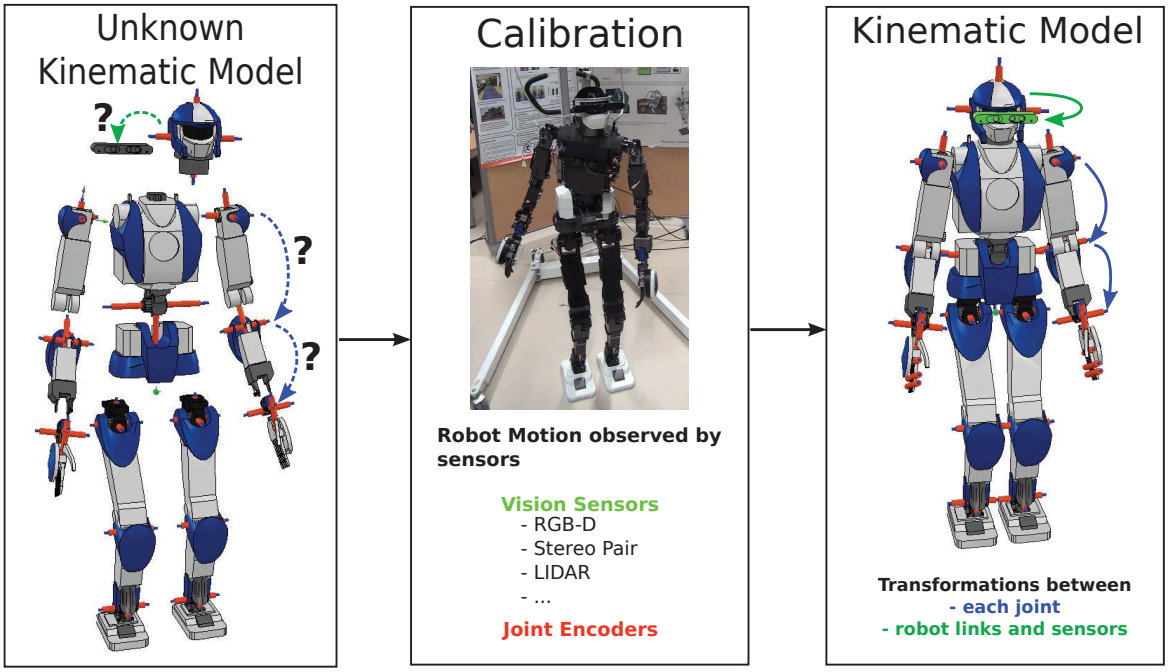


Fig. 2.2 Overview of the *Eye-Robot* calibration method. Transformations between each joint (blue arrows), and between joints and sensors (green arrow) are initially unknown. The proposed *Eye-Robot* calibration procedure uses joint-encoder information along with the estimated RGB-D sensor pose (obtained from *SLAM*) to observe whole body motions. Calibration parameters are obtained by solving *Eye-Robot* least-squares optimization.

may be due to robot modifications and adaptations, passive mechanisms with no sensors, the addition of new sensors, or simply due to normal wear and tear. The accuracy and robustness of a robotic system is highly dependent on these calibration parameters and a solution to this problem is to develop life-long self-calibrating methods. In this thesis a generic solution is proposed to continuously estimate these parameters using only encoder information and Dense Visual *SLAM*, along with the observation of the contact configuration.

We exemplify our study with humanoid robots. They require accurate calibration to continuously interact with the environment. They are complex systems designed to be as versatile as humans in their interactions (*e.g.* locomotion in complex cluttered and uneven terrain) and manipulation (*e.g.* opening a valve, climbing a ladder, drilling). They are also modelled with a large set of parameters, including the kinematic tree topology, link lengths, joint angles, sensor locations, *etc*. Reliable planning and control of their actions need these parameters to be determined with the accuracy required by the tasks to be achieved.

In the literature, apart from few recent papers, kinematic parameters are assumed to be known and unchanging, and mechanical links to be accurately calibrated by the manufacturer. Our contribution is to calibrate the kinematic tree's geometric pose parameters online using an on-board RGB-D sensor (for 6dof localisation and mapping) and joint encoders (optical). The floating-base 6D configuration (translation and rotation) *w.r.t.* to SLAM's environment map is also determined.

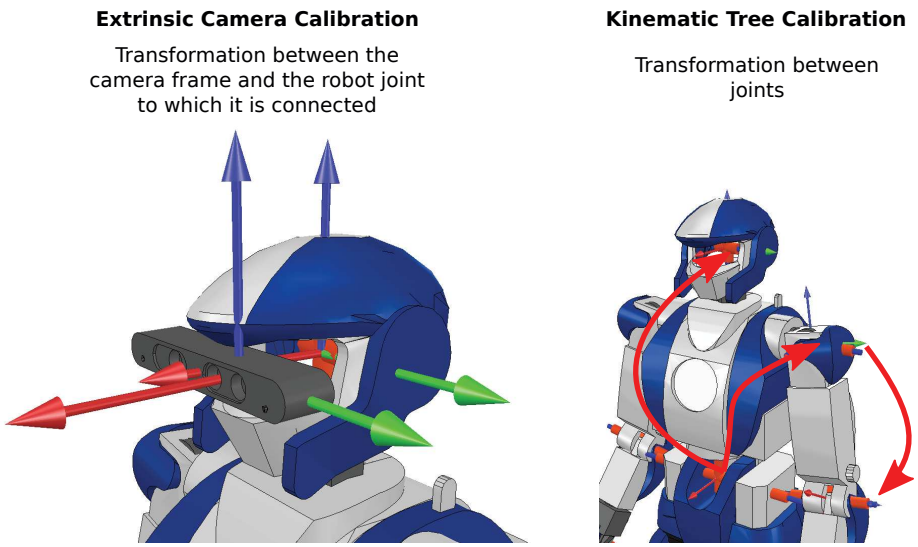


Fig. 2.3 Extrinsic camera calibration and kinematic tree calibration are generally considered separately. The *Eye-Robot* formulation solves both simulateneously.

2.2 Related work

There are several efficient methods for kinematic chain calibration offline. In this chapter, a particular focus is made on methods suitable for online self-calibration, with minimal requirements for both the robot and the environment.

In Roncone et al. [2014] a method based on self-touch was proposed: using sensory covers with thousands of tactile arrays, kinematic loops are formed by exploiting an existing imprecise forward kinematic model. Correspondences are established between predicted contact points and the sensed ones, which allows the refinement of kinematic parameters. While the idea is interesting, the sensory requirements are considerable.

Another common approach consists of forming a virtual closed loop between the camera and an end effector, by exploiting self-observation. In Hersch et al. [2008]; Martinez-Cantin et al. [2010], each end-effector's motion is observed by the robot's camera. As is the case with the sensory skin, this creates closed-loop constraints. Unfortunately this approach only corrects joint offsets and link lengths. These methods also suffer from two important drawbacks: observability issues, and the problem of robustly localizing the end-effectors. The later is often avoided by rigidly attaching calibration markers to the end-effectors during calibration [Kastner et al., 2015; Pradeep et al., 2014]. There are also additional uncertainties regarding the extrinsic pose of the calibration target *w.r.t.* the end effector. In Section 3.2.3 we describe one of our former pattern-free extrinsic RGB-D sensor calibration approaches that was used prior to this chapter's proposed calibration. It relies on finding the sensor pose that minimizes the ICP-based registration error between the robot CAD model and the self-observations of the robot links obtained from *D6DSLAM*. This somewhat alleviates the difficulties caused by pattern-based approaches, but remains both an inconvenient and imprecise method.

Most recently, a generic online self-calibration algorithm was presented in Maye et al. [2016]. It uses information theory to identify measurements that lead to improvements in calibration, and automatically detects and locks unobservable directions in parameter-space. This results in an online algorithm that listens to incoming sensor streams and builds a minimal set of data for estimating calibration parameters. The latter are continuously updated when they are observable; otherwise their initial guess is kept. This is truly a remarkable work, however, it does not provide any guarantee that the parameters will eventually become observable. In calibrating a humanoid robot, one needs to make sure that the robot motions are suitable to guarantee full-observability of all kinematic parameters.

In order to perform Eye-Robot calibration it is necessary to estimate the pose of the camera. The most prominent method for determining the camera extrinsic calibration parameters is the Hand-Eye method, first proposed by Tsai et Lenz [1989]. It commonly

relies on calibration patterns, either fixed to the robot or placed within the environment, a choice mainly driven by lack of suitable marker-less tracking systems. Recently, many state-of-the-art dense visual simultaneous localization and mapping approaches [Engel et al., 2015; Meilland et Comport, 2013a; Newcombe et al., 2011b; Whelan et al., 2012; Zhou et al., 2016] have been made available, and provide a robust pattern-free solution for tracking the camera pose (either RGB-D sensor, or a stereo pair), but also for reconstructing a 3D map of the environment, which will be of use in our approach. Any of these prominent SLAM solution are suitable for our proposed method. We opted for using the SLAM system of Meilland et Comport [2013a] which is robust and particularly developed for robotics applications.

We propose a novel online method to calibrate and estimate each of the aforementioned parameter sets by making full use of dense visual *SLAM*. Removing the need for calibration patterns and manual intervention we can achieve online correction of the calibration parameters at any time, including during normal robot operation. *Kinematic parameters* are determined by solving the proposed Eye-Robot calibration procedure which has been inspired by classic Hand-Eye calibration [Tsai et Lenz, 1989]. Similarly to Chen [1991] for hand-eye calibration, an analysis is performed on parameter observability to show that not all robot configurations are observable in a Eye-Robot setting. With full-observability, the kinematic parameters are obtained without need for any initial guess, which allows to fully calibrate unknown chains. When parameters are not fully observable, their calibration can still be improved along the observable directions, while holding the remaining parameters to their initial guess. Our method is validated in both simulation, and with real experiments performed on and HRP-4 humanoid robot.

2.3 Hand-Eye Calibration

Hand-Eye calibration was first described in Tsai et Lenz [1989], and has since become a standard tool for extrinsic calibration of vision sensors. It relates N relative sensor motions $\mathbf{A}_i \in \mathbb{SE}(3)$ with the corresponding motion $\mathbf{B}_i \in \mathbb{SE}(3)$ of a robot link to which it is rigidly attached to estimate the robot-sensor transformation $\mathbf{X} \in \mathbb{SE}(\mathbf{3})$. To simplify notations, the subscript i will often be omitted. Each motion forms the following kinematic equality

$$\mathbf{A}_i\mathbf{X} = \mathbf{X}\mathbf{B}_i \tag{2.1}$$

- $\mathbf{A}_i$ is typically obtained by tracking the camera *w.r.t.* calibration patterns [Heller et al., 2014; Tsai et Lenz, 1989].
- $\mathbf{B}_i$ is commonly obtained through forward kinematics, by assuming that the robot kinematic parameters are fully available.

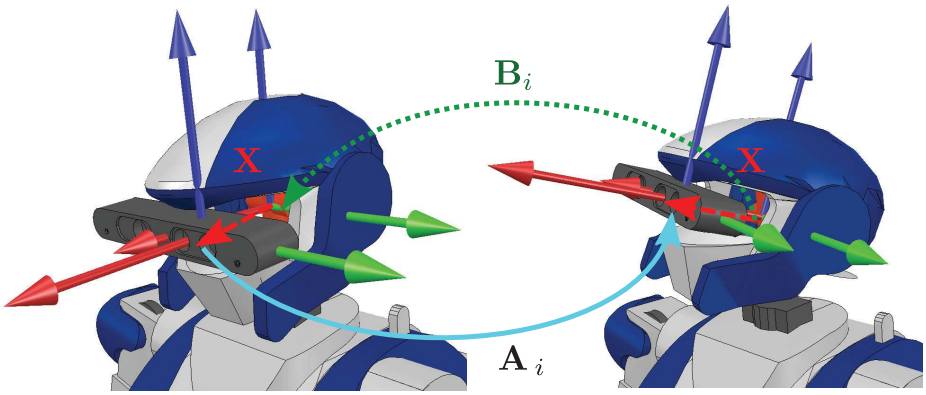


Fig. 2.4 Hand-Eye calibration of the rigid-body transformation $\mathbf{X}$ between the *Hand* frame (HRP4 head) and the *Eye* frame (RGB-D sensor optical frame). By forming a closed-loop kinematic chain between the relative motion of the *Eye* ($\mathbf{A}_i$) and the corresponding motion of the *Hand* ($\mathbf{B}_i$), the Hand-Eye method estimates the unknown transformation $\mathbf{X}$.

Reformulation as a lie-algebra least-square minimization

Several methods have been proposed to solve this equation. In Park et Martin [1994], the rotation and translation are determined separately with standard non-linear least-squares minimization. In Heller et al. [2014], several formulations are proposed leading to multivariate polynomial optimization problems, that are globally solved using convex linear matrix inequality relaxations. A geometric analysis for the uniqueness of a solution is provided in Chen [1991], which shows that translation and rotation should not be decoupled. Following these recommendations from the literature, the solution to Equation 2.1 is determined by solving the following non-linear cost function (Equation 2.2), with the unknown transformation $\mathbf{X} \in \mathbb{SE}(3)$ parametrized using its lie algebra representation $\mathbf{x} \in \mathfrak{se}(3)$ such that $\mathbf{X} = \text{expm}([\mathbf{x}]_\wedge)$, with $[.]_\wedge$ being the skew symmetric operator (see Equation 2.11). For simpler notations, we denote $e^{\mathbf{x}} = \text{expm}([\mathbf{x}]_\wedge)$.

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\text{argmin}} \sum_{i=1}^{N} ||\mathbf{A}_i e^{\mathbf{x}} \widehat{\mathbf{X}} - e^{\mathbf{x}} \widehat{\mathbf{X}} \mathbf{B}_i|| \tag{2.2}$$

The Lie algebra representation allows simultaneously estimating both rotation and translation using a minimal number of 6 parameters (3 for rotation, 3 for translation), and robustness can be improved by using as many measurements as necessary to iteratively solve an overdetermined system of equations.

In this work, we exploit the tracking capabilities of dense visual SLAM systems to advantageously replace the measurements needed for the computation of $\mathbf{A}$. Using real-time SLAM localization offers two main advantages over standard methods: the dense method used is more accurate than conventional methods, and it can also be used to acquire data online. The Hand-Eye calibration problem can thus be formulated as an online calibration method, by solving Equation 2.1 from available robot and tracking data.

Jacobian and solution

We acquire a set of N measurements. Each measurement i is a pair of corresponding hand and eye motions $(\mathbf{A}_i, \mathbf{B}_i)$. For each measurement, the error is defined as

$$\mathbf{e}_i = \mathbf{A}_i e^{\mathbf{x}} \widehat{\mathbf{X}} - e^{\mathbf{x}} \widehat{\mathbf{X}} \mathbf{B}_i \tag{2.3}$$

The Jacobian is defined as

$$\mathbf{J} = \left. \frac{\partial \mathbf{e}_i}{\partial \mathbf{x}} \right|_{\mathbf{x}=0} \tag{2.4}$$

The advantage of this formulation with the Jacobian evaluated at identity is that it can be computed analytically using the properties of the special-euclidean algebra. See Blanco [2010] for details on the derivation of such a Jacobian. Equation 2.5 provides a detailed computation of this Jacobian.

$$\begin{aligned}
\mathbf{J}_a &= \left. \frac{\partial \mathbf{A}_i e^{\mathbf{x}} \widehat{\mathbf{X}}}{\partial \mathbf{x}} \right|_{\mathbf{x}=0} = \left. \frac{\partial \mathbf{AB}}{\partial B} \right|_{\substack{\mathbf{A}=\mathbf{A}_i \\ \mathbf{B}=e^0}} \left. \frac{\partial e^{\mathbf{x}} \widehat{\mathbf{X}}}{\partial \mathbf{x}} \right|_{\mathbf{x}=0} \\
&= [\mathbf{I}_4 \otimes \mathbf{R}_{A_i}] \left. \frac{\partial e^{\mathbf{x}} \widehat{\mathbf{X}}}{\partial \mathbf{x}} \right|_{\mathbf{x}=0} = \begin{bmatrix} \mathbf{0}_{3\times3} & -\mathbf{R}_{A_i}[\hat{\mathbf{x}}_1]_\wedge \\ \mathbf{0}_{3\times3} & -\mathbf{R}_{A_i}[\hat{\mathbf{x}}_2]_\wedge \\ \mathbf{0}_{3\times3} & -\mathbf{R}_{A_i}[\hat{\mathbf{x}}_3]_\wedge \\ \mathbf{R}_{A_i} & -\mathbf{R}_{A_i}[\hat{\mathbf{x}}_4]_\wedge \end{bmatrix} \\
\mathbf{J}_b &= \left. \frac{\partial e^{\mathbf{x}} \widehat{\mathbf{X}} \mathbf{B}_i}{\partial \mathbf{x}} \right|_{\mathbf{x}=0} = \left. \frac{\partial \mathbf{AD}}{\partial A} \right|_{\substack{\mathbf{A}=\mathbf{I}=e^0 \\ \mathbf{D}=\widehat{\mathbf{X}}\mathbf{B}_i}} \left. \frac{\partial e^{\mathbf{x}}}{\partial \mathbf{x}} \right|_{\mathbf{x}=0} \\
&= [\mathbf{D}^T \otimes \mathbf{I}_3] \left. \frac{\partial e^{\mathbf{x}}}{\partial \mathbf{x}} \right|_{\mathbf{x}=0} = \begin{bmatrix} \mathbf{0}_{3\times3} & -[\mathbf{d}_1]_\wedge \\ \mathbf{0}_{3\times3} & -[\mathbf{d}_2]_\wedge \\ \mathbf{0}_{3\times3} & -[\mathbf{d}_3]_\wedge \\ \mathbf{I}_{3\times3} & -[\mathbf{d}_4]_\wedge \end{bmatrix} \\
\mathbf{J}_i &= \mathbf{J}_a - \mathbf{J}_b
\end{aligned} \tag{2.5}$$

where $\mathbf{R}_{A_i}$ is the rotation part of $\mathbf{A}_i$, $\hat{\mathbf{x}}_i$ is the i-th column of $\widehat{\mathbf{X}}$ and $\mathbf{d}_i$ is the i-th column of $\widehat{\mathbf{X}}\mathbf{B}$.
The full Jacobian is obtained by stacking each measurement's Jacobian as

$$\mathbf{J} = [\mathbf{J}_1 \dots \mathbf{J}_N]^T$$
$$\mathbf{e} = [\mathbf{e}_1 \dots \mathbf{e}_N]^T$$

Equation 2.2 is can be solved iteratively by Gauss–Newton (or alternatively Levenberg-Marquardt) minimization. Each iteration provides the update

$$\mathbf{x} = -\mathbf{J}^+\mathbf{e},$$

where $\mathbf{J}^+ = (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}$ is the pseudo-inverse of the stacked error Jacobian $\mathbf{J}$ and $\mathbf{e}$ is the stacked error. The solution is then iteratively refined until convergence as

$$\widehat{\mathbf{X}} = e^{\mathbf{x}}\widehat{\mathbf{X}}$$

The next section determines conditions on the robot motion that guarantee observability of the Hand-Eye calibration.

2.4 Eye-Robot Calibration of a Kinematic Chain

Our goal is to calibrate a robot's kinematic tree, provided that one has an estimate of the camera pose and joint state. In order to achieve this we show that the problem is related to that of hand-eye calibration, even if standard hand-eye calibration methods are not sufficient. In particular, they assume knowledge of the full 6DoF motion of a robot link, which presupposes a known kinematic model, which is exactly what we aim to determine in this work. First the classic approach is presented, then we develop our approach to Eye-Robot calibration that overcomes these limitations, while providing an online calibration procedure.

2.4.1 Eye-Joint Calibration

As mentioned earlier, the most common use of Hand-Eye calibration in robotics has been to perform the calibration of a vision sensor *w.r.t.* a robot link. In that case, both $\mathbf{A}$ and $\mathbf{B}$ commonly have 6 degrees of freedom each. $\mathbf{B}$ is obtained by assuming the kinematic tree to be known and well-calibrated and its motion is computed by forward kinematics. As the goal is to extend the method to the calibration of unknown kinematic chains, it cannot

be determined this way, as the kinematic model is not available for the computation of the motion $\mathbf{B}$.

Most – if not all – robots are fitted with joint encoders in addition to the vision sensor. Encoders provide information about the intrinsic motion of the joint, denoted by $\mathbf{S} \in \mathbb{SE}(3)$ (see Table 2.1 for common joint parametrization). Thus for a sensor attached to a single joint, one can reformulate the Hand-Eye calibration as

$$\mathbf{A}_i\mathbf{X} = \mathbf{X}\mathbf{S}_i \tag{2.6}$$

This can be seen as a special-case of Hand-Eye calibration, where the robot motion is that of a single joint such that $\mathbf{B}_i = \mathbf{S}_i$. As will be seen in Section 2.5.2, depending on the type of joint mechanism used, the motion might not lead to a fully-observable solution. For instance the solution to $\mathbf{X}$ for a revolute joint will not be observable around and along its rotation axis.

Revolute $R_x(\theta)$	**Revolute** $R_y(\theta)$	**Revolute** $R_z(\theta)$
$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) & 0 \\ 0 & sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Prismatic $P_x(\theta)$	**Prismatic** $P_y(\theta)$	**Prismatic** $P_z(\theta)$
$\begin{bmatrix} 1 & 0 & 0 & \theta \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & \theta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \theta \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Table 2.1 Representation of intrinsic joint motion for revolute (pure rotation) and prismatic joints (pure translation).

2.4.2 Eye-Robot Calibration : Online Kinematic-chain Calibration

In the previous sections, we described the classic Head-Eye calibration method, and extended it to online Eye-Joint calibration. We now propose to extend it further to the calibration of a whole kinematic chain inducing a camera motion. As we will see, this formulation can be seen as a recursive Eye-Joint calibration, where each intrinsic joint motion contributes to a part of the full-solution (See section 2.5 for a detailed observability analysis).

Figure 2.5 describes the parameters required for Eye-Robot calibration. Each joint is parametrized by a known intrinsic transformation $\mathbf{S_i} \in \mathbb{SE}(3)$, that depends on the type of joint used: rotation along the joint axis for a revolute joint, translation along axis for a prismatic joint, *etc*. The geometric transformations between joints are defined as the

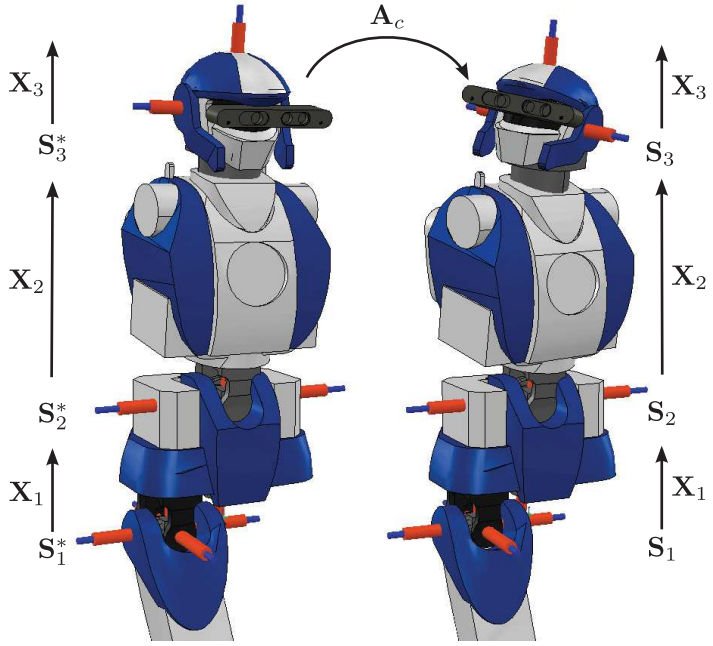


Fig. 2.5 A decomposed view of HRP-4 kinematic chain motion between 2 postures. $\mathbf{A}_c$ is the measured camera motion obtained from visual odometry, $\mathbf{S}_i$ represents each joint's intrinsic motion (the definition of $\mathbf{S}$ depends on the type of joint used). $\mathbf{X}_i$ is the rigid geometric transformation between the successive joints and is defined *w.r.t.* a reference robot posture. Notice that, as is the case for Hand-Eye geometry, a closed-loop is formed, which allows for the estimation of the parameters, even though the camera motion is not necessarily defined *w.r.t.* the same reference frame as the robot motion.

unknown parameters $\{\mathbf{X}_i \,/\, \mathbf{X}_i \in \mathbb{SE}(3)\}_{i=1..N}$ (see Figure 2.5), that we wish to estimate. The minimal unknown parameter vector is defined using their corresponding Lie-algebra twist representation $\mathbf{x} = \{\mathbf{x}_i \,/\, \mathbf{x}_i \in \mathfrak{se}(3)\}_{i=1..N}$. In order to solve for the unknown parameters $\mathbf{x}$, the state of the robot in several postures needs to be acquired. We exploit the same argument as for Hand-Eye calibration: the geometric transformations between joints ($\mathbf{X}_i$), excluding intrinsic joint motion from the transformations, are rigid, thus constant. However, each joint's intrinsic state varies. The standard Hand-Eye equation is extended by expressing the camera motion through all kinematic chain parameters *w.r.t.* to an initial reference posture (Figure 2.5) as follows:

$$\mathbf{A}_c\mathbf{X}_N\mathbf{S}_N \ldots \mathbf{X}_1\mathbf{S}_1 = \mathbf{X}_N\mathbf{S}_N^* \ldots \mathbf{X}_i\mathbf{S}_1^*, \tag{2.7}$$

where

- $\mathscr{S}^* = \{\mathbf{S}_N^*, \ldots, \mathbf{S}_1^*\}$ is the reference intrinsic joint state taken as the first measurement.

- $\mathscr{S} = \{\mathbf{S}_N, \ldots, \mathbf{S}_1\}$ is the current intrinsic joint state.
- $\mathbf{X}_i\mathbf{S}_i$ is the intrinsic motion of joint i, followed by the geometric transformation of link i.
- $\mathbf{A}_c$ is the observed camera motion corresponding to the movement induced by the joints $\mathscr{S}$.

This formulation decouples the geometric calibration parameters from the intrinsic joint state, and ensures that the solution is always provided at the identity ($\mathscr{S} = \{\mathbf{I}_N, \ldots, \mathbf{I}_1\}$). This makes it possible to start the calibration procedure online from any robot posture and obtain results independently of the initial robot posture.

Equation 2.7 is solved by non-linear Levenberg-Marquardt optimization stacking the errors obtained from each individual observed robot posture. However, as the Eye-Joint formulation does not always lead to a full solution, so does the Eye-Robot formulation.

The next section will describe in details conditions for observability, and in case of a partial solution, which degrees of freedom are fully determined.

2.5 Calibration parameter observability

2.5.1 Hand-Eye Observability

Observability of Hand-Eye calibration has been studied in detail in Chen [1991] using the screw motion representation. This representation directly stems from Mozzi-Chasles' theorem [Chasles, 1831] which states that the most general rigid body displacement can be produced by a translation along a line, called its screw axis and a rotation about that line. As such, any twist $\boldsymbol{\xi} = (\boldsymbol{\upsilon}, \boldsymbol{\omega}) \in \mathfrak{se}(3)$ can be expressed equivalently in screw-representation as follows [Murray et al., 1994, p47].

Screw Motion 2.5.1 *A screw S consists of an axis l, a pitch h, and a magnitude θ. A screw motion represents rotation by an amount θ about the axis l followed by translation by an amount $d = h\theta$ parallel to the axis l. If $h = \infty$ then the corresponding screw motion consists of a pure translation along the axis of the screw by a distance θ.*

The screw coordinates can be directly determined from the twist $\boldsymbol{\xi}$. The axis of rotation is given by the line

$$\mathbf{l} = \begin{cases} \frac{\boldsymbol{\omega}\times\boldsymbol{\upsilon}}{\|\boldsymbol{\omega}\|^2} + \lambda\boldsymbol{\omega} : \lambda \in \mathbb{R}, & \omega \neq 0 \\ 0 + \lambda\upsilon : \lambda \in \mathbb{R}, & \omega = 0 \end{cases}, \tag{2.8}$$

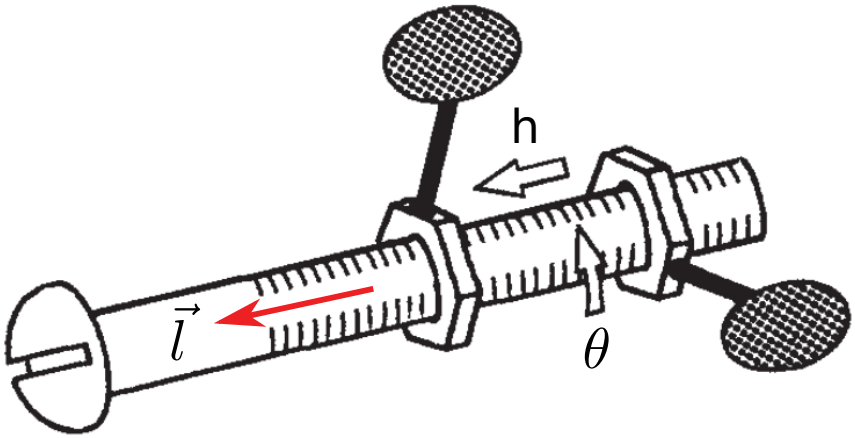


Fig. 2.6 The displacement of a rigid object from one position to another can be described as a screw motion [Chen, 1991] : a displacement of a distance $h\theta$ and angle θ along the screw axis $\vec{L}$

where the axis l is a directed line through a point. For $\omega \neq 0$, the axis is a line in the ω direction going through the point $\frac{\boldsymbol{\omega}\times\boldsymbol{\upsilon}}{||\boldsymbol{\omega}||^2}$. For $\omega = 0$, the axis is a line in the υ direction going through the origin.
The pitch angle is

$$h = \frac{\boldsymbol{\omega}^\top \boldsymbol{\upsilon}}{||\boldsymbol{\omega}||} \tag{2.9}$$

The pitch of a twist is the ratio of translational motion to rotational motion. If $\boldsymbol{\omega} = 0$, we say that $\boldsymbol{\xi}$ has infinite pitch.
The rotation angle is:

$$\theta = \begin{cases} ||\boldsymbol{\omega}||, & \boldsymbol{\omega} \neq 0 \\ ||\boldsymbol{\upsilon}||, & \boldsymbol{\omega} = 0 \end{cases} \tag{2.10}$$

The observability of the pose **X**, based on the observed movements **A** observed in frame F_A and **B** in frame F_B, depends on the screw congruence theorem (Theorem 2.5.2).

Screw Congruence Theorem 2.5.2 (Chen) *Screw Congruence Theorem: Let* $\mathbf{X} = (\mathbf{R}_x, \mathbf{t}_x)$ *be the transformation that takes a Cartesian frame* F_A *into exact alignment with another Cartesian frame* F_B*. Also let* (d_A, θ_A, L_A) *and* (d_B, θ_B, L_B) *be two screw motion descriptions of an object obtained in* F_A *and* F_B*, respectively. L represents the screw axis (defined by a point c and a direction n),* θ *is the rotation angle along that axis, and d is the translation*

along the screw axis. Then

$$\begin{aligned} d_A &= d_B \\ \theta_A &= \theta_B \\ n_A &= R_X n_B \\ c_A &= R_X c_B + t_x - (n_A^T t_x) n_A \end{aligned}$$

, where c_A and c_B are the position vectors of L_A and L_B, and n_A and n_B are the direction vectors of L_A and L_B, respectively.

This theorem states that both **A** and **B** share the same screw transformation since they undergo the same movement, *e.g.* that the screw motion is independent of the frame in which it is observed. This is easy to visualize with a simple example. Consider what happens if you screw a screw into a wall. No matter where you observe the motion from, the screw will have undergone the same rotation around its own axis and the same translation along its axis.

Here Chen's work is reformulated using a Lie Group representation. Consider the tangent spaces of the matrices **A** and **B** such that $[\boldsymbol{\alpha}]_\wedge = \text{logm}(\mathbf{A}) \in \mathfrak{se}(3)$ and $[\boldsymbol{\beta}]_\wedge = \text{logm}(\mathbf{B}) \in \mathfrak{se}(3)$. The tangent matrices $[\boldsymbol{\alpha}]_\wedge$ and $[\boldsymbol{\beta}]_\wedge$ are composed of both angular and linear components as follows:

$$[.]_\wedge = \begin{bmatrix} [\boldsymbol{\omega}]_\times & \boldsymbol{\upsilon} \\ 0 & 0 \end{bmatrix} \tag{2.11}$$

The linear component can be decomposed by projection into a component parallel to the rotation axis $\boldsymbol{\upsilon}^{||} = (\boldsymbol{\upsilon}^\top \mathbf{u})\mathbf{u}$ and a component perpendicular $\boldsymbol{\upsilon}^\perp = \boldsymbol{\upsilon} - \boldsymbol{\upsilon}^{||}$. Following the screw congruence theory [Chen, 1991], it can be shown that the rotation around the screw axis and the translation along the axis are invariant to coordinate frames such that $\mathbf{t}_\alpha^{||} = \mathbf{t}_\beta^{||}$ and $\theta_\alpha = \theta_\beta$.

Using this theorem, Chen has shown that for well-defined screws

- The head-eye geometry can be uniquely determined from two robot motions if and only if the two screw axes are skew or intersecting lines
- Necessary and sufficient condition of a uniqueness solution for the head-eye geometry is that the screw axes of two robot motions are either skew or intersecting lines.

These conditions are easily achieved if the robot motion can perform a full 6D-motion (translations and rotations simultaneously). In other cases, a partial solution may still exist, which will be exploited to formulate our Eye-Robot calibration.

2.5.2 Eye-Joint Observability

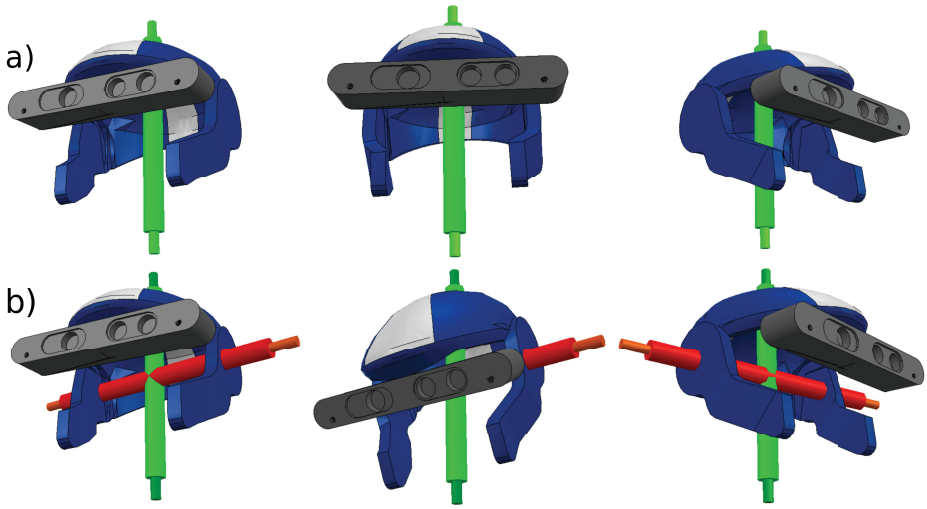


Fig. 2.7 Observability of Eye-Joint motions. $1DoF$ revolute joint (a) only provide observable rotation and translation around their rotation axis, while $2DoF$ revolute joints provide observable rotations and translations around two axes.

The observability of Eye-Joint formulation can be seen as a special case of Hand-Eye observability, where the robot motion $\mathbf{B}_i = \mathbf{S}_i$ only represents the intrinsic joint motion, which is in general not a full 6D transformation, and does not induce a full 6D motion of the eye frame $\mathbf{A}_i$.

Ball-Joint It is clear that in the case of *ball-joint*, the solution $\mathbf{X}$ is fully determined. Indeed in order to obtain full observability, the screw axes L_A and L_B mustn't be parallel, which is possible in the case of a ball joint.

Revolute Joint The motion of a revolute joint is a $1DoF$ rotation around its axis. In that case, $\mathbf{t}^{||} = 0$ and $\theta = \widehat{\theta}$ is measured by the joint encoder. The relationship between the screw axis and its translation are, however, dependent on $\mathbf{X}$. Subsequently it can be seen that the two invariant screw parameters do not provide any constraints for estimating the unknown calibration pose along and around the joint's rotation axis. The remaining 4 degrees of freedom are however fully determined.

From this analysis, we can now obtain a partial solution to the Eye-Joint calibration of a single revolute joint. Consider a joint rotating around its x-axis. Assuming a known initial guess $\mathbf{x}_x^* = [\upsilon_x^*, 0, 0, \omega_x^*, 0, 0]^T$ along the non-observable axis x, the remaining degrees of freedom (translation and rotation along y and z axes) can be determined by solving 2.2 for $\mathbf{x}_{yz} = [\upsilon_y, \upsilon_z, \omega_y, \omega_z]^T$.

$$\hat{\mathbf{x}} = \underset{\mathbf{x}_{yz}}{\operatorname{argmin}} \sum_{i=1}^{N} ||(\mathbf{A_i}\text{expm}([\widetilde{\mathbf{x}}]_\wedge) - \text{expm}([\widetilde{\mathbf{x}}]_\wedge)\mathbf{B_i}|| \tag{2.12}$$

, where $\widetilde{\mathbf{x}} = [\upsilon_x^*, \upsilon_y, \upsilon_z, \omega_x^*, \omega_y, \omega_z]^T$. That is, only the observable DoF are obtained, and the others are kept to their initial value, such as the known manufacturer's calibration if available. This makes it possible to improve calibration of the observable degrees of freedom online while keeping a known calibration (*e.g.* manufacturer's) for the others. A similar method can be applied to all other joint types.

2.5.3 Eye-Robot Observability

The observability of kinematic chain calibration is obtained by recursively applying Eye-Joint observability results, starting from the first joint attached to the sensor and moving down the chain.

In the case of HRP-4 (Figure 2.5), most links are attached to a pair of revolute joints sharing the same rotation center. In that case the solution is fully determined. The RGB-D sensor is attached to the neck pitch and yaw revolute joints. Thus the sensor pose *w.r.t.* to these two joints may fully be determined. Once determined motion of the chest yaw and pitch joint provides additional constraints and the chest-to-sensor transformation becomes fully known as well. In case one of the joint configuration on the robot does not admit a full solution, then the partial-solution strategy presented in Section 2.5.2 can be used to provide an initial guess allowing to ignore the unobservable degrees of freedom in the optimization.

2.6 Results

Our method is applied and evaluated on several practical use-cases that were faced in practice during this thesis, and that provided motivation to this chapter.
The Hand-Eye method presented in Section 2.3 is used to calibrate

- HRP4's RGB-D sensor
- MOCAP markers placed on HRP4 body
- MOCAP markers placed on a hand held object for which ground-truth transformation is known, allowing for evaluation of the results.

The Eye-Robot method presented in Section 2.4 is demonstrated on

- As an alternative method to HRP4's RGB-D sensor calibration
- To simultaneously calibrate HRP4's torso, head, and RGB-D sensor

First we present our efficient C++ implementation that performs real-time Hand-Eye and Eye-Robot calibration. We then detail results on the aforementioned use-cases.

2.6.1 Implementation

We release our open-source C++ implementation[1] of online eye-robot calibration. The implementation relies on the open-source C++ optimization framework Roboptim[2] [Moulard et al., 2013], which enables straightforward definition of optimization problems, and is compatible with a wide range of solvers. Both Hand-Eye (Equation 2.1) and Eye-Robot (Equation 2.2) problems are solved using the Levenberg-Marquardt solver of the Eigen library.

Hand-Eye

The implementation of Hand-Eye minimization is straightforward. The cost function of Equation 2.3 and it's Jacobian from Equation 2.5 are implemented. The problem is then solved using a Levengerg-Marquardt implementation.

Eye-Robot

As discussed in Section 2.5, eye-robot calibration does not always admit a full solution. As such, special treatment is required to ensure that either a solution can be obtained, or that an initial guess is provided for the degrees of freedom that cannot be estimated.

For revolute joints with one degrees of freedom, translation and rotation along its rotation axis cannot be observed. For such joints, an initial guess is fixed along these directions, and only the 4 remaining degrees of freedom are considered in the optimization vector.

For revolute joints with more than one degree of freedom, a solution can be obtained. Amongst those are ball-joint, rarely found on a humanoid robot, for which a solution is directly observable. Another more common case is that of joints sharing the same rotation center, but a different rotation axis. This is for instance the case of HRP-4 neck and torso, both having a yaw and pitch rotation. In that case, a solution can be obtained by considering both joints as one. Let S_y denote the intrinsic yaw rotation and S_p denote the pitch rotation.

[1] https://github.com/arntanguy/robcalib
[2] http://roboptim.net

The full joint motion can be simply expressed as $S = S_y S_p$, whose motion fully constrain the eye-joint calibration.

Whether a joint configuration will constrain the eye-robot calibration can be known a-priori from the robot's kinematic model, and calibration on observable directions can be obtained.

2.6.2 Acquiring Calibration Data

Hand-Eye calibration relies on the availability of a set of N pose pairs ($\mathbf{A}_i$, $\mathbf{B}_i$) that each represent the same motion undergone by both the eye-frame and hand-frame. To ensure robust results, it is important to consider

- That both $\mathbf{A}_i$ and $\mathbf{B}_i$ were acquired at the same time, and represent the same rigid body motion
- That the set of all measurements best constrains the minimization
- That each measurement is as noise-free as possible to ensure best accuracy

In this section we expose practical strategies to obtain a suitable set of measurements that ensure good calibration results.

Ideal sampling frequency

Hand-Eye calibration aims at finding the rigid-body transformation between the hand and eye-frame, with both frames being commonly measured by two unrelated estimators, usually running unsynchronized and at different frequencies. To obtain robust calibration results, it is paramount that the timing errors are kept to a minimum. The estimators used and timing data available is highly dependent on the calibration use-case. For instance

1. For calibrating the transformation between MOCAP markers and a robot, one will use the pose tracked by the MOCAP system obtained at high frequency (commonly more than $100Hz$), and that of the robot link, commonly obtained by forward kinematics at higher frequency ($200Hz$ in case of HRP-4).
2. For calibrating the transformation between MOCAP markers and an RGB-D sensor, one will use the MOCAP tracking, and visual odometry such as SLAM running at lower frequency (typically $30Hz$)

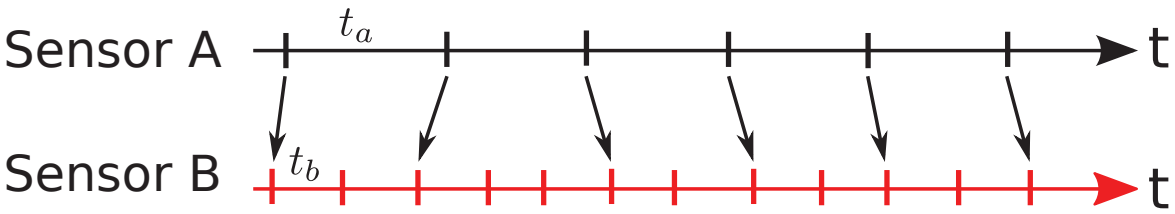


Fig. 2.8 Illustration of sampling from two sensors A and B running at different varying frequencies $f_a = \frac{1}{t_a+\Delta_a}$ and $f_b = \frac{1}{t_b+\Delta_b}$. Each vertical tick represents a sensor data. Δ_a and Δ_b are used to show that the sensor acquisition time is not necessarely constant and might vary over time. For *RGBD-to-MOCAP* Hand-Eye calibration, Sensor A is the tracked RGB-D pose from SLAM with $f_a = 30Hz$, and Sensor B is the tracked pose from MOCAP with $f_b = 100Hz$. The arrows show the closest possible sampling times for each sensor.

Due to the widely different frequencies and sources of tracking, it is generally impossible to guarantee that the data will be acquired at the same time. In an ideal case, one would obtain stamped sensor data synchronized to a common clock time, which would allow to accurately find the closest pose estimates in time. As shown in Figure 2.8, hand and eye data is generally not perfectly synchronized due to the difference in the aquisition frequencies. Obtaining fully synchronized data would in that context require an estimator able to predict the future pose of lower-frequency estimators, or retain a subset of synchronized data.

Even in the ideal case where sensor timing is fully known, the level of synchonization depicted in Figure 2.8 does not prove sufficient to acquire data at the frequency of the slowest sensor. Consider a sensor acquired at a sampling rate $f_a = 30Hz$ and another one at $f_b = 100Hz$. In the worst case, the difference in acquisition time for each absolute pose is $\Delta_a = \frac{1}{2f_b} = 5ms$. For a realtive pose, this brings the maximum to $\Delta_r = 2\Delta_a = 10ms$, a maximum of up to 30% of the sampling frequency!

As will be shown in the next section, sampling data at such a high-frequency is not necessarily the best choice, as the relative motion is quite small and estimation inaccuracies will also play a large role. To finish with sampling frequency consideration, one can guarantee the maximum synchronization delay by choosing an the sampling frequency f_s such that the $f_s\Delta_r < \overline{\Delta_r}$, where $\overline{\Delta}_r$ is the maximum error expressed as a percentage of sampling time. For instance, if one wants to guarantee less than 1% of sampling-time error with these example frequencies, a sampling rate of $f_s < \overline{\Delta}_r f_b = 1Hz$ needs to be selected, which gurantees a maximal time error of $\frac{\overline{\Delta}_r}{f_s} = 10ms$ on data sampled every $1000ms$.

Constraining the Hand-Eye Minimization

We have shown how to best choose sampling frequency in the case of hand and eye-frame estimations obtained at different rates. Section 2.5.1 demonstrated minimal and sufficient

conditions on the input data to guarantee that a valid unique solution is obtained. However, it left out practical considerations on how to make sure that this result is the most accurate.

Systems such as MOCAP and SLAM both provide an absolute pose *w.r.t.* to a global reference frame at high frequency ($30-200Hz$). From these absolute poses, one needs to obtain a set of relative poses that is both best suited for constraining the Hand-Eye minimzation, and that minimizes the amount of noise and outliers. It has been shown by Tsai et Lenz [1989] that it is best to maximize the rotational motion between measurments by selecting pairs of poses that are as far as possible in that respect (see Figure 1.11).

Precision and Robustness using SLAM Keyframe-Graph

The afformentioned method to obtain good measurements for calibration provide a good general-purpose way of calibrating any sensor. However, in the case of SLAM, we can further improve on the robustness of the results by exploiting the keyframe-graph (see Figure 1.11). Besides combining all of the afformentioned recommendations (a lower sampling rate and maximizing the distance and rotation between the poses), the keyframe-graph offers one considerable advantage: it is continously refined even after the keyframe has been created. This is done by exploiting loop-closures (*i.e.* recognizing that we are near previously explored keyframes) to perfom a bundle-adjustement step of all poses in the graph. As such, the keyframe poses are much more precise and reliable than those obtained during tracking.

Relative poses may be simply obtained by computing the relative transformation between each keyframe and the all other keyframes in the graph. This provides the largest amount of constraints with N^2 transformations, N being the number of keyframes. A sequence of 30 seconds of motion will easily provide more than 20 keyframes, which provides more than 400 robust poses to constrain the minimization.

2.6.3 MOCAP-Xtion Calibration

To evaluate the dense Hand-Eye formulation, we have created a simple set-up where both the Xtion and reflective VICON markers are fixed on a rigid metal plate, providing groundtruth for the pose to be calibrated. These frames where placed manually as accurately as possible, but some uncertainty on the order of a couple of degrees remains about the orientation around roll and pitch axes.

Several trials are performed where motion in front of a staircase are performed for a short duration varying between 1 and 2 minutes. The Dense Hand-Eye method described in Section 2.3 is applied online to compute the extrinsic pose parameters between the MOCAP markers frame and the Asus Xtion optical frame. The Asus Xtion is tracked with PX2M

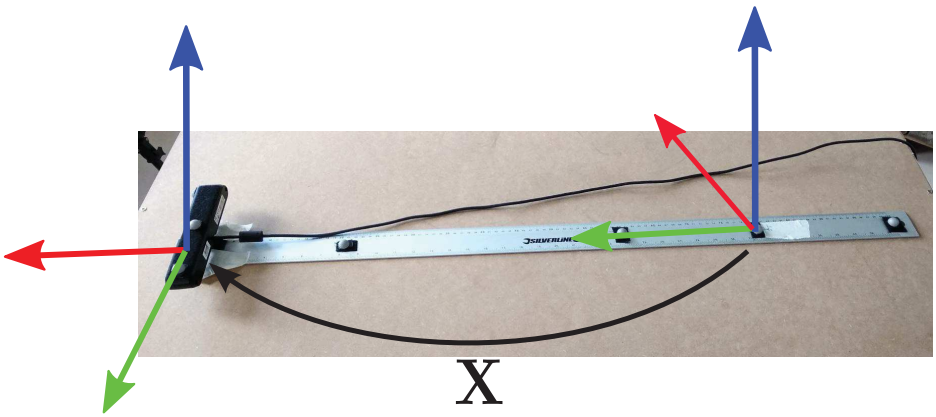


Fig. 2.9 Setup for groundtruth evaluation of dense Hand-Eye calibration. The Xtion and VICON markers have been disposed in a known configuration to provide groundtruth for **X**.

SLAM, and the MOCAP markers are tracked with a VICON system. Figure 2.11 and Table 2.2 report calibration results obtained from one of these calibration using various pose sampling strategies. All results were obtained from the same recorded sequence.

Fig. 2.10 Images from the calibration sequence, camera-ruler calibration is moved by hand around the staircase, camera motion is tracked with SLAM while the ruler motion is tracked with the VICON MOCAP system to provide input data for Hand-Eye calibration

Keyframe result referers to the sampling strategy described in 2.6.2. $N = 34$ keyframes were generated by PX2M for this sequence, and the maximum amount of relative poses ($N^2 = 1156$) was generated from it. This is the only method for which reliable results could be obtained for each calibration. Furthermore results were consistent across all calibration trials. Rough convergence was achieved with as little as $N = 10$ keyframes, and the most accurate stable convergence is achieved with more than $N = 30$ keyframes.

Relative results were obtained by sampling each pose at PX2M's frequency ($30Hz$). Convergence results is highly variable depending on the quality of tracking in each sequence.

Method	Translation (m)	Rotation (deg)	Translation Error (m)	Rotation Error (deg)
Keyframe	$\begin{bmatrix} -0.0454 \\ 0.8022 \\ 0.0309 \end{bmatrix}$	$\begin{bmatrix} 89.5959 \\ -6.4655^* \\ 3.0252^* \end{bmatrix}$	0.0124	7.1497
Relative	$\begin{bmatrix} -0.0407 \\ 0.5999 \\ 0.0115 \end{bmatrix}$	$\begin{bmatrix} 87.8761 \\ -7.0858^* \\ 1.9854^* \end{bmatrix}$	0.2002	7.6591
Expected	$\begin{bmatrix} -0.0400 \\ 0.800 \\ 0.0200 \end{bmatrix}$	$\begin{bmatrix} 90.0000 \\ 0.^* \\ 0.^* \end{bmatrix}$	0	0^*

Table 2.2 Hand-Eye Calibration results for various pose-sampling strategies. Note that groundtruth values marked with a $*$ are imprecise.

The poor convergence results are due to noise and outliers in the pose tracking. Additionnaly, PX2M loop-closure mechanism results in sharp jumps in the estimated pose, which are highly detrimental to the calibration dataset.

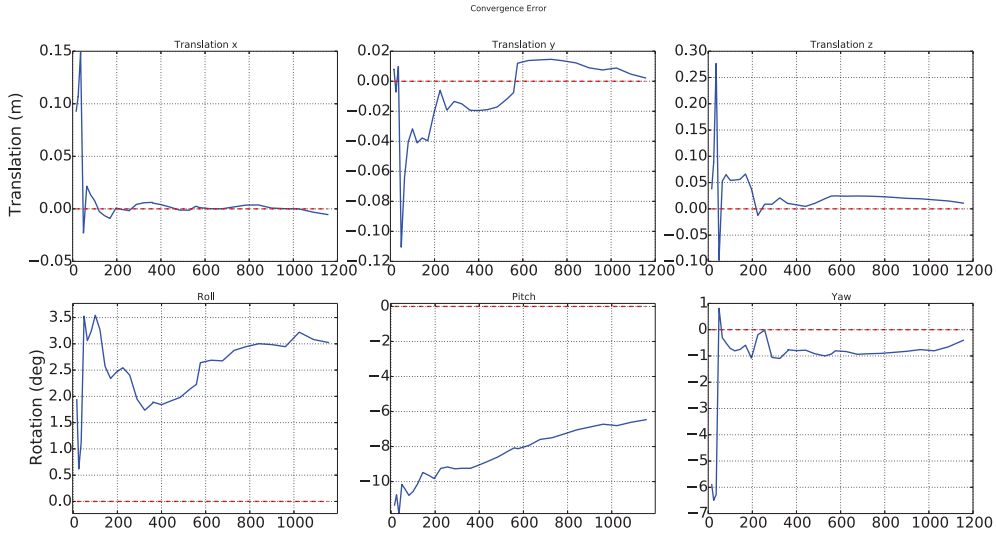


Fig. 2.11 Convergence of Hand-Eye calibration on the MOCAP-Xtion calibration setup. Imprecisions in convergence in roll and pitch rotations are mainly due to inaccuracies in the groundtruth measurements for the calibration object. Best convergence is reached for $N = 34$ keyframes corresponding to $N^2 = 1156$ relative pose measurements.

2.6.4 Head-Eye Calibration of HRP-4

To enable control strategies of HRP-4 *w.r.t.* to its environment knowing the precise pose of the vision sensors on the robot is of paramount importance. As illustrated by Figure 2.1,

unexpected events during robot control can affect the position of the sensor *w.r.t.* the robot. Here, during stair climbing trials, the robot fell on its supporting rope, which damaged the right side of the head, and in particular the screw-mechanism holding the Asus Xtion sensor on the robot. This unexpected event modified the extrinsic robot-to-sensor calibration, and a pattern-free method can recalibrate the sensor online without requiring to stop operations. In practice, the calibration of HRP-4 was performed by generating both translational motions of its CoM in all directions within its double support polygon for stability, and rotations of the RGB-D sensor. To avoid an issue of *SLAM* with pure rotations (see 3.2.3) in poorly textured environment, such motion is purposefully avoided, and the waist joints are used to generate non-pure rotations. However, in sufficiently contrainting environments, it is best to add head movements as well, as they will better constrain the hand-eye solution.

2.6.5 Eye-Robot calibration of HRP-4

First, using real data with HRP-4, we show that using the Eye-Joint method presented in Section 2.5.2 the extrinsic camera calibration can be obtained. We then illustrate the use of online Eye-Robot calibration in its full observability case by exploiting the ball-joint equivalence of HRP-4 torso and head joints. We show that the calibrated kinematic chain is consistent with our observability analysis in Section 2.5. All real experiments are performed using the SLAM of Meilland et al. [2013b], and the HRP-4 humanoid robot, controlled with the quadratic programming controller [Vaillant et al., 2016].

The HRP-4 humanoid robot has many degrees of freedom, an is fitted with very precise high-end optical encoders, and an Asus Xtion RGB-D sensor, making it a perfect candidate robot to test our proposed calibration techniques. All of HRP-4 joints are revolute, and many of its joints can be assimilated to fully-observable ball-joint. The head rotates around yaw and pitch axes, so does the torso, as well as the leg waist joints. Other joints, such as the knee have only one degree of freedom.

Experiments

In a first experiment performed on the HRP-4 robot, we illustrate the equivalence between the classic Eye-Hand calibration, and the proposed Eye-Joint calibration to determine the camera-to-robot extrinsic calibration parameters. The initial camera-to-robot transformation is randomly chosen, and a motion is performed using both neck joints simultaneously. The two neck joints are considered as a single ball-joint. As predicted by the screw-congruence theory, the algorithm starts to converge with only two relative measurements, but greater accuracy is obtained over time. The complementary video further illustrate the online aspect

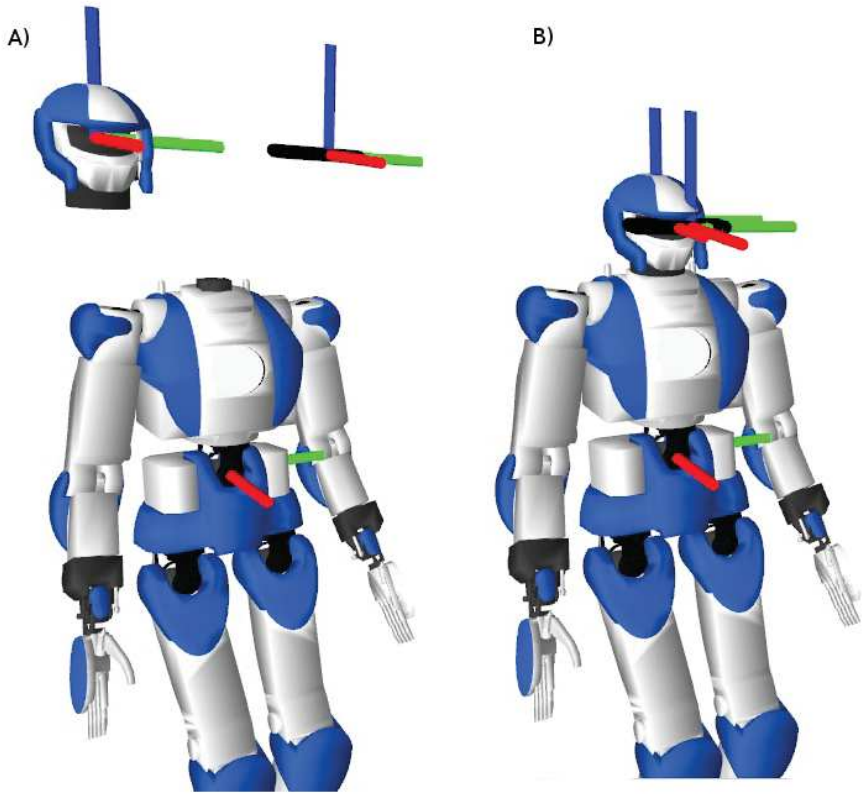


Fig. 2.12 Eye-Robot calibration of HRP-4 waist-to-camera transformations. A) The transformation between the torso joint and the head joint, and that between the head joint and the camera are initially unknown. B) Results of the eye-joint calibration procedure obtained from a simulated robot motion, where both the torso and head joints are moved simultaneously. The camera pose along with encoder values are obtained from the simulated kinematic tree. Calibration results are continuously improved online as new data becomes available.

of the calibration by estimating the location of the robot *w.r.t.* SLAM's map [Tanguy et al., 2016] using the live calibration results. As can be seen in the video and in Figure 2.13 the erroneous initial calibration results in the environment being misplaced *w.r.t.* the robot, while the end-result of the calibration correctly locates the robot.

In a second experiment, we show the Eye-Robot calibration method applied to the whole kinematic chain between the waist and the camera. In this configuration, the full geometric transformation between the torso and head joints, and head joint to the RGB-D sensor can be determined. Figure 2.12 shows the initial joint configuration along with the calibration results obtained after a few seconds of simulated robot motion. Both the torso and head joints are exited along their two degrees of freedom (rotation around the pitch and yaw axes) to guarantee full observability. Camera pose measurements are determined by forward kinematics of the ground-truth kinematic chain.

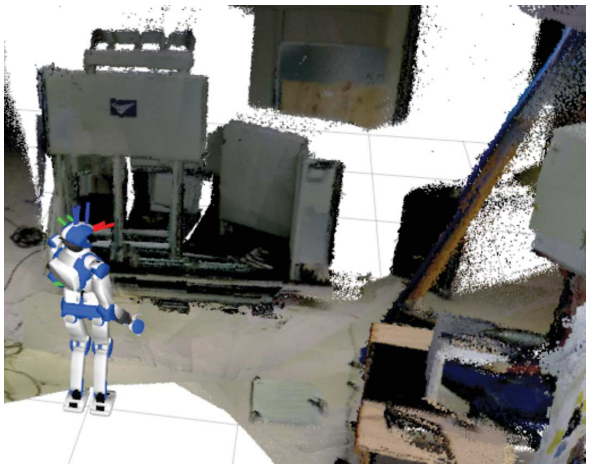


Fig. 2.13 Online Eye-Robot calibration of the kinematic tree geometric parameters for HRP-4's torso, head joints and camera (Hand-Eye). The floating base 6D (orienation and translation) is additionally calibrated *w.r.t.* the environment's point cloud.

The same experiment is then performed on a real HRP-4 humanoid robot. Here, camera motion is obtained from dense visual SLAM [Meilland et Comport, 2013a], and joint angles from high-resolution optical encoders. As can be observed in Figure 2.13, aside from obtaining convincing calibration for the considered joints, the pose of the robot's floating base *w.r.t.* to SLAM's real-time map can also be determined. This is achieved by exploiting the known pose of the RGB-D sensor within SLAM's map, along with the kinematic chain being refined. Once convergence is reached, the robot attitude is determined. The associated video further illustrate the online aspect of the calibration by estimating the location of the robot *w.r.t.* SLAM's map [Tanguy et al., 2016] using the live calibration results. As can be seen in the video and in Figure 2.13 the erroneous initial calibration results in the environment being misplaced *w.r.t.* the robot, while the end-result of the calibration correctly locates the robot.

2.6.6 Discussion

Calibration feasibility depends on the type of joints and their configuration. In particular, ball-joints parameters are fully observable. This is the case for many robots, in particular humanoids for which a common design is to have head, torso, and hips in a configuration akin to ball-joints. In this case, the proposed method is a direct generalization of the Hand-Eye calibration method to Eye-Joint calibration. For joints with only one degree of freedom, eye-joint calibration never admits a fully observable solution. In particular, for revolute joints, an ambiguity remains in both the translation and rotation along the joint's rotation

axis. Such joint configuration is commonly found and we propose two main solutions for their calibration. The first focuses on calibrating the 4 observable DoF, which is achieved by locking the unobservable parameters (along the joint's axis) to an initial reference value, assumed to be available (Section 2.5.2). Obtaining an accurate estimate of this initial value is however non-trivial, and one would most-likely fall back to using the manufacturer's calibration. For humanoid robots, the compromise of using the online calibration approach for the observable parameters, while using a known reference value for the others (manufacturer data, external calibration...) is best suited.

2.7 Conclusion and Future Work

Starting from Hand-Eye calibration results, we propose a novel method that performs online whole-body calibration of a kinematic chain using only joint encoders, and an RGB-D sensor. Contrary to Hand-Eye methods, no a-priori calibration of the kinematic chain is assumed available. It is necessary to replace the use of end-effector robot motion by that of the joints intrinsic motion, which leads to the study of Eye-Joint geometry, where the robot motion is entirely determined by its intrinsic joint motion. Hand-Eye observability results are applied to this restricted problem, and we demonstrate that full observability of the geometric parameters is achieved for ball-joints, or equivalently two or more revolute joints sharing the same rotation center. We further consider solving Eye-Joint calibration for common joint types for which full observability cannot be achieved. In this case, screw theory is used to determine and calibrate the observable directions, while keeping an initial guess for the non-observable directions. Building upon these results, we then reformulate Hand-Eye calibration to a full kinematic chain calibration, denoted as Eye-Robot calibration, and successfully apply the method with both simulated and real experiments to the calibration of the upper-body of an HRP-4 humanoid robot. Future work will focus on quantitative evaluation of the calibration results and of the influence of uncertainties in sensor measurements.

Chapter 3

Multi-contact Planning and Control in a Real Environment

In the previous chapter, we have seen how the kinematic model of the robot can be obtained, including the transformation between vision sensors and the robot body. This is a prerequisite to localizing any of the robot surfaces within the *SLAM* map. In this chapter, the problem of exploiting the robot localization *w.r.t.* to its environment will be considered.

Multi-contact control and interaction is a key behaviour for humanoids because it allows to: (i) create closed kinematics chains to drive higher forces; (ii) plan for stable postures using additional contact supports by means of hands and/or any other limbs; and (iii) perform complex locomotion in confined spaces. State-of-the-art multi-contact planning methods cannot yet be computed in real-time *w.r.t.* to an unknown environment. Fortunately, with a lot of applications, especially in industrial settings such as shipyards or buildings, a large part of the environment can be known a-priori. Planning can then be achieved *w.r.t.* to CAD models of the environment surfaces relevant to the tasks at hand.

Dense localization and mapping has established itself as one of the most promising solution to both obtain a precise map and localization of a robot *w.r.t.* to it. Even so, few working solutions have been exploited in the context of humanoid locomotion. Previously real-time walking and navigation coupled to dense real-time vision was first presented with Meilland et Comport [2013a]. More recently real-time footstep-planning abilities coupled to real-time vision have been demonstrated using dense stereo *SLAM* in Fallon et al. [2015] for the case of uneven terrain walking. To our knowledge no attempts have gone further than walking.

We propose a full system that integrates the multi-contact planner and QP controller framework detailed in Bouyarmane et Kheddar [2011b]; Vaillant et al. [2016] with the state-of-the-art 6D dense SLAM detailed in Meilland et Comport [2013a]. Whilst both perception

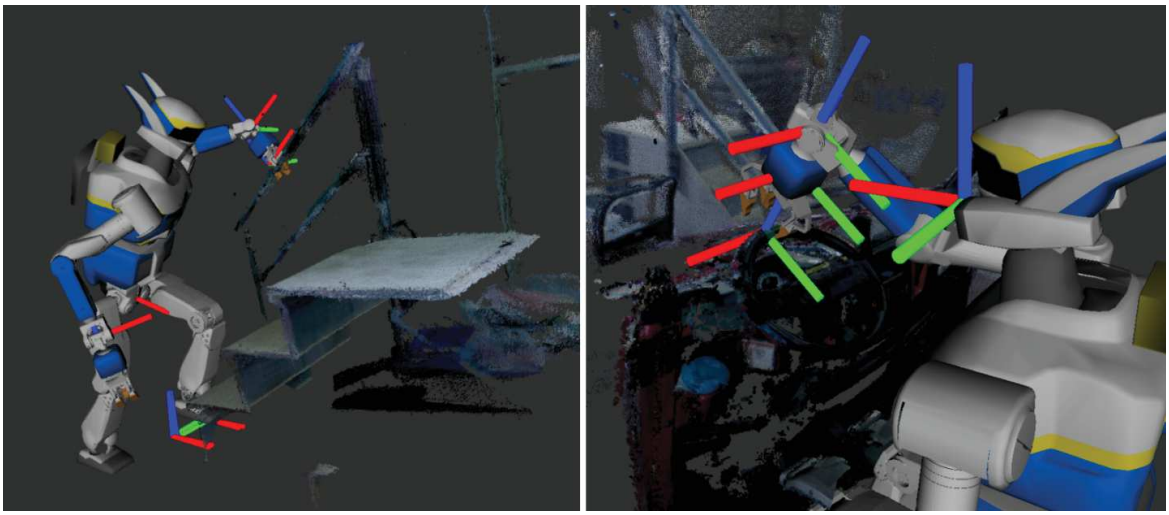


Fig. 3.1 HRP-2Kai reaching for contacts in challenging environments. Localization of the robot is shown *w.r.t.* to *D6DSLAM*'s map.

and multi-contact control modules have been previously published, combining them together is a non-trivial task that has lead to many integration challenges. In this chapter, we will look at how to

1. obtain a 3D representation of the environment?
2. find objects in this representation, and in particular surfaces on these objects?
3. localize a robot *w.r.t.* to these objects?
4. use the localization of the robot *w.r.t.* to desired surfaces to robustly achieve contact with its end-effectors

Answering these questions make it possible to execute multi-contact plans in a real environment, without presupposing that the robot is precisely placed *w.r.t.* to the surfaces of interest. We demonstrate such possibilities via a set of several representative humanoid tasks: walking, climbing and grasping tasks. In summary, the following trials have all been conducted after an extended walking phase starting from an arbitrary unknown initial location: (i) Grasping a valve, and (ii) Grasping a car handlebar. The following are performed without walking, but from an approximate initial location: (iii) Grasping a steering wheel, (iv) Executing the first step of a complex multi-contact stair-climbing plan.

3.1 Registration of polygonal meshes with a keyframe-map

Three dimensional scenes may be stored and manipulated with several representations. That which is most suitable is heavily dependant on the application. In the field of computer graphics, polygonal meshes (represented as a collection of triangular surfaces) are by far the most commonly used for their properties that greatly simplify the rendering process. Robotics has had a long history of using such polygonal meshes, to model robot links, but also the surfaces they are expected to interact with. Another common representation heavily used in robotics application is that of point clouds, in which objects are defined as a set of independent three dimensional points. They are often acquired directly through sensors such as LIDARs.

In the case of *SLAM*, several representations are used depending on the method. In Comport et al. [2007] and subsequent methods [Kerl et al., 2013; Meilland et Comport, 2013a] a keyframe representation is used, while Newcombe et al. [2011a] and others [Whelan et al., 2012] uses a voxel representation. Each representation has its own set of advantages, and provides an efficient way of storing and accessing the map, thus allowing the *SLAM* approaches to run at camera rate. It is necessary to keep these internal representations, as they are best suited for their applications: keyframe-graphs and voxels are well-suited for *SLAM*, meshes for computer graphics, point clouds for 3D sensors, *etc*. Yet it is often necessary to relate these representations to solve a specific problem; here the localization of a 3D meshes *w.r.t.* to *SLAM*'s keyframe-graph.

The first, proposed by Viola et Wells III [1997] exploits *mutual information* (a measure of the quantity of information shared by signals based on entropy) as a cost-function for minimization. Its effectiveness was demonstrated in many scenarios : alignment of a 3D head model (pointcloud representation) with a grayscale 2D image; Computed Tomography (CT) scan and Magnetic Resonance Image (MRI) of a patient, *etc*. In Dame et Marchand [2010], a similar method was used to align aerial images with a template extracted from a geographic map. For our application, one could directly choose to exploit mutual information between the mesh representation and the keyframes. Better yet, a keyframe-map representation of the object of interest could be built a-priori, and then used for registration with the map acquired online. This would allow to fully exploit the rich map built by our *SLAM* algorithm (such as *high dynamic range* [Meilland et al., 2013a]) to obtain robust registration. Unfortunately, due to time constraints preceding the DARPA Robotics Challenge, this option was not explored further.

Instead, we chose to convert both meshes and keyframe-maps to the simpler pointcloud representation, with some trade-offs (conversion time, loss caused by sampling, *etc*). Pointclouds were chosen as the common representation as they are (i) easy to obtain from both

meshes and keyframe-maps, (ii) many registration algorithm have been developed to exploit them. Amongst those, the most popular is the *Iterative Closest Point* algorithm proposed by Besl et McKay [1992] (described in Section 1.2.3 and Appendix B). In the remaining of this chapter, we will show how pointclouds can be sampled from meshes, and converted from keyframe-maps; and present two main robustness improvements that were developed for the DARPA Robotics Challenge. The first improves mapping and tracking of *SLAM* in the presence of self-observations, the second provides robust registration with scale variations between the mesh and pointcloud.

3.1.1 From mesh to pointcloud : uniform mesh sampling

This section focuses on obtaining uniformly distributed pointclouds from meshes. Meshes are assumed to be composed of triangular surfaces, as they are the most widely available, and conversion from other polygonal representations is often trivial. Furthermore, we wish to obtaing a uniformly distributed cloud of points which provides a neutral distribution of points, where no implicit importance is given to any part of the mesh. The method described here is inspired by the python pyntcloud implementation [David de la Iglesia, 2017], and I have implemented it as an open-source C++ library[1] to be used with the registration tools developped in this thesis. This sampling can be further filtered with Poisson disk sampling, to guarantee that all samples are at least distance r apart [Bridson, 2007]. Figure 3.2 shows sampling results obtained by uniformly sampling a total of $N \in \mathbb{N}^+$ points on a triangular mesh as follows.

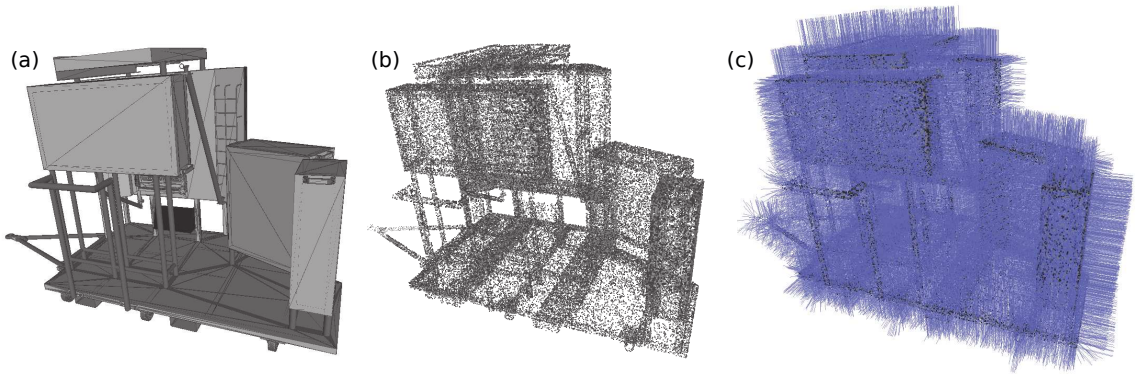


Fig. 3.2 Uniform point cloud with $N = 100000$ points sampled from a triangular mesh of the Airbus A400M demonstrator (scale 1:1 aircraft part). (a) Triangular faces and surface rendered with flat lighting (b) Uniform point cloud representation (c) Uniform point cloud representation with surface normals

[1] https://github.com/arntanguy/mesh_sampling

The mesh is defined here by M triangles. Each of the 3 triangle vertices are represented by its vertex coordinates $\mathbf{v}$, vertex color $\mathbf{c}$ and vertex normal $\mathbf{n}$, such that

$$\begin{array}{rcll} \mathscr{T} & = \{ & \mathscr{T}_1, ..., \mathscr{T}_M & | \quad M \in \mathbb{N}^+ \}, \\ \mathscr{T}_i & = \{ & (\mathbf{v}_1^i, \mathbf{v}_2^i, \mathbf{v}_3^i) & | \quad \mathbf{v} \in \mathbb{R}^3 \\ & & (\mathbf{n}_1^i, \mathbf{n}_2^i, \mathbf{n}_3^i) & | \quad \mathbf{n} \in \mathbb{R}^3 \\ & & (\mathbf{c}_1^i, \mathbf{c}_2^i, \mathbf{c}_3^i) & | \quad \mathbf{c} \in \mathbb{R}^3 \} \end{array} \tag{3.1}$$

,

The amount of points sampled per triangle $\mathscr{T}_i$ is dependent on its area a_i.

$$a_i = \frac{||(\mathbf{v}_2^i - \mathbf{v}_1^i) \wedge (\mathbf{v}_3^i - \mathbf{v}_1^i)||_2}{2} \tag{3.2}$$

The number of point to sample per-triangle can be obtained as

$$s = N * \frac{a_i}{\sum_{j=0}^{M} a_j} \tag{3.3}$$

Thus s_i points need to be sampled per triangle. Randomly sampling a point to lie within a triangle can be easily and efficiently achieved using a barycentric coordinate system. In this coordinate system, a random point lying inside a triangle can be generated simply by a linear combination with weights $\alpha, \beta, \gamma \in [0,1]$ such that

$$\begin{array}{l} \alpha + \beta + \gamma \leq 1 \\ p_s = \alpha \mathbf{v}_1^i + \beta \mathbf{v}_2^i + \gamma \mathbf{v}_3^i \end{array} \tag{3.4}$$

Since $\gamma = 1 - (\alpha + \beta)$, one only needs to randomly choose a value for $\alpha \in [0,1]$ and $\beta \in [0,1]$. To ensure that the barycentric constraint of Equation 3.4 is respected, if $\alpha + \beta \geq 0$ then $\alpha = 1 - \alpha$ and $\beta = 1 - \beta$. Note that if the mesh has vertex colors defined as $c_1^i, c_2^i, c_3^i \in \mathbb{R}$, then they can be interpolated for each sampled point using Equation 3.2 using the color vectors instead of the vertex coordinates.

If the normal direction n^i is known for each vertex v^i, then the sampled point normal can be interpolated as

$$\mathbf{n}_s = \frac{\mathbf{n}_1^i + \mathbf{n}_2^i + \mathbf{n}_3^i}{||\mathbf{n}_1^i + \mathbf{n}_2^i + \mathbf{n}_3^i||_2} \tag{3.5}$$

If the vertex-normal direction is unknown, it can still be obtained as $n = (\mathbf{v}_2^i - \mathbf{v}_1^i) \wedge (\mathbf{v}_3^i - \mathbf{v}_1^i)$. However, in such a case, further steps should be taken to ensure that the normal direction remains consistent across all triangles, and always points towards the outside of the mesh.

Thus, the set of points sampled for a triangle $\mathscr{T}_i$ is

$$\mathscr{P}_i = \{(\mathbf{p}_s, \mathbf{n}_s, \mathbf{c}_s)_1, \ldots, (\mathbf{p}_s, \mathbf{n}_s, \mathbf{c}_s)_s\} \tag{3.6}$$

, and the whole cloud is defined as the union of each triangle cloud

$$\mathscr{P} = \bigcup_{i=1}^{M} P_i \tag{3.7}$$

3.1.2 From keyframe-map to pointcloud

Keyframe-based SLAM methods represent the map as a graph of keyframes, where each keyframe contains both refined color and depthmap images, and its pose is determined *w.r.t.* to nearby keyframes in the graph. This pose relates each keyframe to its neighbours in the graph, which allows to locate it *w.r.t.* to a global reference frame. Furthermore, each point in a keyframe (color and depth) can be re-projected as a 3D point thanks to the pinhole camera model, where Equation 1.17 relates each pixel p_n in a keyframe with its 3D counterpart P_n through the camera intrinsic parameters (see Figure 1.7,1.8). However, when directly applied to all pixels of all keyframes, the resulting pointcloud size quickly becomes considerably large. A voxel-grid filter is used to reduce the amount of points to a desired density. The space occupied by the pointcloud is subdivided into cubes of dimension d. Each cube contains an arbitrary number of points from the original cloud, and only the centroid of these points is kept in the final filtered pointcloud.

3.2 Robust registration of 3D objects with dense visual SLAM

In the previous section, we have shown how 3D meshes and *SLAM*'s keyframe-map can be converted to a common pointcloud representation, and motivated the choice of the Iterative Closest Point algorithm. In this section, we consider several methods of initialization for the algorithm, extend it to support registration of objects with different scales (a requirement of the DARPA Robotics Challenge), and also describe how visual *SLAM* can be made robust to self-observations.

3.2.1 Initial Alignement

In its formulation presented in Algorithm 1, the ICP algorithm is clearly a local method. From an arbitrary initial alignment between pointclouds, no guarantee can be provided that the algorithm will converge globally to the desired solution. This is due to the nearest-neighbour distance formulation, where only the closest point to the current alignment estimate are

considered, which can easily lead to local minimas. Because of this local matching step, the algorithm can only improve the alignment of already pre-registered point clouds, but is not suitable to globally locate a model in a general scene. Solving the registration problem globally is a hard open problem. Recently, Yang et al. [2013] proposed a first solution that guarantees global convergence of the ICP algorithm. It alternates between using *branch and bound* and ICP to efficiently explore the space of possible rotations and translations. While this solution is very promising, it is currently very slow, with solutions ranging from a few minutes to a few hours depending on how much of the space has to be explored. Further work is looking at improving the convergence speed by increasing the convergence radius of the ICP step [Ireta Munoz et Comport, 2017; Straub et al., 2017]. While real-time alignment is not required as *SLAM*'s map remains globally consistent and registration can be performed only once, a long convergence time (currently of the order of minutes) remains prohibitive for real applications.

A pragmatic approach is to find, an initial alignment $\hat{\mathbf{T}}$ that brings the pointclouds close enough to be within the convergence radius of ICP. In Rusu et al. [2009] it was proposed to use *Fast Point Feature Histograms (FPFH)* features to describe local geometry around a subset of 3D points in a cloud, in conjunction with a *RANSAC* algorithm. The idea is to randomly select a set of points in both clouds, around which features are computed. The solution with minimal distance between sets of features is then kept as an initial guess to be refined with ICP. Such methods are interesting, but no guarantee is provided that they will converge to a valid initial guess. Furthermore, in our practical experience, such methods were rarely able to find an exploitable guess. This is due in part to the nature of the environments, which are mostly composed of planar surfaces and do not provide enough local geometry around sampled points. A remaining option is to manually provide the initial estimate, which can be achieved in several ways. Assuming no a-priori knowledge about the model and the environment, the estimate needs to be manually defined by a human operator. This task can be made simple by manually picking correspondences between a minimum of three non-coplanar points in each cloud (or four coplanar ones), and performing a simple SVD decomposition, which guarantees a unique initial guess. This provides a generic way of initializing ICP, at the cost of requiring a human operator.

However, in the context of controlling a humanoid robot, a lot of contextual information is available: what the robot is trying to achieve, where the object is likely to be according to the robot, *etc*. In Tanguy et al. [2016] we showed that this information can often be used in practice to successfully initialize ICP, without requiring human intervention. For example, in an experiment, the robot is placed by an operator in a sitting posture within the driving seat of a vehicle. It's initial posture can thus be guessed with reasonable accuracy: its encoders

provide initialization for the kinematic parameters, and floating base posture can be guessed by supposing the buttocks link to be in flat contact with the seat. To get an initial guess of the driving wheel pose, its pose *w.r.t.* the driving seat is also required, which can be obtained from the car model. Another commonly occurring scenario is that of re-registering an object in case odometry tracking fails. In that situation, assuming the robot hasn't moved much since the loss of odometry, the initial guess can simply be taken as the latest known position of the robot *w.r.t.* the object.

In summary:

- Global ICP registration is prohibitive because of its large computation time. While real-time is not necessary as registration occurs only once while planning the motion, operational time is desired (*e.g.* reactive enough to be used in a teleoperation setting, such as that of the DRC).
- Feature-based initial alignment does not provide any guarantee to succeed, and has been found to be quite unreliable in practice.
- Manual initialization is not ideal as it requires a human operator, but it is sometimes possible to use known information about the robot in its environment to obtain a sufficient initial guess.
- The *SLAM* algorithm re-located based on testing local keyframes for a minimal error and variance threshold could also be used to directly initialize the CAD model within its initial calibration.

3.2.2 Robustness of SLAM

Self-occlusion

With the exception of recent developments in deformable *SLAM* that remain limited to highly controlled motion, such as Innmann et al. [2016]; Newcombe et al. [2015] and many more, state-of-the-art approaches make the assumption of a rigid scene in order to keep geometric consistency throughout tracking. Unfortunately, non-rigid motion within any of the environment in which the robots are expected to evolve (factories, households, *etc*) is rarely avoidable. Motion observed within the camera field-of-view can be separated into motions that do not depend on robot actions (such as a human moving, objects in motion...), and motions that are caused by the robot itself (self-observation of the robot links, interacting with the environment). For the first category, no a-prioi assumption can be made, the motion is entirely outside of our control. Most state-of-the art SLAM systems have some robustness

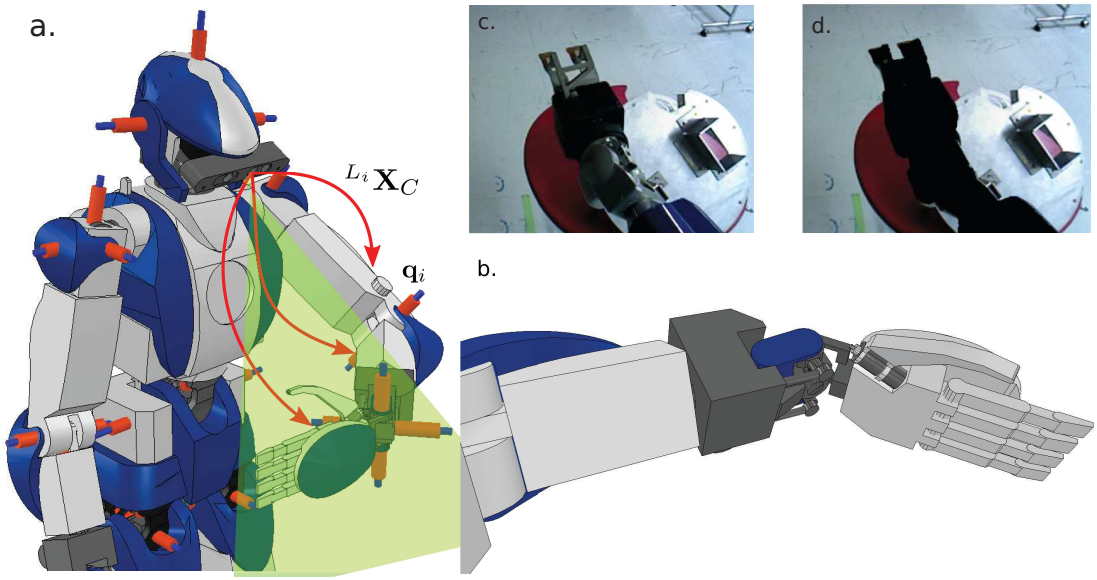


Fig. 3.3 a. An external view of HRP4 CAD model – b. Links visible within the RGB-D sensor filed-of-view (green area) are rendered from the optical frame perspective – c-d. Live example on HRP-2. All visible links are masked.

mechanisms that will allow to discard motion as outliers. *D6DSLAM* uses iteratively re-weighted least-square with M-Estimators to filter out outliers (Appendix A). This method is robust to small discrepancies within the field-of-view, but cannot handle the large image occlusions that are likely to occur with self-observation. Imagine the robot performing the valve turning scenario of the DRC. It will need to look at the valve, and grasp it with it's end effector. In this scenario, the arm motion occupies a large part of the sensor's field-of-view (FoV) and, if left unchecked, can cause poor tracking of *D6DSLAM*, along with erroneous keyframes degrading the map quality.

Contrary to external motions, self-observation is entirely within our control. It cannot be prevented from happening, but we can know when and where they are occurring, and prevent them from interfering with *D6DSLAM*'s tracking and mapping. The main idea is to alter the photometric matching algorithm to avoid taking into account areas where the robot links are visible. To do so, using the robot's kinematic state obtained from forward kinematics of the joint encoders and the robot model, we render a binary mask of the robot's CAD model from the sensor's perspective (see Fig. 3.3). In order to account for small deviations between the CAD rendering and the actual robot state, we apply a small dilatation to the generated mask.

To each of the robot link frame L_i is associated a set of triangles, defined by their vertices $\{\mathbf{v}_i^j\}$ (refer to Section 3.1 for a description of mesh representation). Each vertex can be

expressed *w.r.t.* to the camera frame as

$$ {}^{\mathbf{v}_i^j}\mathbf{X}_C = \mathrm{FK}(\mathfrak{B}, L_i, \mathbf{q})^{-1}\, {}^{\mathbf{v}_i^j}\mathbf{X}_{L_i}^{-1}\mathrm{FK}(\mathfrak{B}, C, \mathbf{q}) $$

Each triangle, defined by 3 of these vertices, can then be rasterized onto the image plane using the pinhole model of Equation 1.17. This equation links a pixel $\mathbf{p}_i$ on the image plane with a 3D point through the intrinsic matrix. The position of a vertex *w.r.t.* to the camera itself depends on the extrinsic sensor calibration, which once again illustrates the importance of Hand-Eye calibration. Rasterization is achieved efficiently with *GLSL fragment shaders*, which results in a mask defined as

$$ \begin{aligned} M : \Omega &\to [0,1] \\ (\mathbf{p}_i) &\mapsto \begin{cases} 0, & \mathbf{p}_i \text{ is in at least one link triangle} \\ 1, & \mathbf{p}_i \text{ is not in any link triangle} \end{cases} \end{aligned} $$

D6DSLAM's tracking equation 1.23 is modified to incorporate this mask in the following way

$$ \mathbf{e}_m(\mathbf{x}) = \mathbf{M}\left(w(\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x}), \mathbf{P}^*)\right)\mathbf{e}(\mathbf{x}) \tag{3.8} $$

In effect, we are considering the error as an outlier at each pixel where the robot is visible. Figure 3.3 shows an example of this masking strategy on HRP-2.

While this masking scheme effectively improves the robustness, potentially large portions of the input sensor data are effectively ignored, which can lead to tracking inaccuracies, or even in extreme cases loss of tracking. In addition to dealing with self-occlusion, some steps, not considered here ought to be taken in order to update *D6DSLAM*'s map according to the robots predictable actions within the environment, i.e. if the task is to move an object, then the corresponding object in the map ought to be updated.

Pure rotations

Our experiments revealed an issue that seems to be inherent to the dense visual SLAM approach, whereby pure rotations in poorly textured environments do not sufficiently constrain the tracking equation 1.23, which leads to erroneous estimations. Instead of a pure rotation, considerable drift in translation in a plane normal to the rotation axis is observed (as much as 10cm in extreme cases). For instance, a pure rotation of 55 along the vertical axis z might lead to 13cm error along the y axis (Figure 3.4). This occurs mainly when there is little texture in the environment, and the issue has been observed with two of the leading dense

SLAM approaches: *D6DSLAM* [Meilland et Comport, 2013a] and ElasticFusion [Whelan et al., 2015b].

Looking at textured parts of the environment with a minimal amount of large planar surfaces helps to better constrain the visual tracking equations. As the robot is capable of changing the camera viewpoint, active vision [Aloimonos et al., 1988] was considered to improve the performance of SLAM in such environments. The main idea is to automatically choose the viewpoint that maximizes the amount of information (entropy) constraining the tracking cost function (1.23). However, this idea was not investigated further, and instead a human-in-the loop approach was favoured. We implemented a new task within our QP controller, that drives the RGB-D sensor to look at specific surfaces in the environment. This task minimizes the error between a vector normal to the focal plane of the RGB sensor and a vector going from the RGB sensor to the environment's surface. These surfaces were provided manually during the planning phase to avoid viewpoints that would lead to known tracking inaccuracies. In the future, it would be interesting to reconsider the use active vision strategies, which would provide robustness improvements in a-priori unknown environments.

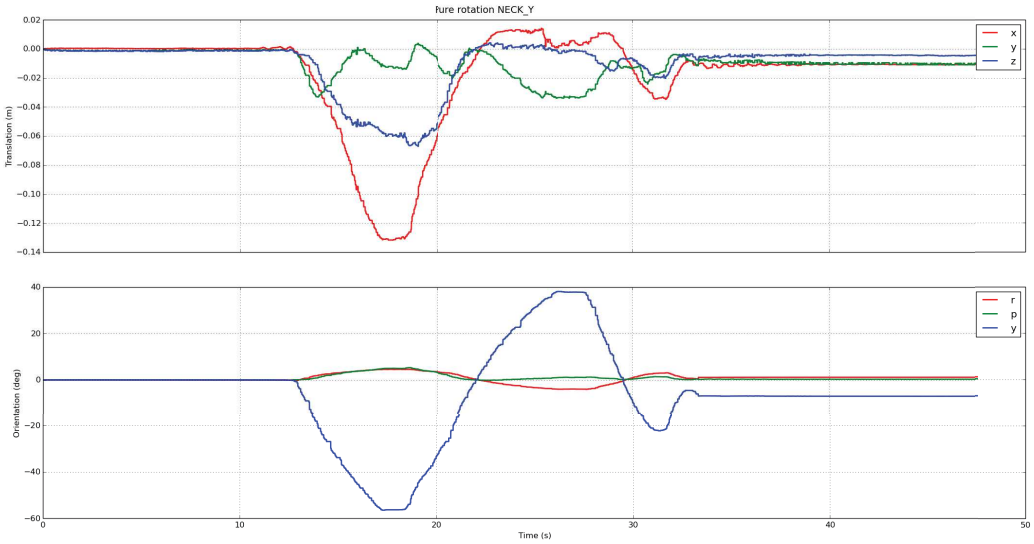


Fig. 3.4 Example of tracking error with pure rotation in untextured environment. The RGB-D sensor is rotated manually mainly along its yaw (vertical) axis. As this is a near-pure rotation, no translation is expected, but erroneous translation of up to 13cm along the yaw axis is observed.

3.2.3 ICP Robustness

One of the prerequisites of the DARPA Robotics Challenge was the ability to handle some discrepancies in the objects laid out in the final's test environment. The expected discrepancies were of two kind: either a difference of scale on the whole model (such as bigger steps on a staircase, bigger valve...), or variations in the shape of the object (such as a valve with 3 or 4 inner connecting rug).

Robustness to scale-variations : ICP formulation in Sim(3)

Scale discrepancies can be integrated in all of the previous ICP formulations with minor alterations. The state is redefined as an element of Sim(3) with the addition of a scale parameter $\lambda \in \mathbb{R}$. The formulation of the cost-functions remain the same, and the Jacobian can be trivially recomputed for the Sim(3) formulation. Solving equation B.1 then simultaneously finds the best pose that aligns the model with the environment, and the scale factor that minimizes the discrepancies between the model and the target cloud.

$$\begin{aligned} \mathbf{T} &= \begin{bmatrix} \lambda r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & \lambda r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & \lambda r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{SIM}(3) \\ \xi &= [v_x \quad v_y \quad v_z \quad \omega_x \quad \omega_y \quad \omega_z \quad \lambda] \in \mathrm{Sim}(3) \end{aligned} \tag{3.9}$$

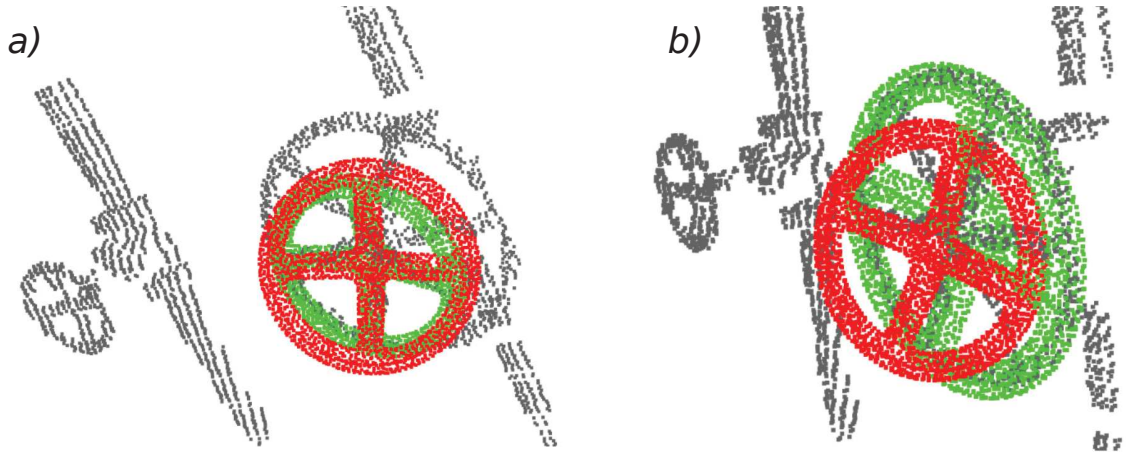


Fig. 3.5 ICP alignment for the DRC valve task. A pointcloud is sampled from a valve 3D mesh, and placed at an initial location (red). It is then registered (green) onto a pointcloud acquired from HRP-2Kai LIDAR sensor (grey). *a)* point-to-point ICP - due to scale differences between the model and the real valve, the alignment is of poor quality. *b)* point-to-point ICP in $\mathrm{Sim}(3)$ is able to find the correct scale factor $\lambda = 1.3$ and obtain robust registration results.

Figure 3.5 shows an example of this method used on the DRC Valve turning task. Note that the scale and alignment are correctly estimated here, even though the number of rungs in the valve differs. To obtain better results, one could use a parametric model for the valve rungs, and estimate the best number of rungs online by performing successive ICP alignments and gradually increasing the number of rungs depending on the residual error.

Detecting discrepencies between the model and the environment

One of the steps of the ICP algorithm consists in computing the distance between the CAD model and map point clouds. Once the ICP algorithm has converged, this error directly represent the minimal distance between each point in the CAD model cloud and its closest-neighbour in the map. This metric can be used to detect large model discrepancies: model points, where the error is larger than usual, are likely to be outliers, *e.g.* pieces that are present in the model but not in the environment. Figure 3.6 provides a visual representation of this metric as a heatmap. A smart way to define a threshold to discriminate between true outliers and points that are within the registration error, is to use M-Estimators on the model's distance map. Indeed, assuming that more than 50 percent of points in the model are present in the environment and have been correctly aligned by ICP, the remaining points can be robustly identified as outliers.

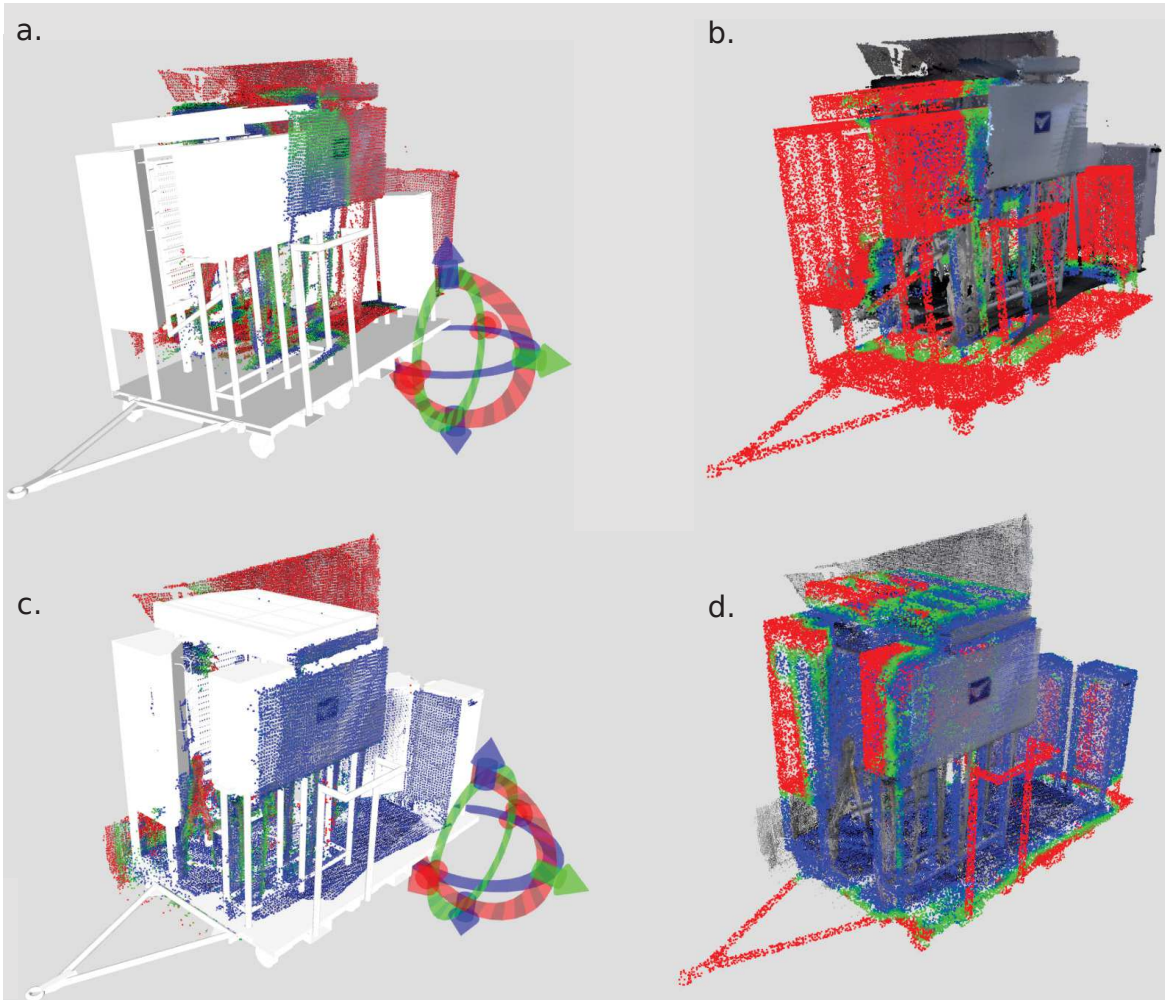


Fig. 3.6 Registration of a CAD model of an A400M Aircraft circuit breaker panel with the environment reconstructed from *D6DSLAM*. *a,b.* Initial registration can be modified manually using an interactive 3D marker, *c,d.* Registration results. Heatmaps represent the closest distance between the model and pointcloud and vice-versa, scale ranges from no error in blue to red which represents 2cm of squared distance (1.4cm). Notice that from this heatmap, it is clear that there are discrepancies between the model and the pointcloud, such as a missing hand-rail, since some areas that were not reconstructed as the environment wasn't thoroughly scanned.

ICP-based Extrinsic RGB-D Calibration

Before developing the dense Hand-Eye calibration method presented in Chapter 2, we formulated a calibration procedure relying on self-observation of the robot end-effectors. Inspired by the self-observation methods requiring calibration patterns, such as Birbach et al. [2012], we devised a pattern-free self-observation method by using ICP to align the 3D mesh of the robot with the corresponding observed pointcloud obtained with SLAM.

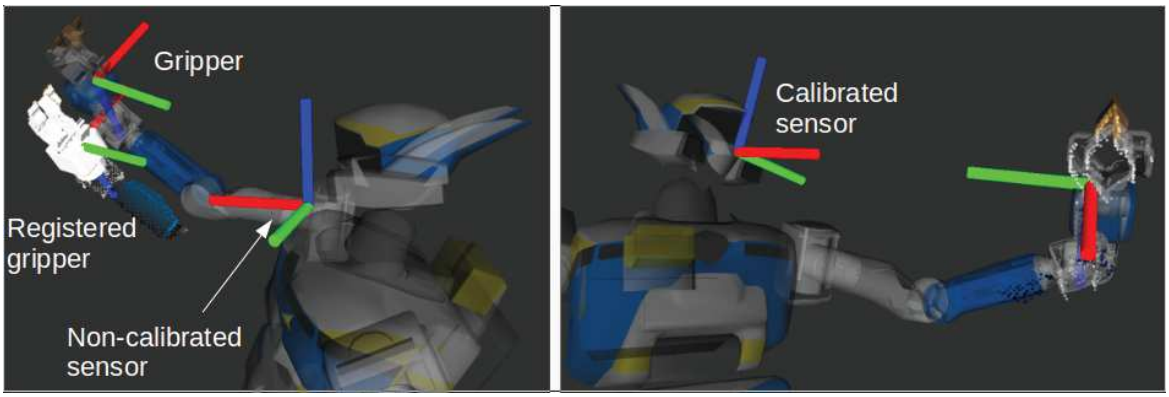


Fig. 3.7 Example of ICP-based extrinsic calibration. Left shows the uncalibrated sensor pose, the gripper frame, and a CAD model of the gripper registered onto the pointcloud acquired from *D6DSLAM*. The right part validates the calibration result on a posture that was not part of the calibration set. Notice that the pointcloud is correctly aligned onto the robot's gripper.

The full transformation from an arbitrary inertial frame 0 to the end-effector frame E can is expressed as

$$^{E}\mathbf{T}_0 = {}^{E}\mathbf{T}_C \, {}^{C}\mathbf{T}_H(\mathbf{x}) \, {}^{H}\mathbf{T}_0$$

where

- $^{C}\mathbf{T}_H(\mathbf{x}) \in \mathbb{SE}(3)$ Is the transformation between the robot's head link and the RGB-D optical frame, *i.e.* the pose to be calibrated
- $^{E}\mathbf{T}_C \in \mathbb{SE}(3)$ is the transformation between the camera optical frame and the end-effector, estimated with ICP.
- $^{H}\mathbf{T}_0 \in \mathbb{SE}(3)$ is the transformation between the inertial frame and the robot's head link, obtained by forward kinematics of the robot encoders
- $^{E}\mathbf{T}_0 \in \mathbb{SE}(3)$ is the transformation between the inertial frame and the robot's end-effector link, obtained by forward kinematics of the robot encoders

This leads to the following iterative least-squares minimization formulation

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{i=1}^{N} || \, {}^{E}\mathbf{T}_C \, {}^{C}\widehat{\mathbf{T}}_H \operatorname{expm}(\mathbf{x}) \, {}^{H}\mathbf{T}_0 - {}^{E}\mathbf{T}_0 || \tag{3.10}$$

The robot assumes a set of N random but feasible static postures for which the arm gripper is visible in the camera sensor's field-of-view (see Fig. 3.7). For each posture, a simple head-scan motion is performed to generate a 3D map of the gripper and obtain the required

transformation ${}^{E}\mathbf{T}_{C}$. All other transformations are obtained through forward kinematics of the current robot state, defined by the robot model and joint encoders. This non-linear error is iteratively minimized using a Gauss-Newton approach such that

$$\mathbf{x} = -(\mathbf{J}^{\top}\mathbf{J})^{-1}\mathbf{J}^{\top}\mathbf{e}, \tag{3.11}$$

where $\mathbf{J}$ contains the stacked Jacobian matrices of the errors of equation 3.10 (refer to Equation 2.5 for a similar analytical computation) and $\mathbf{e}$ the stacked errors from each of the N posture measurements. The calibration pose is updated with

$${}^{C}\widehat{\mathbf{T}}_{H} = {}^{C}\widehat{\mathbf{T}}_{H}\,\mathrm{expm}(x)$$

The final pose ${}^{C}\widehat{\mathbf{T}}_{H}$ yields the transformation from the previously non-calibrated sensor pose to the calibrated one. Its accuracy depends on several factors: accuracy of the encoders, quality of the registration and variability in the gripper postures. Due to the lack of variability in the possible self-observable motions that can be generated, this calibration method is far inferior to the proposed dense Hand-Eye calibration.

3.3 Control *w.r.t.* to the Environment

We have shown how a 3D environment can be acquired with keyframe-based dense SLAM, and how the mesh of an object can be located within that environment thanks to ICP registration. Let's now consider how to exploit this capability to plan and execute multi-contact actions *w.r.t.* to the environment.

3.3.1 Multi-Contact Planning and Registration

A multi-contact planning algorithm provides a sequence of contacts between surfaces on the robot and surfaces in the environment. Typically, this plan is generated *w.r.t.* to CAD models of the environment, whereby contact surfaces are either automatically sampled, or manually provided. In any case, during the offline planning phase, a set of contact surfaces are defined with respect to the reference CAD model M of the expected environment with which the robot needs to interact (staircase, car, ladder...). A frame on each contact surface, where the robot end-effector will need to establish contact is defined such that the pose of each contact *w.r.t.* to the model M is $\mathscr{C} = \{{}^{C_1}\mathbf{X}_M, \cdots, {}^{C_n}\mathbf{X}_M\}$. With the proposed ICP-based method, this model can be registered onto the *SLAM* map defined *w.r.t.* to a global reference

frame S, which provides the transformation ${}^{M}\mathbf{X}_{S}$. The pose of each contact within *SLAM* map is then obtained as ${}^{C_i}\mathbf{X}_{S} = {}^{C_i}\mathbf{X}_{M}\,{}^{M}\mathbf{X}_{S}$. Figure 3.8 provides an overview of this process.

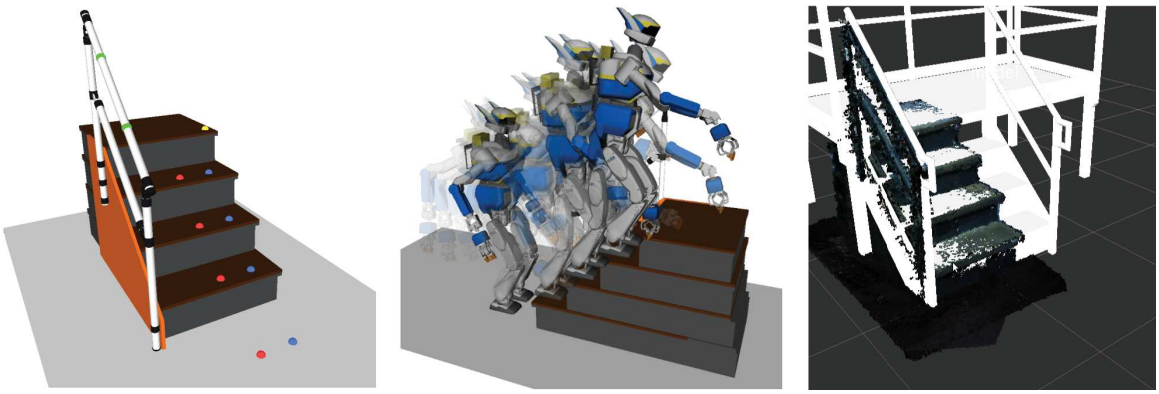


Fig. 3.8 Generation of a Multi-Contact stair climbing plan. Suitable contact surfaces are identified on the 3D model of a staircase: red dot for left foot contact, blue dot for right foot contact, and green dot for hand-to-handrail contacts. The Multi-Contact planner finds a sequence of statically stable postures and contact transitions needed to perform the step climbing motion. Finally, ICP is used to relate this offline plan with the real staircase through registration of the 3D model with SLAM's acquired map, converted to pointcloud representation.

3.3.2 Closed-loop End-Effector Control

In our multi-contact planning framework [Escande et Kheddar, 2009], contact transitions are achieved by removing one contact, while ensuring that the robot remains statically-stable on all remaining contacts. The robot end-effector, that we denote as E, is then controlled to reach the next desired contact frame, that we denote as C_d. In practice, the whole-body QP controller ensures that the multi-contact planner constraints are satisfied (stability by placing the CoM *w.r.t.* to the other contacts, joint and torque limits, *etc*), and is tasked with moving the end-effector. In this section, we show how the localization of the robot *w.r.t.* to *SLAM* map can be exploited to achieve closed-loop end-effector control, and robustly reach contact surfaces in the environment, even in the presence of perturbations. Furthermore, the initial configuration $\mathbf{q}_i$ of the robot does not need to correspond to the initial configuration $\mathbf{q}_i^*$ considered during the offline planning phase, as long as the end-effector can still reach the desired contact surface without breaking any constraints.

First, let's express the end-effector position ${}^{E}\mathbf{X}_{S}$ *w.r.t.* to the *SLAM* reference frame S.

$$ {}^{E}\mathbf{X}_{S}(t,\mathbf{q}) = {}^{E}\,\mathbf{X}_{C}(\mathbf{q})\,{}^{C}\mathbf{X}_{S}(t) \tag{3.12} $$

, where ${}^{C}\mathbf{X}_S(t)$ is the pose of the RGB-D sensor frame tracked by *SLAM*, and ${}^{E}\mathbf{X}_C(\mathbf{q})$ is the transformation between the RGB-D sensor frame and the robot's end-effector frame, obtained by forward kinematics using the robot high-precision optical encoders. Note that this transformation depends on extrinsic hand-eye calibration (Chapter 2) between the RGB-D sensor and the robot's head-link, and as such, any calibration error will lead to imprecision in end-effector control within SLAM map.

The relative transformation between the end-effector and the desired contact surface is

$$ {}^{C_d}\mathbf{X}_E(t,\mathbf{q}) = {}^{C_d}\mathbf{X}_S \, {}^{E}\mathbf{X}_S^{-1}(t,\mathbf{q}) \tag{3.13} $$

This defines a tracking error that can be added as an end-effector task to our QP controller, which will find the robot configuration $\mathbf{q}$ that minimizes the error (3.13), while respecting the constraints imposed by the multi-contact plan. Note that the cost function here depends on odometry, which means that the error will be altered according to both *SLAM* tracking and encoder readings. As such, it is able to react to perturbations (see Figure 3.13). Also note that this method only supports local adjustments: if the current robot configuration is too far away from the one initially planned, it might be impossible for the local QP controller to realize the motion while satisfying all constraints. In that case, replanning becomes necessary, which can not be achieved online with our current framework.

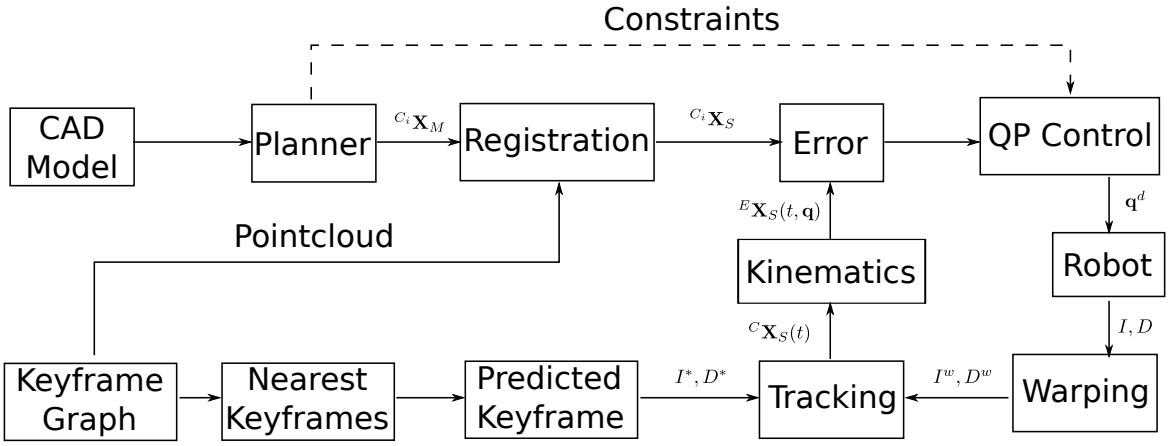


Fig. 3.9 Closed-loop end-effector control. Contact frames are defined *w.r.t.* to a CAD model by a multi-contact planner, and then registered onto a keyframe-graph map acquired with dense visual SLAM. The RGB-D pose is tracked, and used to compute the position of the robot end-effector in the *SLAM* map. A task is formulated based on the relative error between the end-effector and the desired contact, and a QP controller computes the desired joint configuration that moves the end-effector towards this frame.

3.4 Experiments and Results

The following experiments aim to demonstrate the ability of the autonomous system to reach the desired contacts, and execute multi-contact plans within an environment where no a-priori map, nor well-calibrated initial pose for the robot is available. All trials are carried out on an HRP-2Kai (used for the DRC), using a low-cost Asus Xtion Pro Live RGB-D sensor and *PIXMAP*'s *D6DSLAM* system [Meilland et Comport, 2013a].

3.4.1 Walking phase

For the walking phase two fixed walking targets are defined *w.r.t.* the registered CAD model $\mathscr{M}$. The first is a waypoint used to manually ensure a collision-free path to the second, a final target is placed at the expected starting position used during the offline multi-contact planning phase. The posture generator and walking controller from [Kajita et al., 2006] are used to effectively execute the walk. Although technical limitations did not allow us to adjust the walking plan online *w.r.t.* visual odometry, *D6DSLAM*'s tracking and mapping is left active so as to demonstrate its ability to provide sufficiently robust localization to achieve other tasks. Chapter 4 will demonstrate the ability to robustly walk to a desired Cartesian location with dense visual *SLAM* and a closed-loop MPC (see Figure 4.9).

3.4.2 Valve

Inspired by the valve task of the DRC, we aim to, after an extended walking phase from an uncalibrated initial location, establish a gripper-valve contact that would allow us to manipulate it.

Setup

HRP-2Kai is placed in an initial configuration ensuring that the valve is visible within the environment to be mapped. That is, facing in the direction of the valve from roughly 4 meters away. The walking targets of Section 3.4.1, along with a grasp target are attached to the valve CAD model.

Walk

As no a-priori map is available, not enough information is yet available for precise registration. Thus, a rough manual initial registration is performed, which provides walking targets (waypoint and destination). As the robot walks, the *D6DSLAM* algorithm maps and tracks

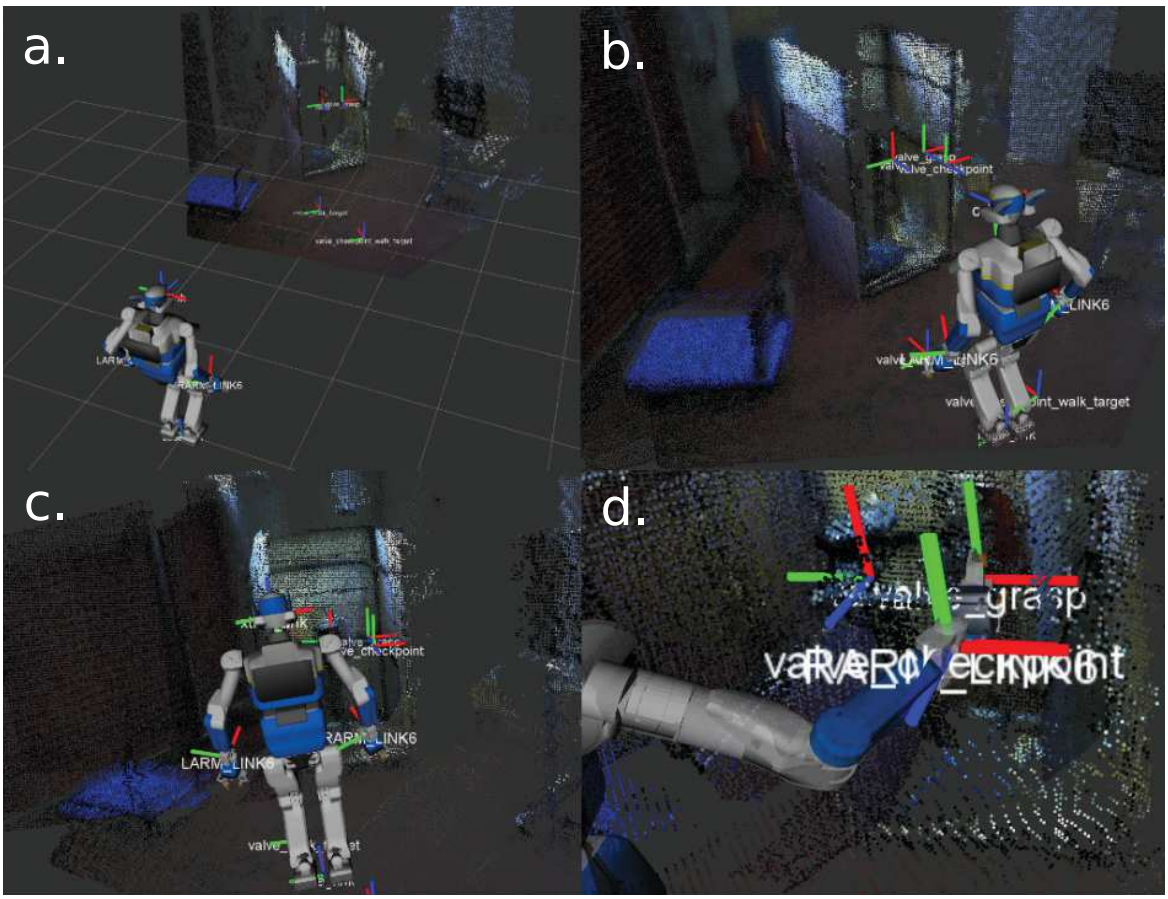


Fig. 3.10 HRP-2Kai localized within *D6DSLAM*'s 3D reconstructed map at various stages of the valve-grasping trial. *a.* Initial position arbitrarily chosen far from the valve, *b-c.* Walking towards the valve. The SLAM map becomes more accurate as new images are observed. *d.* Grasping the valve using the proposed end-effector control method.

the robot's motion. Rolling-shutter and motion blur estimation [Meilland et al., 2013b] are used to minimize the impact of the high velocity and jerkiness of the walking motion. Still, we observe some inaccuracies in the pointcloud, such as the ones visible in Fig. 3.10, most notably on the rightmost part, where sections of the pointcloud are mapped in double. This is most likely due to inaccuracies in the pose estimation of some keyframes added during the fast walking motion. This problem could be lessened by first scanning the environment from a stable posture, and walking within the scanned area without mapping, to prevent the creation of inaccurate keyframes. Instead, we opted to perform a head-scan of the target (without resetting the map) from the final walking target, to provide a locally accurate map followed by a second more accurate registration.

Grasp

A simple control strategy is adopted. Without a-priori planning, the QP-controller is tasked with moving the end-effector to the valve target. In order to avoid collisions with the valve, a waypoint in both position and orientation is added directly in front of the desired contact. Throughout the whole motion, the relative pose between the end-effector and its target is updated according to the *D6DSLAM* pose, as described in 3.3.2. For additional robustness, a guarded approach is used whereby the motion is stopped once a force threshold is attained on the gripper. Our experiment shows that, even after walking with live-mapping and no a-priori map, successful grasping may be achieved.

3.4.3 Steering Wheel

HRP-2Kai is manually placed into the driver's seat before being actuated. As such the initial pose of the robot cannot be fully controlled, and trying to blindly grasp the wheel would be impossible. First, a head-scan is performed to acquire a map of the car's dashboard, including the target steering wheel. Even though the initial posture of the robot cannot be fully determined, good convergence of the ICP registration may be achieved without manual initialization, by assuming a plausible initial position of the steering wheel with respect to the robot. We apply the same control strategy as for the valve. A waypoint is defined to avoid collision with the dashboard and the steering wheel, and the target is continuously updated. One should note that due to the narrow space between the robot and the wheel, self-observation is unavoidable. The masking scheme presented in section 3.2.2 prevents this from affecting the tracking and mapping process.

3.4.4 Stairs climbing

While the previous experiments are simple enough not to use a multi-contact planner, stair climbing uses it fully. In this section, we demonstrate our ability to robustly reach planned contacts from an unknown initial configuration, and under heavy perturbations.

Setup

HRP-2Kai is placed in an initial configuration $\mathbf{q}_i$ close to the one $\mathbf{q}_s$ given as input to the MCP. It is placed so that the QP controller is in a configuration where all its tasks are feasible. The stairs are mapped using a simple head-scan, and their planning model is registered, along with all planned contacts.

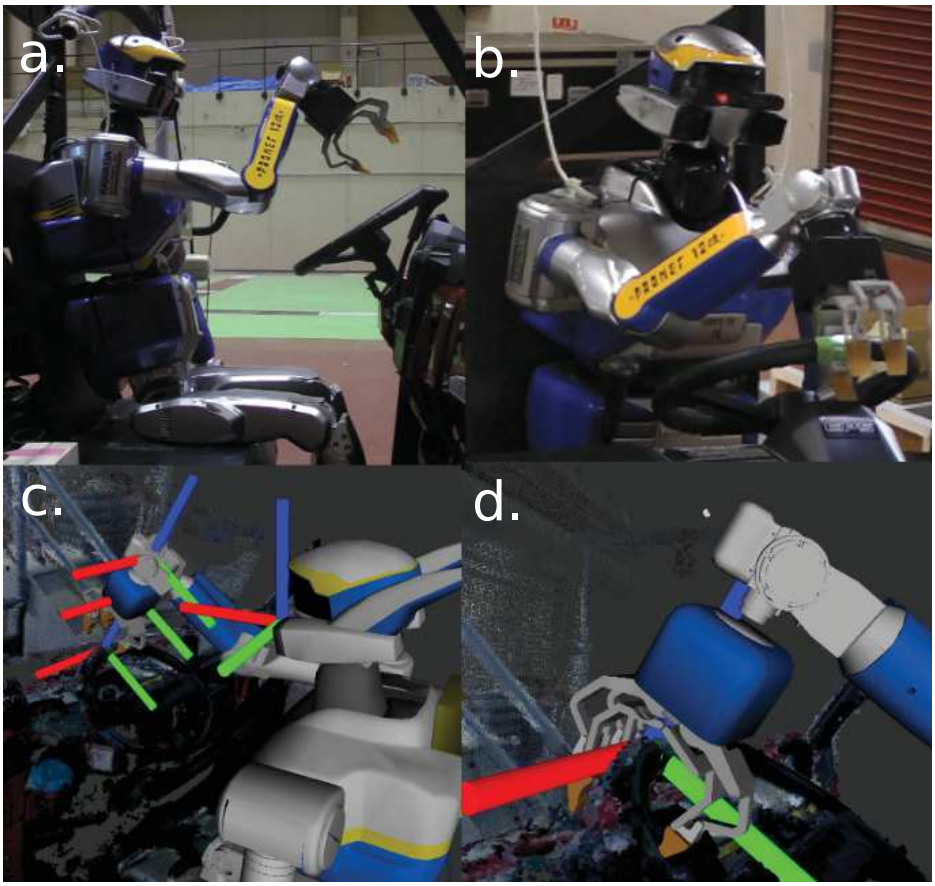


Fig. 3.11 Wheel grapsing. *a-b.* Real HRP-2Kai during various while grasping the wheel. *c-d.* Control view with the robot localized *w.r.t.* to SLAM map.

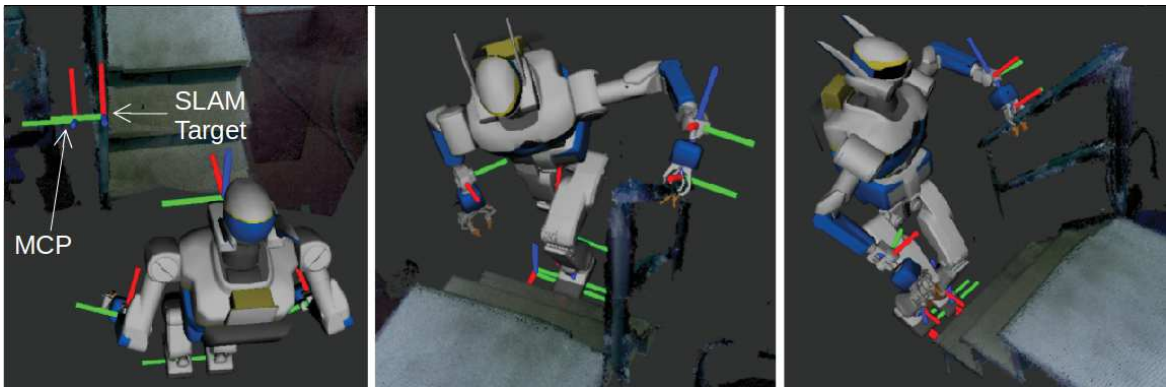


Fig. 3.12 Combined view of the stairs climbing experiment. The left figure shows the gripper's grasping pose (leftmost frame) that would have been tasked by the MCP without registration, and the corrected registered grasping frame onto the handrail. The others show HRP-2Kai climbing the first step.

Closed-loop

The QP is tasked with reaching the first handrail contact of a predefined stair-climbing sequence. Figure 3.13 starts with open-loop control *w.r.t.* to the MCP only and transitions to a closed-loop control strategy. It can clearly be seen that the contact position is modified when transitioning to closed-loop control. This is the result of registration, that accounts for the difference of initial configuration between $\mathbf{q}_i$ and $\mathbf{q}_s$. This can be visually observed in Figure 3.12 where both the planned contact and registered contact are shown. We introduce considerable perturbations by pulling the robot backwards on its ankle flexibilities. The contact trajectory is continuously updated based on *D6DSLAM* odometry as described in 3.3.2, and consequently, the desired contact is reached.

3.5 Conclusion

In this chapter, we have demonstrated how offline multi-contact planning *w.r.t.* to CAD models can be related to keyframe-based maps acquired online with dense visual SLAM. This is achieved by first converting both the keyframe-map and the CAD model to a pointcloud representation, and applying Iterative Closest Point registration to align the CAD model with the observed environment. Robustness to outlier is improved with M-Estimators, and to global scale variation by reformulating ICP cost functions in $\mathbb{SIM}(3)$. Once registration is achieved, the robot can be localized in real-time *w.r.t.* to the environment thanks to the *D6DSLAM* pose tracking, and the robot's kinematic model (including extrinsic camera calibration), obtained by exploiting the calibration method presented in Chapter 2. Robustness to self-observation is achieved by using the current state of the kinematic model to render a binary-mask of the areas where self-observation occurs from the RGB-D sensor's perspective. Knowing the robot's location *w.r.t.* to desired contact surfaces of the multi-contact plan allows to formulate a closed-loop end-effector control method which is shown to robustly achieve contact establishment, even under heavy perturbations.

However, whole-body control of a humanoid robot consists of much-more than just end-effector control. To robustly achieve complex dynamic multi-contact actions, one needs to precisely control its center of mass and interaction forces. In the next chapter, we will see how SLAM can be exploited in the context of closed-loop Model Predictive Control for walking under perturbations.

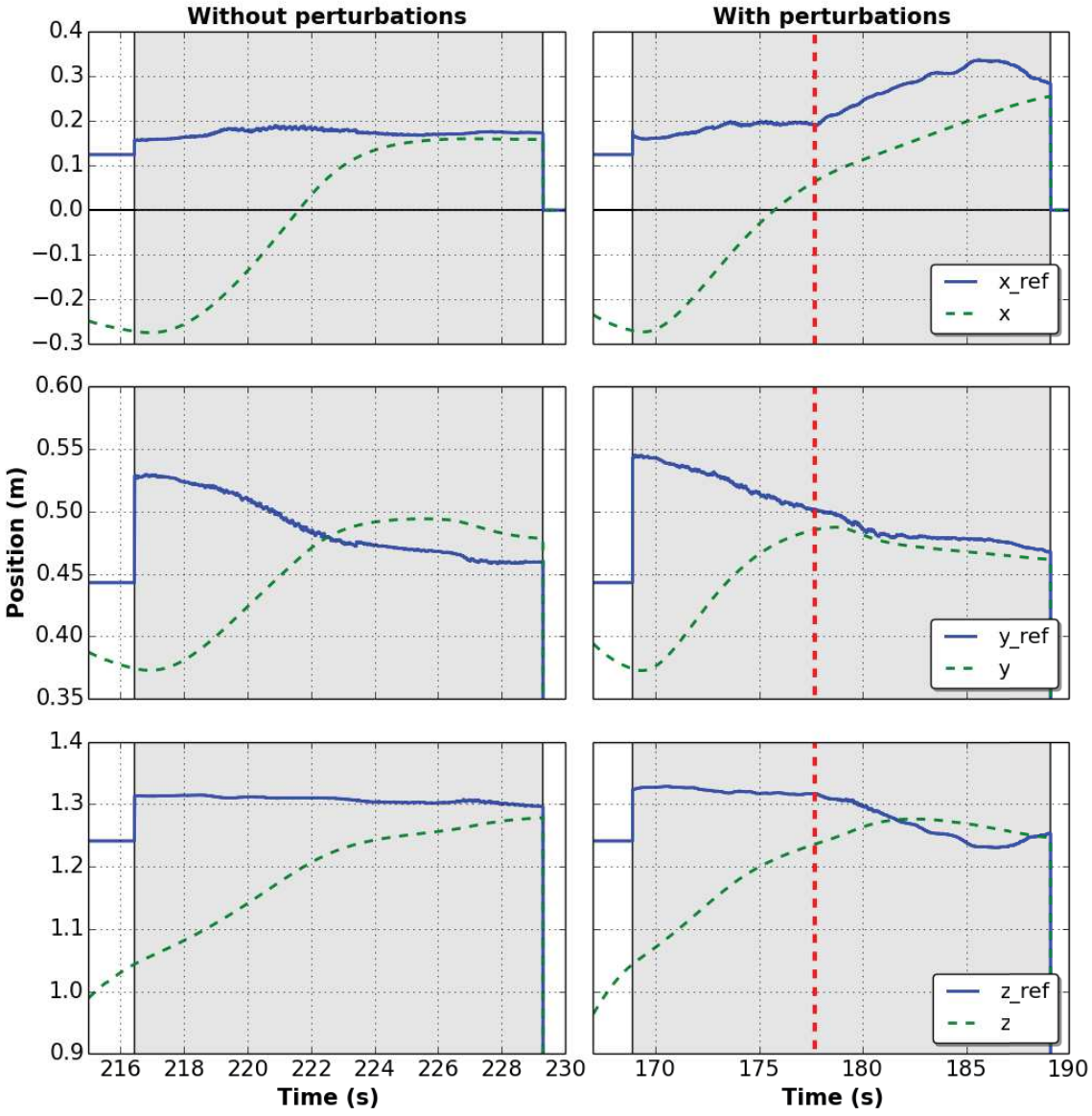


Fig. 3.13 Closed-loop control of the gripper end-effector tasked with grasping the first handrail contact of a multi-contact stair climbing plan. The white area represents control *w.r.t.* the planning reference, while the gray area corresponds to closed-loop SLAM control. Two independent experiments are considered, on the left contact is reached without additional perturbations, while on the right the robot is manually pulled back at the vertical line. The expected contact position $(x_{ref}, y_{ref}, z_{ref})$ and controlled gripper position (x, y, z) are reported.

Chapter 4

Closed-loop MPC with Dense Visual SLAM - Stability through Reactive Stepping

In the previous chapter, we have shown how a dense map generated reconstructed with visual *SLAM* can be exploited to achieve multi-contact plans starting from arbitrary initial configurations. Furthermore, we have shown how end-effectors can be robustly controlled to reach targets based on both SLAM's tracking and the map information. We now consider the problem of locomotion within that map, and show that walking can be seamlessly combined with the execution of multi-contact plans described in the preceding chapter.

Model Predictive Control (MPC) is a widely used technique for humanoid gait generation due to its capability to handle several constraints that characterize humanoid locomotion. The use of simplified models to describe the humanoid dynamics (the Linear Inverted Pendulum) allows to perform computations in real time, giving the robot the fundamental capacity to re-plan its motion to follow external inputs (e.g. reference velocity, footstep plans). However, usually the MPC does not take into account the current state of the robot when computing the reference motion, losing the ability to react to external disturbances. In this paper a closed-loop MPC scheme is proposed to estimate the robot's real state through Simultaneous Localization and Mapping (SLAM) and proprioceptive sensors (force/torque). With the proposed control scheme it is shown that the robot is able to react to external disturbances (push), by stepping to recover from the loss of balance. Moreover the localization allows the robot to navigate to target positions in the environment without being affected by the drift generated by imperfect open-loop control execution. We validate the proposed scheme through two different experiments with a HRP-4 humanoid robot.

Fig. 4.1 HRP-4 walking with the proposed closed-loop MPC with dense visual SLAM feedback. The estimated LIP pendulum state obtained from SLAM and force-sensor measurements is shown as a red line going from the ZMP estimate to the robot's CoM. The green line represents the desired pendulum state. Blue and green squares are respectively the desired optimal footstep, and a non-optimal footstep that minimizes changes of the footstep trajectory.

4.1 Introduction and state-of-the-art

In recent years, humanoid robots have been given increasingly more attention due to their ability to perform complex tasks, thanks to their highly redundant kinematic structure. However, those systems are also challenging to control because they have very complex dynamics. Therefore researchers tend to use simplified models to approximate the humanoid robot dynamics. When dealing with humanoid locomotion, the most commonly used model

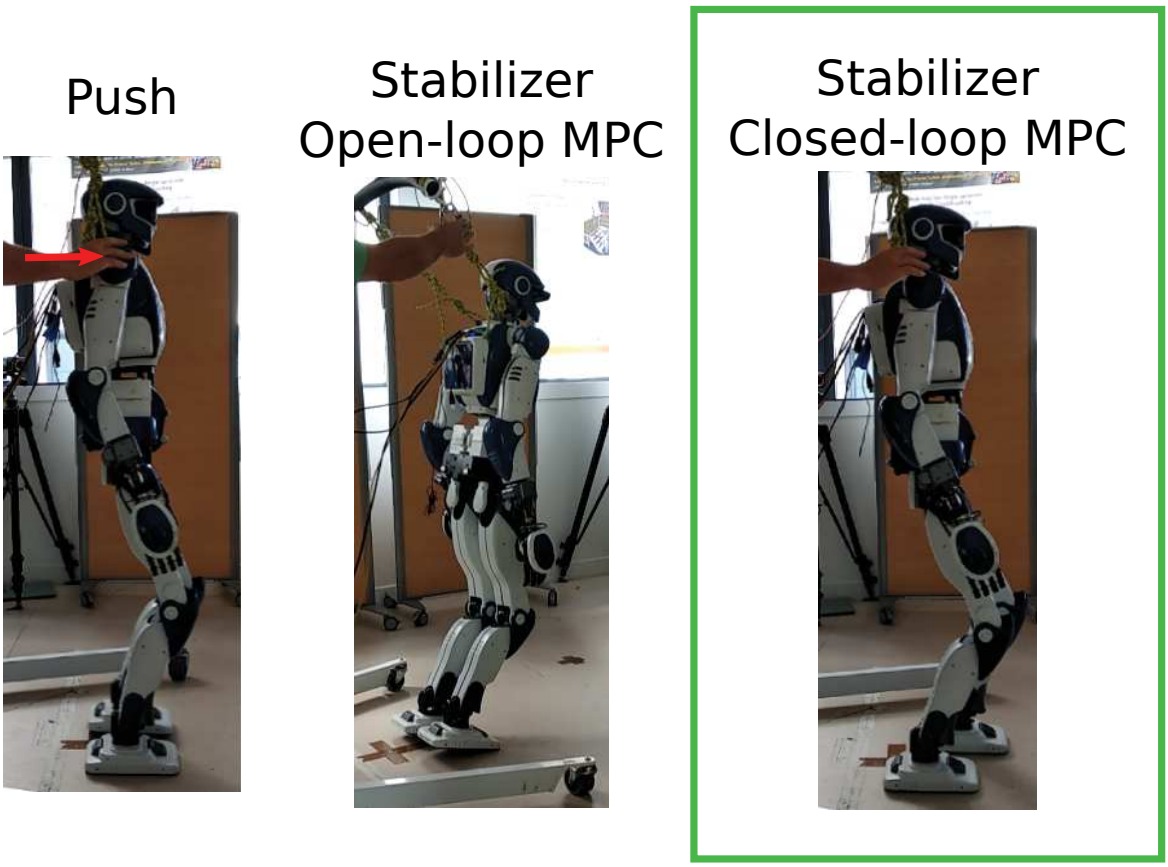


Fig. 4.2 Closed-loop MPC causes the robot to step to prevent the ZMP from going outside of the support foot area, which cannot be compensated with the embedded fixed-footstep stabilizer.

to approximate the dynamics of walking is the Linear Inverted Pendulum (LIP) [Kajita et al., 2014], and the use of Model Predictive Control (MPC) has become predominant [Herdt et al., 2010; Naveau et al., 2017]. In traditional MPC-based gait generation techniques, a reference motion is generated without taking into account the current robot state, however, planning a motion starting from the current robot state is often necessary to react to unexpected situations and to obtain more robust motions. In this paper the problem of closed-loop MPC is tackled [Feng et al., 2016; Villa et Wieber, 2017], and the robot state is estimated using SLAM and force/torque sensor measurements.

The use of SLAM for humanoid locomotion planning has been widely used, since the introduction of dense RGB-D approaches, such as Meilland et Comport [2013a]; Whelan et al. [2012] because they both provide a dense 3D map of the explored environment and 6D localization at sensor framerate with centimeteric precision. With this information, it becomes possible to plan robot actions *w.r.t.* the environment. In Fallon et al. [2015], a pair

of Multisense stereo cameras are used to generate RGB-D inputs for use with Kintinuous SLAM [Whelan et al., 2012]. A footstep plan for walking over a rough brick field is generated from its map, while the robot state is estimated with a kinematic-inertial estimator Fallon et al. [2014]. Footsteps are selected within the map, but no attempt is made to change the next robot's footstep should its state differ from a feasible one. In Scona et al. [2017], kinematic-inertial measurements are used to improve the robustness of ElasticFusion [Whelan et al., 2015b], and are applied to a Cartesian walking controller to repeatedly walk towards goal positions in the environment.

4.2 MPC-based gait generation

When dealing with a complex system like a humanoid robot, it is a common practice to rely on a simplified model, the Linear Inverted Pendulum (LIP) [Kajita et al., 2014] to describe the humanoid behaviour during locomotion. The reason why the LIP is widely used when dealing with humanoid locomotion is because its dynamics approximate well the motion of the Center of Mass (CoM) of a biped robot, and the differential equations governing the two directions of motion x and y are linear, identical and decoupled. So, from now on, only the sagittal component of motion will be referred to.

Consider, without loss of generality, the evolution of the sagittal component of motion x

$$\begin{pmatrix} \dot{x}_c \\ \ddot{x}_c \\ \dot{x}_z \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ \eta^2 & 0 & -\eta^2 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_c \\ \dot{x}_c \\ x_z \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \dot{x}_z \tag{4.1}$$

where x_c is the position of the CoM, x_z the position of the Zero-Moment Point (ZMP), $\eta = \sqrt{g/h}$ and h the constant height of the CoM. In the motion model (4.1) we assume controls $\dot{x}_z$ which are piece-wise constant over time intervals of duration δ.

The gait generation is based on the intrinsically stable MPC [Scianca et al., 2016], where the decision variables are the ZMP velocities $(\dot{x}_z^i, \dot{y}_z^i)$, $i = 1, \ldots, N$ and the footstep positions and orientations $(x_f^j, y_f^j, \theta_f^j)$, $j = 1, \ldots, M$, over a prediction horizon $T_h = N\delta$.

The choice of a MPC-based gait generation allows us to give the robot a high-level task, through an appropriate cost function to be minimized, and also to impose constraints to enforce stability, to maintain balance and to guarantee the kinematic feasibility of the robot motion.

4.2.1 Cost Function

In the formulation proposed here, the aim is to have the robot track high-level reference sagittal and coronal velocities (v_x, v_y), and an angular velocity ω, by an appropriate selection of ZMP velocities $(\dot{x}_z^i, \dot{y}_z^i)$ and footstep positions and orientations $(x_f^j, y_f^j, \theta_f^j)$. However to maintain linearity in the MPC formulation, the footstep orientations must be chosen before the computation of their positions [Herdt et al., 2010].

Therefore, a first optimization problem is solved, minimizing a function that takes into account the reference angular velocity ω to determine the footstep orientations θ_f^j

$$\sum_{j=1}^{M} \left(\frac{\theta_f^j - \theta_f^{j-1}}{T_s} - \omega \right)^2, \tag{4.2}$$

where T_s is the constant duration of a step, subject to the linear constraint $|\theta_f^j - \theta_f^{j-1}| \leq \theta_{max}$, that limits to θ_{max} the maximum difference in orientation between two consecutive footsteps.

Once the foot orientations are decided, the footstep locations and the ZMP velocities can be computed via the minimization of a second cost function

$$\begin{aligned} \sum_{i=1}^{N} \Big(& (\dot{x}_z^{k+i})^2 + (\dot{y}_z^{k+i})^2 + \\ & k_x (\dot{x}_c^{k+i} - v_x \cos(i\omega\delta) + v_y \sin(i\omega\delta))^2 + \\ & k_y (\dot{y}_c^{k+i} - v_x \sin(i\omega\delta) - v_y \cos(i\omega\delta))^2 \Big), \end{aligned} \tag{4.3}$$

where the first two terms penalize the control effort, while the last two terms are meant to minimize the deviation from the reference velocities (v_x, v_y).

4.2.2 Constraints

In the following the three constraints enforced in the proposed MPC scheme will be briefly presented. The first, introduced in Scianca et al. [2016], is the stability constraint, which takes the form

$$\frac{1}{\eta} \frac{1 - e^{\delta\eta}}{1 - e^{N\delta\eta}} \sum_{i=1}^{N} e^{i\delta\eta} \dot{x}_z^{k+i} = x_c^k + \frac{\dot{x}_c^k}{\eta} - x_z^k. \tag{4.4}$$

This constraint will guarantee the boundedness of the computed CoM trajectory.

In order to guarantee balance during locomotion, we must guarantee that the ZMP is at any time instant inside the robot current support polygon. To do so we define a rectangle

with sides $d_x^{\mathrm{z}}, d_y^{\mathrm{z}}$, and therefore the balance constraint takes the form

$$R_j^T \begin{pmatrix} \delta \sum_{l=k}^{k+i-1} \dot{x}_{\mathrm{z}}^l - x_{\mathrm{f}}^j \\ \delta \sum_{l=k}^{k+i-1} \dot{y}_{\mathrm{z}}^l - y_{\mathrm{f}}^j \end{pmatrix} \leq \frac{1}{2} \begin{pmatrix} d_x^{\mathrm{z}} \\ d_y^{\mathrm{z}} \end{pmatrix} - R_j^T \begin{pmatrix} x_{\mathrm{z}}^k \\ y_{\mathrm{z}}^k \end{pmatrix}, \tag{4.5}$$

where R_j is the rotation matrix associated to the angle θ_{f}^j.

The last constraint is to guarantee that the choice of the next footstep is in a location that avoids self collisions and is inside the robot kinematic limits:

$$R_{j-1}^T \begin{pmatrix} x_{\mathrm{f}}^j - x_{\mathrm{f}}^{j-1} \\ y_{\mathrm{f}}^j - y_{\mathrm{f}}^{j-1} \end{pmatrix} \leq \pm \begin{pmatrix} 0 \\ \ell \end{pmatrix} + \frac{1}{2} \begin{pmatrix} d_x^{\mathrm{f}} \\ d_y^{\mathrm{f}} \end{pmatrix}, \tag{4.6}$$

where d_x^{f} and d_y^{f} are the sides of a rectangle defining the feasibility zone, and l is a reference distance between two consecutive footsteps.

4.3 Closed-Loop MPC

Decision variables (future ZMP velocities and next footstep locations) are computed by solving the QP formulation of the MPC described in Sect. 4.2, which requires an initial state of the pendulum, defined by its CoM position and velocity $x_{\mathrm{c}}, \dot{x}_{\mathrm{c}}$ and its ZMP position x_{z}. Traditional MPC gait generation typically envolves this state from an initial value (supposed known) by forward integration of its decision variables through a LIP motion model. This assumes that the underlying system represented by the reduced LIP model is behaving perfectly as expected. Unfortunately, real physical systems are subject to interactions with their environment, which can easily perturbate the state of the system (imperfect contacts, external forces, poor tracking of the reference pendulum by the underlying system). It is proposed here to estimate the state of the system, and periodically use it to compute an updated MPC solution. The next sections will show how this estimation is achieved, and used to obtain a closed-loop MPC formulation.

4.3.1 Estimation of the floating base

The robot's floating base pose ${}^B\mathbf{X}_0$ and velocity ${}^B\mathbf{V}_0$ is a prerequisite to estimating its CoM and ZMP state. In this work, two simple estimators are considered.

The first provides ground-truth measurement obtained from a high-precision VICON mocap system composed of 8 infrared cameras. Reflective markers are placed on the robot's

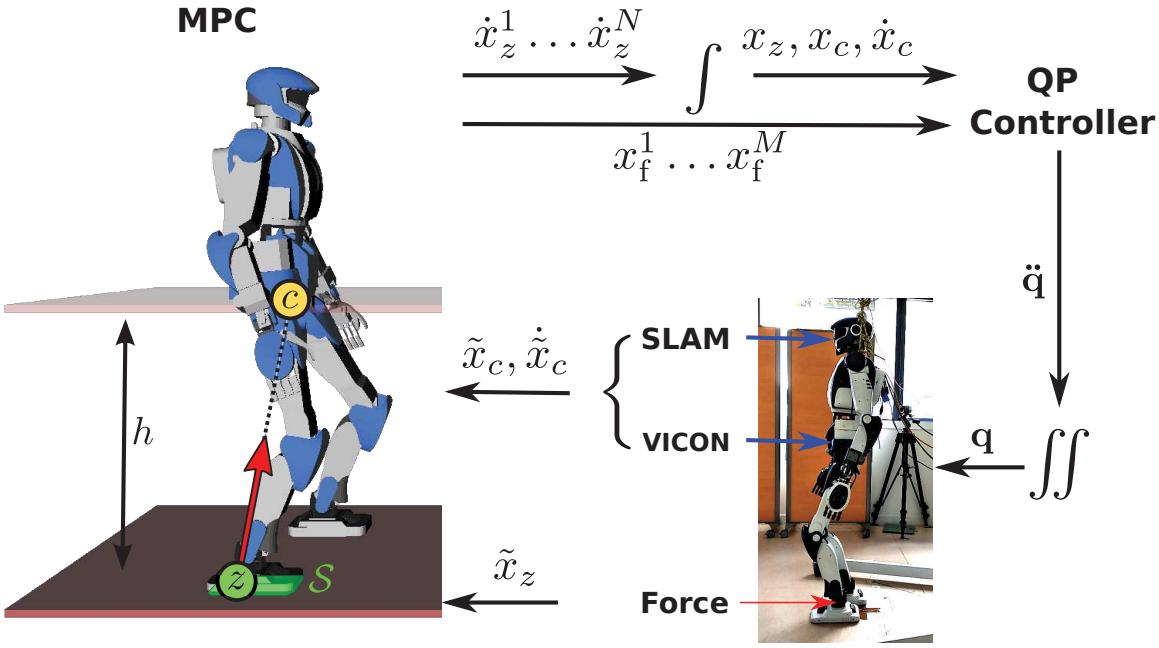


Fig. 4.3 Overview of the proposed closed-loop MPC. A LIP reduced model is used to represent the dynamics of a humanoid robot walking on flat floor. The pendulum state is estimated from HRP-4 sensors: its CoM position and velocity is obtained with dense visual slam from RGB-D measurements, and its ZMP from force-sensor measurements. This state is used to compute the MPC predictions, *i.e.* the future ZMP velocities and associated footsteps. Through integration, these references converted to reference position are then tracked by a quadratic programming controller that sends a desired whole-body configuration to the position-controlled robot.

floating base link, forming a frame V, and tracked by the system with high accuracy at a rate of 100Hz.

$$ {}^{B}\mathbf{X}_0 = {}^{B}\mathbf{X}_V \, {}^{V}\mathbf{X}_0. \tag{4.7} $$

The second is obtained from the dense visual SLAM method of Meilland et Comport [2013a], where RGB-D images from the robot's Asus Xtion sensor are used to estimate the pose of the sensor's optical frame S. Contrary to IMU-based estimators usually used in walking scenarios, visual SLAM provides the full position and orientation in a global world-frame.

$$ {}^{B}\mathbf{X}_0 = {}^{B}\mathbf{X}_H \, {}^{H}\mathbf{X}_S \, {}^{S}\mathbf{X}_0. \tag{4.8} $$

Note that the transformation between the VICON markers frame and the floating base ${}^{B}\mathbf{X}_V$, and between the camera optical frame and the robot's head link ${}^{S}\mathbf{X}_H$ requires calibration. Both are obtained by Hand-Eye calibration [Tanguy et al., 2018b] with *RobCalib* software[1].

[1] https://github.com/arntanguy/robcalib

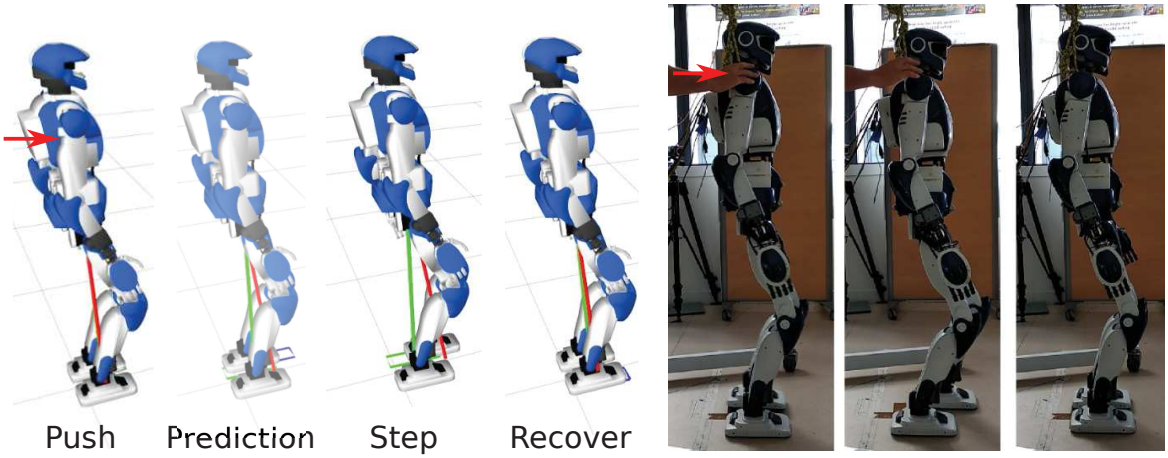


Fig. 4.4 Reaction to a push during a real experiment with HRP-4. During the push, the ZMP-CoM pendulum estimated from SLAM and force sensor measurements reaches the edge of the foot (red line) while the robot's CoM is displaced forward. As a result, the MPC computes a new optimal footstep (blue square) and computes a footstep trajectory to bring its next swinging foot there. The cartesian-regulator controls the MPC target velocity to walk back to its initial position.

The transformation between the head link and the floating base ${}^{B}\mathbf{X}_{H}$ depends on an accurate kinematic tree calibration, and joint-encoder measurements. HRP-4 is fitted with high-accuracy optical encoders, providing this information at control rate (200Hz).

Pose estimates obtained from both VICON and SLAM are noisy, and its velocity is not directly estimated. The use of the Savitzky-Golay filter Savitzky et Golay [1964] is proposed, for which an open-source implementation[2] is provided based on Gram-Polynomials [Gorry, 1990]. This filter has two main advantages: first, when applied at the end-point of a filtering-window, no additional delays are introduced, and the n^{th} order derivative can be readily obtained. Second, it can be efficiently implemented as a convolution, whose weights depend on the size of the time window, the order of Gram-polynomials, and its derivative order.

Position can be directly smoothened by the filter. Rotation smoothing for a time window consists of solving the following maximum likelihood

$$\widehat{\mathbf{R}} = \underset{\mathbf{R}\in\mathbb{SO}(3)}{\operatorname{argmin}} \sum_{k} d_{\text{geo}}(\mathbf{R},\mathbf{R}_k), \tag{4.9}$$

where $d_{\text{geo}}(\mathbf{R}_1,\mathbf{R}_2)^2 = \frac{1}{2}||\text{logm}(\mathbf{R}_1^T\mathbf{R}_2)||_{\text{frobenius}}$.
Many methods exist for solving this problem. One of them consists in applying a temporal convolution, followed by an orthogonalization, which can be shown to be a 2^{nd} order Taylor

[2] https://github.com/arntanguy/gram_savitzky_golay

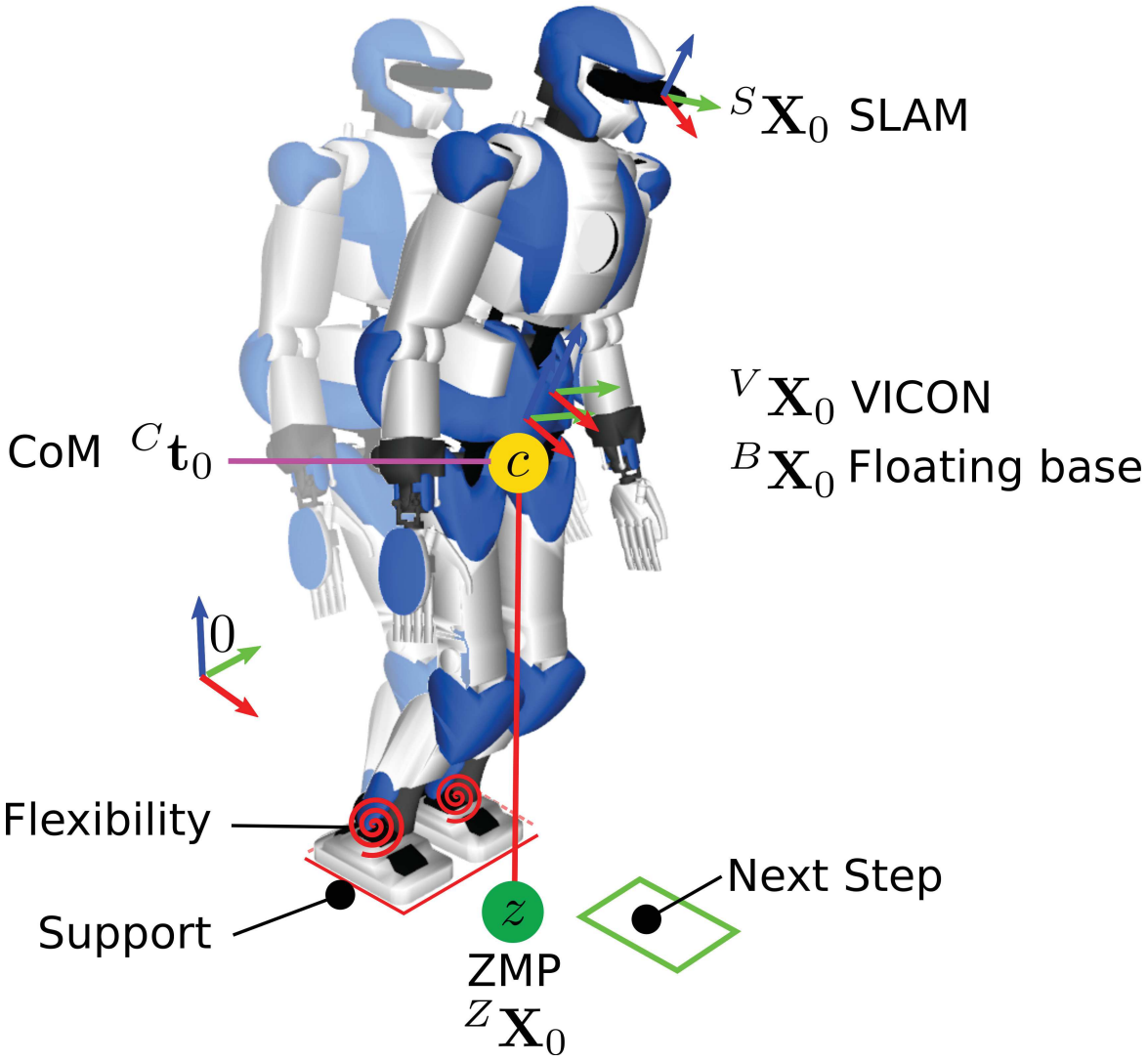


Fig. 4.5 This HRP-4 is fitted with a passive mechanism between its ankles and feet soles, which provides a spring-damper compliant system. Its state is taken into account by estimating the robot's floating base with dense visual SLAM and validated with a VICON tracking system, with reflective IR markers attached to the floating base link. The ZMP is estimated from ankle force-torque sensors. Note that here it is depicted outside of the robot's support polygon and triggers a step from the MPC.

approximation of the geodesic distance [Gramkow, 2001].

$$\begin{aligned} \widetilde{\mathbf{R}} &= \textstyle\sum_k w_k \mathbf{R}_k \\ \mathbf{UDV}^T &= svd(\widetilde{\mathbf{R}}) \\ \widehat{\mathbf{R}} &= \mathbf{UV}^T \end{aligned} \tag{4.10}$$

This implies that rotation matrices can be directly smoothened at any point in a time window by the Gram-Savitzky-Golay convolution coefficients. Consequently, the filtered floating base position and velocity can be readily obtained.

4.3.2 CoM State

Knowing the floating base position and velocity, and the kinematic and inertial model of the robot, the CoM position is trivially obtained as ${}^{C}\mathbf{t}_0$. However, our MPC formulation expects its state to be expressed *w.r.t.* the current support foot. HRP-4 feet are fitted with a passive mechanical flexibility between its ankles and the feet soles that act as a spring-damper to protect the force sensors by reducing impacts, and provide some mechanical compliance while walking. The state of this mechanism is not measured but needs to be taken into account in order to correctly express the CoM in the support foot frame.

In the proposed MPC, the support foot is assumed to be flat on the floor. Let's denote by $\boldsymbol{p}_0^f = (x^f, y^f, 0)$ the position of the support foot in the inertial frame, and θ_z^f its rotation around the floor normal axis $\boldsymbol{n}_z = [0,0,1]^T$. This frame assumes that the support foot stays in perfect contact with the floor, and that discrepancies in the CoM state are coming from the robot's passive ankle mechanism. In this frame, the CoM is expressed as

$$\tilde{x}_c = \begin{bmatrix} \boldsymbol{R}(\theta_z^f) & \boldsymbol{p}_0^f \\ 0 & 1 \end{bmatrix} {}^{C}\mathbf{t}_0. \tag{4.11}$$

Its velocity is also expressed in the flat support foot frame. This implicitly embeds the state of the flexibility in the measurement of the CoM state, as long as the real robot's foot sole remains in perfect contact with the floor.

4.3.3 ZMP State

The ZMP can be computed *w.r.t.* to the CoM as

$$x_z = x_c - \frac{\ddot{x}_c}{\eta^2}. \tag{4.12}$$

In practice $\ddot{x}_c$ can be obtained from accelerometer measurements, or by differentiating the CoM position obtained from the above estimate twice, which proves unreliable.

Since HRP-4 is fitted with force-torque sensors under its ankles providing measurements at control-rate (200Hz), a more reliable expression of the ZMP can be obtained, without the need for differentiation. The main interest is in the ZMP expressed in the ground plane, passing through a point $\boldsymbol{p}_0$ and orthogonal to the unit vector $\boldsymbol{n}_z$. Let $(\boldsymbol{f}_0, \boldsymbol{\tau}_0)$ be the total

contact-wrench measured by the left and right foot force-torque sensors, expressed at the point $\boldsymbol{p}_0$. The ZMP is defined as a point Z where the moment of the contact wrench aligns with $\boldsymbol{n}_z$ [Caron et al., 2017], that is $\boldsymbol{n}_z \times \boldsymbol{\tau}_Z = 0$. Consequently

$$-\boldsymbol{n}_z \times (\boldsymbol{p}_z \times \boldsymbol{f}) + \boldsymbol{n}_z \times \boldsymbol{\tau}_0 = 0 \tag{4.13}$$

$$-(\boldsymbol{n}_z \cdot \boldsymbol{f})\boldsymbol{p}_Z + (\boldsymbol{n}_z \cdot \boldsymbol{p}_Z)\boldsymbol{f} + \boldsymbol{n}_z \times \boldsymbol{\tau}_0 = 0 \tag{4.14}$$

With the additional constraint of the ZMP being obtained in the ground plane $\boldsymbol{n}_z \cdot \boldsymbol{p}_Z = 0$, we obtain

$$\boldsymbol{p}_Z = \frac{\boldsymbol{n}_z \times \boldsymbol{\tau}_0}{\boldsymbol{n}_z \cdot \boldsymbol{f}} \tag{4.15}$$

4.3.4 Computing the MPC from its estimated state

Once the estimated state $\tilde{x} = (\tilde{x}_c, \tilde{\dot{x}}_c, \tilde{x}_z)$ is computed, the MPC is initialized with it. This means that the prediction of the evolution of the system via the LIP model (4.1) over the prediction horizon T_h is performed using the estimation as the initial condition. Moreover the cost function (4.3) and the constraints (4.4,4.5,4.6) are built using the estimated robot state.

However, the robot state estimation $\tilde{x}$ is only available at the frequency of the slowest sensor, here SLAM ($30Hz$) or VICON ($100Hz$), both lower than the control rate ($200Hz$). Hence the state is not available at every control iteration. When no measured state is available, the initial LIP state is computed by integration of (4.1). The controls $\dot{x}_z^i$, $i = 1, \ldots, N$, are those computed by the optimization performed by the MPC in the previous iteration. The integration leads to

$$\begin{pmatrix} x_c^{k+1} \\ \dot{x}_c^{k+1} \\ x_z^{k+1} \end{pmatrix} = A \begin{pmatrix} x_c^k \\ \dot{x}_c^k \\ x_z^k \end{pmatrix} + B\dot{x}_z^k \tag{4.16}$$

where the matrices A and B have the form

$$A = \begin{pmatrix} \cosh(\eta\delta) & \frac{\sinh(\eta\delta)}{\eta} & 1-\cosh(\eta\delta) \\ \eta\sinh(\eta\delta) & \cosh(\eta\delta) & -\eta\sinh(\eta\delta) \\ 0 & 0 & 1 \end{pmatrix}, \tag{4.17}$$

$$B = \begin{pmatrix} \delta - \frac{\sinh(\eta\delta)}{\eta} \\ 1-\cosh(\eta\delta) \\ \delta \end{pmatrix}. \tag{4.18}$$

In this way, a reference whole-body control can be provided at every iteration, by propagating the MPC solution, computed either from the estimated robot state, or its integrated internal state.

4.3.5 Choice of footstep

When the loop on the MPC is closed, its solution is computed based on the estimated state $(\tilde{x}_c, \tilde{\dot{x}}_c, \tilde{x}_z)$. Hence, at any time instant the optimal solution found by the optimization, will in general be different from the previous one, even in the absence of a real perturbation in the state (due to measurement noise and uncertainties). This leads to different optimal footstep target solutions at every MPC iteration. When tracked with a whole-body QP controller, these abrupt changes will be smoothened to some extent depending on the task gains, reducing tracking precision of the swing foot trajectory, and leading to shaky motions. To overcome this issue, two possible solutions are considered.

The first consists in adding to the cost function (4.3) a term of the form

$$k_f \left(\left(x_f^j - \bar{x}_f^j \right)^2 + \left(y_f^j - \bar{y}_f^j \right)^2 \right), \tag{4.19}$$

that penalizes the difference between the predicted footstep (x_f^j, y_f^j) and the previously chosen footstep $(\bar{x}_f^j, \bar{y}_f^j)$. However this also affects the other high-level tasks assigned to the robot, e.g. the reference velocity tracking, because the robot minimizes the difference between two footsteps therefore making steps as short as possible.

The second option relies on the fact that the solution found by the MPC is the optimal one, but it's not the unique solution that satisfies the constraints. Therefore, if the constraints are still satisfied by keeping the previously found footsteps with the new ZMP trajectory, it is chosen to keep it, even if it is not the optimal in terms of cost function. In this way we somewhat filter the continuous change of predicted footsteps, and only the latest optimal solution of the MPC is chosen if the previous one becomes invalid because it violates the constraints, which guarantees robot balance.

4.3.6 Quadratic Programming Controller

The desired MPC solution is then tracked by a whole-body quadratic controller (QP), which generates whole-body motion for the real model. A detailed explanation of the QP controller used in this work can be found in [Bouyarmane et Kheddar, 2012; Escande et al., 2013]. Its decision vector is $\mathbf{z} = (\ddot{\mathbf{q}}, \boldsymbol{\lambda})$, where $\ddot{\mathbf{q}}$ gathers the linear and angular acceleration of floating-base coordinates, and the generalized joint velocities. $\boldsymbol{\lambda}$ denotes the vector of conic

coordinates of linearized Coulomb friction cones, such that the contact forces $\mathbf{f}$ are equal to $\mathbf{S}_f\boldsymbol{\lambda}$ with $\mathbf{S}_f$ the span matrix of cone generators. The cost function includes various tasks $\mathcal{T}$ that drive the whole-body state towards a desired configuration. We consider here

- $\mathcal{T}_{CoM}$ tracks the desired CoM position x_c and velocity $\dot{x}_\mathrm{c}$.
- $\mathcal{T}_{Trajectory}$ tracks the desired swing foot target x_f.
- $\mathcal{T}_{CoP}$ tracks the desired ZMP through admittance control, achieved by the embedded stabilizer.
- $\mathcal{T}_{Posture}$ is a reference posture that keeps the robot upright.

The overall QP can be summed up as follows:

$$\begin{aligned}\mathbf{z} = \underset{\mathbf{z}}{\operatorname{argmin}} \sum_{i=1}^{N} w_i \|\mathcal{T}_i(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}})\|^2 + w_\lambda \|\boldsymbol{\lambda}\|^2 \\ \text{subject to:} \\ 1)\ \text{dynamic constraints} \\ 2)\ \text{sustained contact positions} \\ 3)\ \text{joint limits} \\ 4)\ \text{non-desired collision avoidance constraints} \\ 5)\ \text{self-collision avoidance constraints}\end{aligned} \tag{4.20}$$

Here, w_i and w_λ are task weights, and $\mathcal{T}_i(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}})$ denotes the residual of the i^{th} task. The reader is referred to Vaillant et al. [2016] for details on the formulation of all these constraints. The obtained reference acceleration $\ddot{\mathbf{q}}$ is integrated twice to obtain a desired joint configuration $\mathbf{q}$ which is then executed by the actuators *proportional-derivative (PD)* controllers.

4.4 Experiments

The proposed work is based on the MPC formulation presented in Scianca et al. [2016]. Including footstep generation into the formulation enables an intuitive and flexible way to control the locomotion of a humanoid robot. Walking is achieved by specifying a reference velocity. Footsteps, CoM and ZMP trajectories are chosen accordingly. This work has previously been demonstrated for open-loop control of a NAO humanoid robot, including pursuit avoidance scenarios in the presence of obstacles [De Simone et al., 2017]. However, it has never been demonstrated applied to large-scale humanoids such as HRP-4. In this

section, we show that the MPC can be successfully applied to such a robot while validating its closed-loop implementation.

A first experiment validates the walking performance with a simple Cartesian regulator based on SLAM localization that makes the robot walk to specified targets in its environment.

The second experiment validates the closed-loop aspect of the MPC by reacting to perturbations that would otherwise make the robot fall, even in the presence of a stabilizer.

For all experiments, ground-truth floating base position is obtained from a VICON motion capture system, and SLAM estimations are obtained from Asus Xtion's RGB-D frames with *D6DSLAM* [Meilland et Comport, 2013a] software. A video is provided for each experiment respectively.

4.4.1 Stabilization

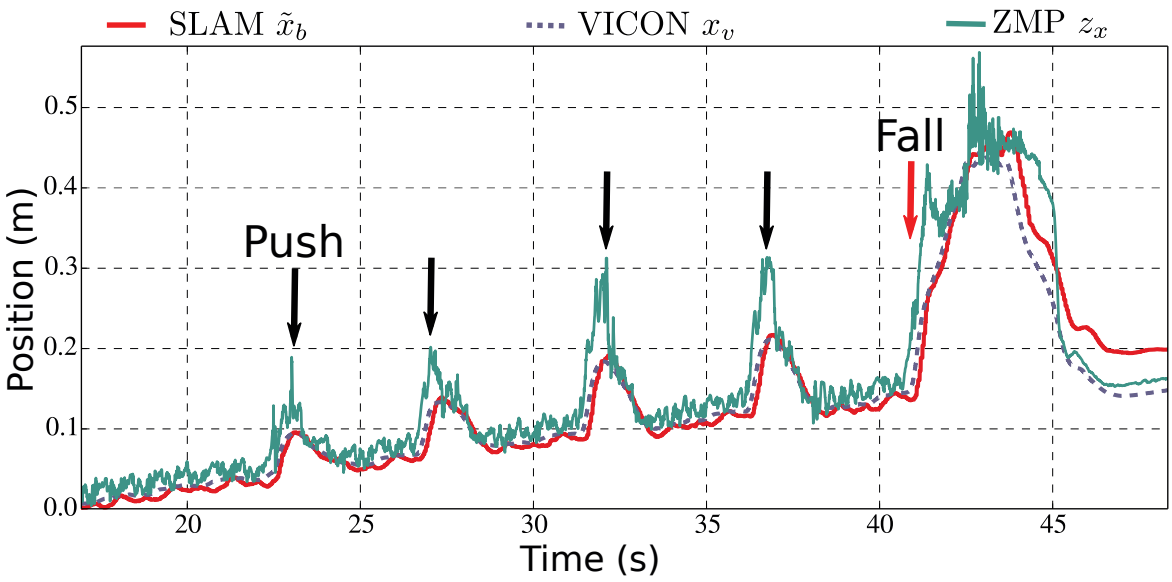


Fig. 4.6 HRP-4 pushed with Kawada stabilizer and an open-loop MPC. The black arrows correspond to pushes that were not strong enough to force the ZMP outside of the foot support area. For the red arrow at time 41s, the robot was pushed sufficiently strongly, and the stabilizer could no longer keep the robot balance. The only way to recover in that situation is to make a step forward, and capture the ZMP in the next footstep, which can be achieved with the proposed closed-loop MPC.

The desired ZMP computed by the MPC is sent to the embedded stabilizer of HRP-4 [Yokoi et al., 2004]. Its role is to compensate deviations from a reference ZMP trajectory by accelerating the torso in the opposite direction. This is achieved by modifying the desired ankle joint reference. In case of a perturbation too large to be recovered without taking a

step, the robot would normally fall. Instead, with our proposed closed-loop MPC the robot takes a step forward, and the stabilizer tracks the new reference ZMP (see Figure 4.10, and the video).

Recently, Caron et al. [2018b] proposed a review of such stabilization methods, and extended it to whole-body admittance control, demonstrated in the context of stair climbing. The reader is referred to this review for further details on common stabilization algorithms.

4.4.2 Drift-free Cartesian space control

An additional benefit to the proposed closed-loop control is its ability to robustly reach target position specified in the environment. To illustrate this, we implemented a simple Cartesian regulator that computes reference velocities (v_x, v_y, ω) for the MPC according to its position and orientation relative to a desired world position (x_d, y_d, θ_d). The floating base pose $(\tilde{x}_b, \tilde{y}_b, \tilde{\theta}_b)$ is chosen as the robot's reference surface, and is estimated with SLAM. Ground-truth measurements are obtained as (x_v, y_v, θ_v) with the VICON tracking system.

$$\begin{pmatrix} v_x \\ v_y \\ \omega \end{pmatrix} = \begin{pmatrix} \lambda_x(x_d - \tilde{x}_b) \\ \lambda_y(y_d - \tilde{y}_b) \\ \lambda_\theta(\theta_d - \tilde{\theta}_b) \end{pmatrix} \tag{4.21}$$

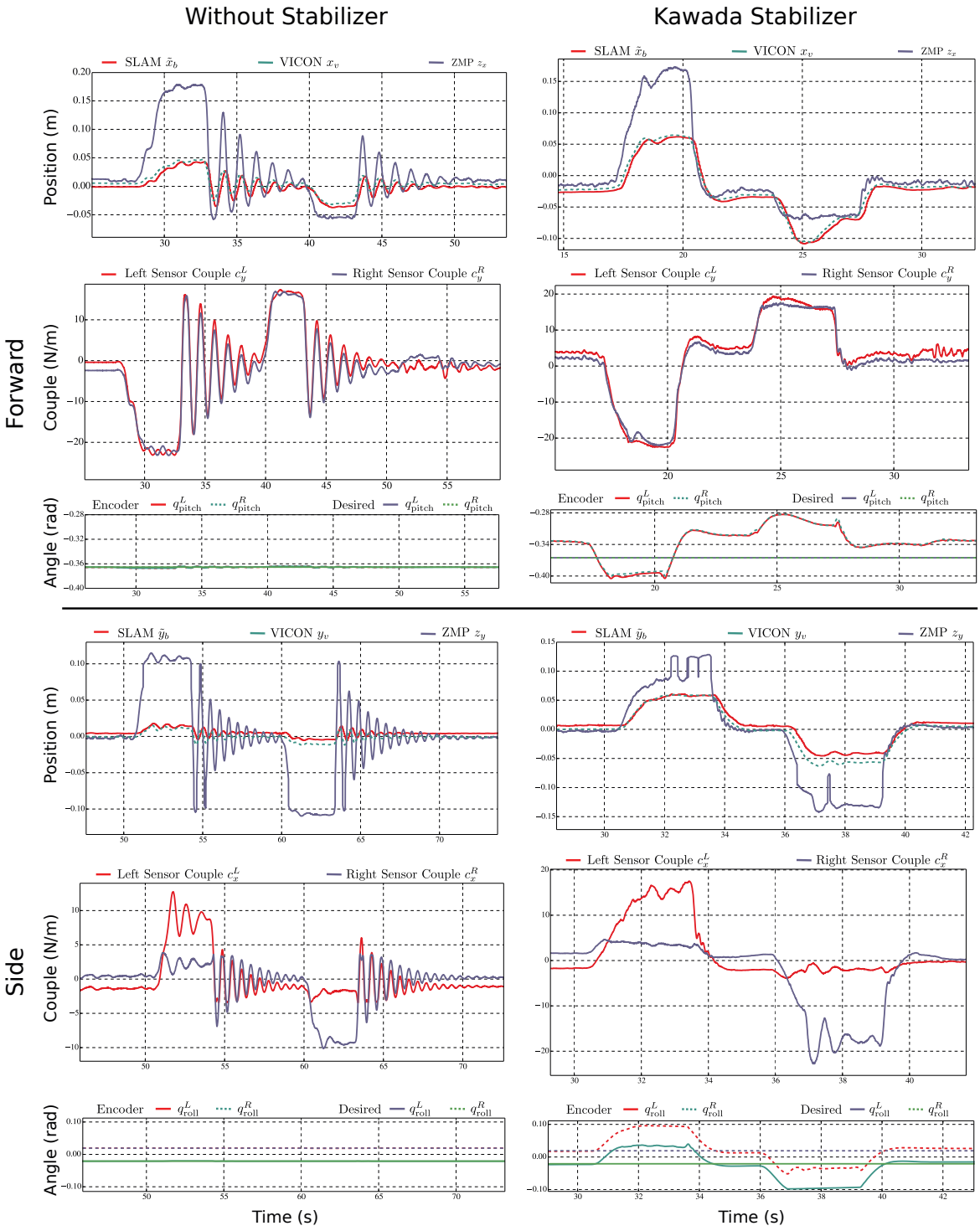


Fig. 4.7 Effect of HRP-4 passive spring-damper mechanism located between it's ankle force-sensor and foot sole. In each case, the robot is manually pushed until the ZMP reaches the edge of the foot, *i.e.* just when the opposite edge of the foot starts lifting, and suddenly released. Floating base displacement is observed by both VICON and SLAM measurements, force-sensor couple measurements are shown along the axis affected by the push, and ZMP is estimated from both feet force-sensors. The measured and desired ankle pitch and roll angles are also shown, and clearly show the effect of Kawada stabilizer.

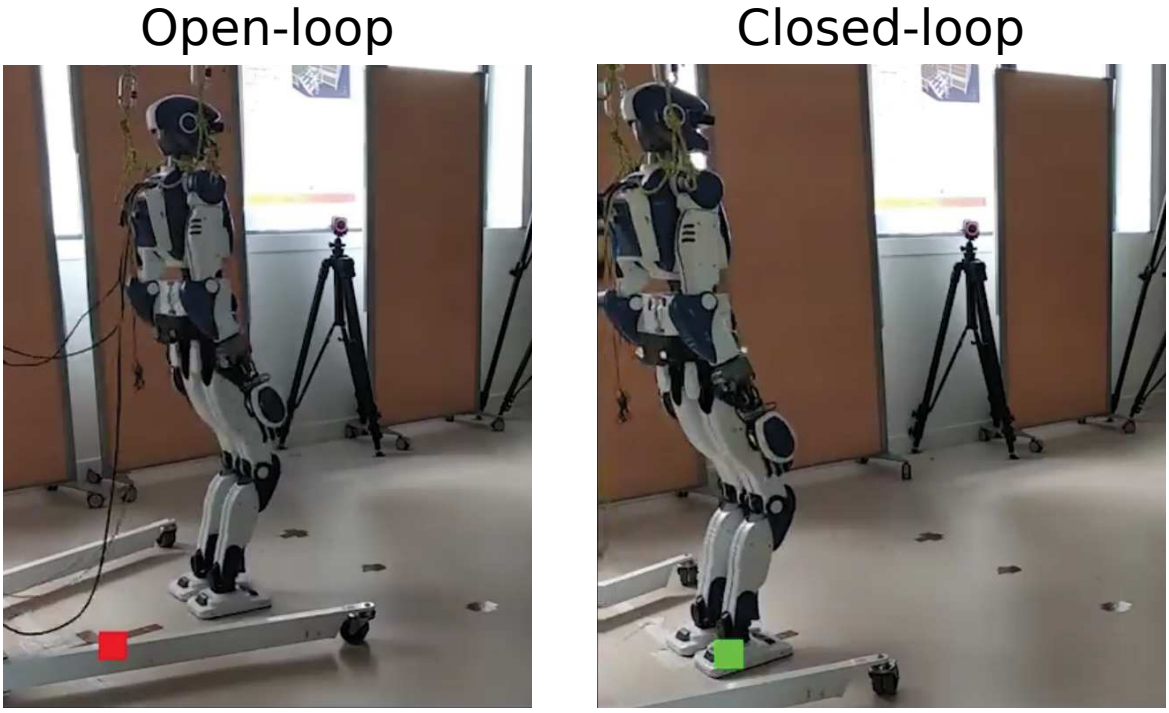


Fig. 4.8 Comparison between closed-loop and open-loop Cartesian regulation after walking forward and to its left twice. Considerable drift is observed in the open-loop case, while the closed-loop controller regains the expected position. This image corresponds to $t = 110$s for open-loop and $t = 200$s for closed-loop in Figure 4.9

Figure 4.9 shows the result of HRP-4 repeatedly walking from an initial position $t_0 = (0,0,0)$ towards a target $t1 = (1,0.5,0)$, then coming back to its initial position before walking towards a second target $t_2 = (1,-0.5,0)$. Without feedback on the regulator, the open-loop walking controller drifts considerably, it accumulates around 10cm of translational error for a walk of 1m, which quickly leads to dramatic errors of localization, reaching about $80cm$ after 200s of walking. No particular effort was made to limit the drift, as closed-loop Cartesian regulation can easily mitigate it. The drift itself is the result of two main effects. The first is that the desired footstep trajectories are not perfectly tracked by the QP controller. To lessen the undesirable impacts at touchdown of the swing-foot, due to unaccounted effects arising from the ankle flexibilities that are not accounted for by the open-loop low-level QP controller, it was experimentally determined that it was best to undershoot the z component than to potentially reach the floor with non zero velocity. This trick could be avoided by more accurate closed-loop end-effector control as proposed in Section 3.3.2. This effect likely causes the robot to tilt forwards slightly (which is easily compensated by the stabilizer), and thus walk further than expected. The second, is due to imperfect foot-floor interactions. The ground in the experiment hall has quite low friction, which can easily result in the feet

slipping due to unaccounted for momentum effects. Not seen on the figure, a considerable drift was observed in rotation, as well as translation.

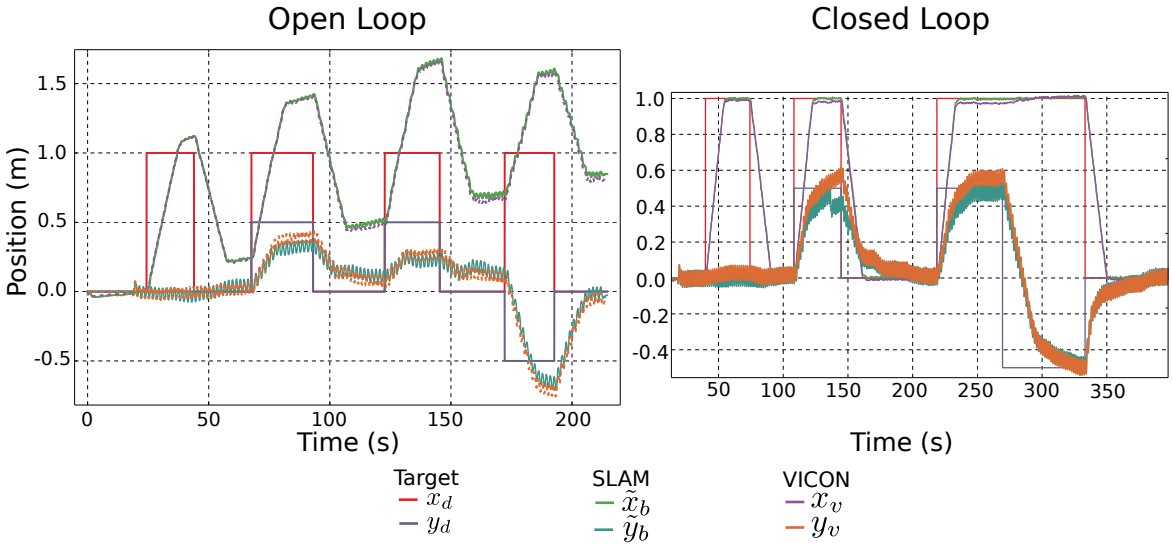


Fig. 4.9 HRP-4 walks to specific places in the environment. A Cartesian-regulator continuously computes MPC reference velocities based on the robot's position *w.r.t.* its target as estimated from SLAM measurements. It's trajectory is validated by both VICON ground-truth data, and visual markers on the floor. Considerable drift up to 70cm along x and 20cm along y is observed with the open-loop Cartesian regulator, while its closed-loop counterpart converges with less than 5cm error.

When using the estimated robot state, the closed-loop regulator is however able to drive the velocity-based MPC to accurately reach each target, without noticeable drift. Notice a sharp jump in the pose of *SLAM* $\tilde{x}_b$ at time $t = 140s$ of the closed-loop experiment, and the corresponding slight overshoot of the target measured by the vicon x_v. The robot having reached a previously explored part of the environment, a loop-closure event was triggered, and the keyframe-graph was optimized to take it into account, thus correcting the accumulated drift of *SLAM* over the first 140s of walking. The subsequent targets are reached accurately.

Note that the Cartesian-regulator formulation in Equation 4.21 is too simple for most practical application. As it stands, weights $(\lambda_x, \lambda_y, \lambda_\theta)$ need to be adaptively increased when the error reaches close to zero to ensure good convergence to the desired position, and reduced when the error is large to avoid walking at maximum velocity all the time. This was mainly used as a simple example, but any regulation scheme that selects MPC velocities to go towards a target can be used instead. For instance, in De Simone et al. [2017], a unicycle model was used to generate more natural looking motions and avoid humans walking towards the robot.

4.4.3 Push reaction

In the second experiment, the robot is stepping in place with a low-weight Cartesian-regulator ($\lambda_x = \lambda_y = \lambda_\theta = 0.2$) when it is pushed from behind. The MPC is computed from the estimated robot state $(\tilde{x}_c, \tilde{\dot{x}}_c, \tilde{x}_z)$, obtained by the combination of SLAM and force-torque sensors. When perturbed, the MPC decides to change it's desired footstep, and computes a new corresponding ZMP velocity trajectory. This behaviour emerges as a result of the optimization: in order to find an optimal solution that satisfies the constraints (bringing the future ZMP trajectory back inside the support foot polygon while respecting kinematic feasibility), the MPC has to move the ZMP forward thus performing a step.

Figure 4.10 shows this behaviour on a real perturbation. As can be seen from the first plot, around time 194.7s, the perturbation becomes large enough for the ZMP to exit the balance constraint area (defined as a sub-rectangle of the real foot) with sufficient velocity, and the MPC has no choice but to alter its desired footstep. The shaky behaviour of the footstep solution can be seen on the MPC footstep plot, as the experiment was performed without penalizing any change of footstep. Note that here no attempt is made at explicitly smoothing the desired footstep trajectory with strong damping, as suggested in Feng et al. [2016]. The swing-foot trajectory task in the whole-body QP controller somewhat smoothes jerks in the footstep trajectory, but as the error remains tracked in position with high stiffness to ensure that the footstep trajectory is ultimately followed, the actual foot motion remains shaky. We also noticed the predicted effect of this jerkiness onto the CoM state, causing in extreme cases self-amplifying perturbations (rarely observed to be significant enough for failure). Furthermore as they suggested, we also noticed that reactive footstep allowed for less accurate control of the CoM and ZMP, as mistakes are quickly recovered at the next step. The estimates of the floating base state obtained from *SLAM* proved sufficiently robust to achieve closed-loop MPC control, although significantly less shaky footstep trajectories were observed when performing feedback from VICON tracking data instead.

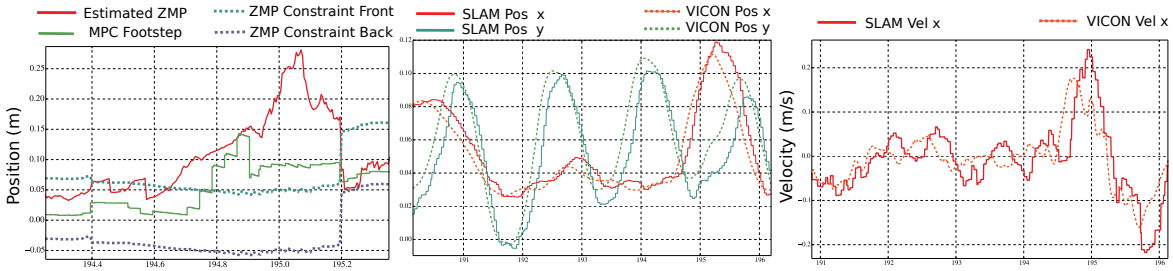


Fig. 4.10 A strong push forward causes the MPC to step forward to prevent loss of balance. Left - the estimated ZMP from force-sensor measurements is leaving the stability constraint area at time 194.7s, the MPC plans a step forward to compensate. Middle - Estimation of the floating base position from SLAM and VICON measurements along the x and y direction. Right - Estimation of the floating base velocity.

4.5 Towards walking in uncontrolled environments

The experiments presented thus-far do not exploit the map of *SLAM*, as its use has not yet been implemented into the closed-loop MPC controller. Yet, with the MPC formulation presented here, two obvious ways of doing so present themselves. The first is to use the registration scheme presented in Chapter 3 along with the closed-loop Cartesian regulator to walk towards an object in the environment. This could be used in-stead of the walking pattern generator used and run open-loop for the experiments presented in this chapter. To ensure collision-free path with the environment, more elaborate path planning trajectories are of course required. This leads to the second opportunity to exploit the map. The MPC formulation can trivially be extended to add obstacle-avoidance constraints, as proposed in De Simone et al. [2017], further restricting the kinematic-reachability constraint of Equation 4.6 with additional inequality constraints. The obstacles can be easily segmented from the map by exploiting the simple assumption that the robot is walking on flat floor, which is a prerequisite of this MPC-formulation. In that case, any point in the map above a small threshold above the ground plane can be considered as a potential obstacle. A simple clustering on those points, followed by the estimation of their convex hull (or simply its projection on the ground plane) provides all the information needed for preventing footsteps from colliding with the environment. Unfortunately, neither has been implemented here, and future work will take it into account.

These improvements concern the MPC presentation formulated here. However, only the implementation of more elaborate methods is preventing us from applying this to more complex environments. In Caron et Kheddar [2017], and his subsequent relative work, an MPC formulation was presented to walk over rough terrain of any shape (assuming

planar contacts and kinematic feasibility). This particular work hasn't been tried outside of simulation yet, but a similar method was used and demonstrated in recent work in stabilizing the HRP-4 [Caron et al., 2018b] in the case of stair climbing. In that experiment, and for walking over any kind of rough terrain, the missing block is clearly the estimation of the contact surfaces in the map, as was done in Fallon et al. [2015]. One could go further, by exploiting Dense planar *SLAM* [Salas-Moreno et al., 2014] (or similar implementations) to obtain potential footstep in real-time, along with the solver formulation proposed in Caron et al. [2018a] to automatically select the best footstep to make. With a closed-loop formulation of such an MPC formulation, it becomes conceivable to walk over rough terrain, such as the brick field at the DRC challenge without stopping for planning any footstep, and with the ability to react to perturbations by changing the footsteps. The only missing ingredient here is its implementation.

4.6 Conclusion

In this work a closed-loop formulation of the intrinsically stable MPC for humanoid gait generation Scianca et al. [2016] has been presented, where the robot CoM position and velocity are estimated with visual SLAM, while the ZMP position is computed from force/torque sensor measurements. The closed-loop implementation of the MPC allows the robot to react to external disturbances through stepping, enhancing the overall robustness of the control scheme. Moreover, the visual localization of the robot in the environment, allows it to navigate with precision to specific locations, recovering from the drift generated by uncertainties and unmodelled interactions with the environment (e.g. foot slipping).

However, the capabilities of *SLAM*, haven't been fully exploited, and in particular its map has remained unused. With the MPC formulation presented here, obstacle avoidance could easily be integrated by segmenting objects above the floor and adding them as inequality constraints in the MPC. With a better path-planning algorithm, one could then walk on flat floor in between obstacles.

An even more complete and interesting extension to this work would be to apply its main ideas to an efficient MPC formulation designed to walk over rough terrain such as Caron et al. [2018a] along with an automatic detection of flat surfaces as recent example with dense planar SLAM has shown possible to walk over any rough terrain without any manual planning, and automatically select the best footstep, even under perturbations.

Conclusion

In this thesis, we have outlined the importance of estimating both the state of the robot and its environment, and exploiting it for both planning and control of a humanoid robot behaviour. In particular, we showed that recent developments in dense visual Simultaneous Localization and Mapping have enabled such an endeavour, by providing both the location of the robot's vision sensor, and a three dimensional map of the explored parts of its surroundings.

To precisely localize any surface on the robot body within the map reconstructed with *SLAM*, it is necessary to have an accurate model of the robot's kinematic tree, including the pose of the vision sensor *w.r.t.* to that body. A novel method named *Eye-Robot* calibration, extending the classic Hand-Eye formulation to whole-body calibration was proposed. It relies solely on sensors available on any humanoid robot, that is joint encoders and a vision sensor (here RGB-D) suitable for tracking using dense pose estimation algorithms. The motion of the vision sensor tracked with dense visual *SLAM*, and that of the joint encoders are jointly considered to estimate the kinematic model. The original intent was to provide whole-body online calibration of any robot. However, during the development of the method, it was realized that information considered about the robot motion does not always provide full observability of these parameters. Driving on previous work considering the observability of the Hand-Eye calibration using screw-congruence theory, the observability of the Eye-Robot calibration was analysed, and used to obtain a solution along the observable directions, while retaining the manufacturer's calibration along the others. Robot-Eye calibration of the HRP-4 upper body was demonstrated (including its extrinsic camera calibration), and dense Hand-Eye calibration on calibrating MOCAP markers *w.r.t.* to either an RGB-D sensor, or the robot body.

A method to exploit the map and robot localization to successfully accomplish multi-contact plans generated offline *w.r.t.* to CAD models was proposed. The models used for planning are located within the reconstructed map of the real environment by a proposed variant of the Iterative Closest Point algorithm. The proposed ICP formulation is robust to change of scales between the model and the map, and Iteratively Re-weighted Least-Squares with robust M-Estimation is used to provide a robust estimation in the presence of outliers.

Furthermore, it was realized that unavoidable robot motion within the RGB-D sensor field-of-view contradicts the rigid-scene assumption of the dense visual *SLAM* algorithms, leading to tracking inaccuracies and failures. For lack of a suitable non-rigid *SLAM* algorithm, we proposed to modify the photometric and geometric tracking function to discard the pixels corresponding to the visible limbs of the robot as outliers. This is achieved by rendering the robot's CAD model from the RGB-D sensor's viewpoint onto a binary mask, subsequently used as an outlier filter in the cost function. The pose estimated from *SLAM* was then used to formulate a closed-loop task to our quadratic programming controller that aligns any robot surface with the registered target surfaces provided by the multi-contact plan. The approach was successfully demonstrated on experiments with an HRP-2Kai, tasked with executing several multi-contact plans, such as grasping a valve, a car wheel, *etc*. Starting from an initial uncontrolled configuration, the robot was able to robustly achieve the plans, by relying on a walking controller to reach the desired initial configuration when necessary, and robustly establishing contacts even under voluntary perturbations (push).

Finally, it was shown that *SLAM* can successfully be used, in conjunction with force-torque sensors, to estimate the position and velocity of the robot's center-of-mass and zero-moment point, with sufficient accuracy to enable the formulation of a closed-loop MPC controller. Experiments on the HRP-4 humanoid show that it can successfully recover from heavy perturbation by computing suitable footsteps and ZMP trajectories, where it would otherwise have fallen without stepping. With the addition of a simple closed-loop Cartesian controller, the ability to precisely walk to any Cartesian location within the map is demonstrated.

The Dream

When starting this thesis, it was already clear that dense visual *SLAM* had the potential to revolutionize the control of humanoid robots and to finally move away from the controlled laboratory experiments towards evolving in uncontrolled and a-priori unknown environments. First steps were taken in that direction both in this thesis and related works, but so far, only the surface of what can ultimately be achieved has been scratched. Considerable advancements have been made in both humanoid robotics and *SLAM* in the four years since this thesis begun. Walking has evolved from flat floors, to any flat surface in 3D, gravels, rubble, staircases, and even snow! Whole-body multi-contact planning that was initially restricted to laborious and slow offline planning is starting to reach online solutions, which will, in the near future, enable online planning in any environment. And nothing prevents it from

exploiting estimated map information in stead of the classical model-based approach that has been adopted so far.

On its side, *SLAM* is undergoing a major shift towards higher level reasoning about map information. Algorithms reasoning at the level of surfaces (Planar SLAM) and objects (SLAM++) have been developed, and a lot of recent work has focused on semantic representations of the map. Semantic maps will provide the missing link to full control a robot in unknown environment. They will provide all the information needed to inform multi-contact planning. Where can the robot establish contact with its surroundings? Which object can it interact with? What can it do with the objects?

With these challenges solved, work on humanoids will shift from the tedious control aspects with humans-in-the-loop towards higher level reasoning. Instead of asking *how*, we'll soon find ourselves asking *what*? What task do we want the robot to accomplish? Only then will it be able to explore and understand it's environment, navigate through it, and accomplish complex tasks without human intervention.

Appendix A

Non-Linear Least Square Minimization on the Lie Algebra

A.1 Non-Linear Least Square Minimization

In computer vision or robotics, it is common to require fitting a model to a set of measured data. In the case of least-square problems, this dataset consists of N data pairs $\mathbb{S} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1..N}$ obtained by observation. The aim is to find the parameters $\xi \in \mathbb{R}^m$ of a model $f(\mathbf{x}, \xi)$ that best fit the data. This fit is measured by its residual that represents the difference between the actual value and a value predicted by the model.

$$\mathbf{r}_i = \mathbf{y}_i - f(\mathbf{x}_i, \xi)$$

The state variable ξ is obtained by minimizing the sum of squared residuals $S = \sum_{i=1}^{n} \mathbf{r}_i^2$. The sum of residuals should decrease when the estimated state improves the data fit, and increase otherwise.

$$\hat{\xi} = \underset{\xi}{\operatorname{argmin}} \sum_{i=1}^{n} \mathbf{r}_i^2$$

In some cases a closed-form solution exists to such a non-linear least-square problem, but this is in general not the case. When no such closed-form solution exist, it is common to rely on numerical iterative algorithms. One of the most common method is to use Gauss-Newton algorithm. The concept is to linearise the state around its best estimate $\hat{\xi}$, and iteratively improve that estimate as

$$\xi^{k+1} = \xi^k + \hat{\xi} \tag{A.1}$$

The solution as iteration k is incremented by the shift vector $\hat{\xi}$. Each residual is related to the state variables

$$\mathbf{e}_i = \mathbf{J}_i \hat{\xi}$$

where $\mathbf{J}_i = \frac{\partial \varepsilon_i}{\partial \xi}$ is the error Jacobian. That is each residual contributes to directing the state in a direction that reduces the total residual error. The full system is solved by stacking the Jacobian and errors for each residual and inverting the system

$$\hat{\xi} = \mathbf{J}^{+}\mathbf{e}$$

where $\mathbf{J}^{+} = (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T$ is the pseudo-inverse of the stacked Jacobian $\mathbf{J} = [\mathbf{J}_1, \ldots, \mathbf{J}_N]^T$ and $\mathbf{e} = [\mathbf{e}_1, \ldots, \mathbf{e}_N]^T$.

The update equation A.1 is computed until convergence of the algorithm. The definition of convergence depends on the application. Most commonly, the algorithm is considered to converge when the the sum of squared residuals is below a threshold, or when the rate of convergence becomes too small. Note that this method is highly subject to local minima, and global convergence cannot be guaranteed.

Ordinary least-square is optimal under some very restrictive conditions. It assumes a constant variance in the errors: outliers in the dataset have a tendency to pull the least-square solution towards them since their residual error is much larger than that of the valid measurements. Fortunately, least-square problems can be reformulated using robust statics, and solved with the iteratively re-weighted least-square method.

A.1.1 Solving M-Estimator with Iteratively Reweighted Least-Square

In 1964, Huber Huber et al. [1964] proposed generalizing maximum likelihood estimation to the minimization of $\sum_{i=1}^{n} \rho(\mathbf{r}_i(\xi))$. The solutions to

$$\hat{\xi}_{\mathbf{M}} = \underset{\xi_M}{\operatorname{argmin}} \sum_{i=1}^{n} \rho\left(\mathbf{r}_i(\xi)\right)$$

are called M-Estimators. $\rho(.)$ is a function chosen to provide the estimator desirable properties in term of bias and efficiency when the data fits the assumed distribution. Note that assuming normally distributed residuals, $\rho(z) = \frac{1}{2}z^2$ results in the ordinary least-square estimate. Some M-Estimators are influenced by the scale of the residuals, thus we formulate

a scale-invariant version of the M-Estimator. Let $\tilde{r}(\xi) = \frac{\mathbf{r}_i(\xi)}{\tau}$

$$\hat{\xi}_{\mathbf{M}} = \underset{\xi_M}{\operatorname{argmin}} \sum_{i=1}^{n} \rho\left(\tilde{\mathbf{r}}_i(\xi)\right) \tag{A.2}$$

where τ is a measured of the scale

$$\tau = \frac{\text{median}(|\mathbf{r}_i - \text{median}(\mathbf{r})|)}{0.6745}$$

It is often simpler to differentiate *w.r.t.* to ξ and solve for the root of the derivative. When that differentiation is possible, the M-Estimator is said to be of ψ-type, otherwise it is said to be of ρ-type.

ψ-type M-Estimator are used in most practical case. If ρ is differentiable, the computation of $\hat{\xi}$ is much easier. Set

$$\frac{\partial \rho}{\partial \xi_j} = 0$$

for each $j = 1, \ldots, m$, resulting in a set of m non-linear equations

$$\sum_{i=1}^{n} \frac{\partial \tilde{\mathbf{r}}_i(\xi)}{\partial \xi} \psi(\tilde{\mathbf{r}}_\mathbf{i})$$

where $\psi = \rho' \circ \tilde{\mathbf{r}}_i(\xi)$, called the influence function. Table A.1 references weights obtained for the most commonly used M-Estimator influence functions. Taking from the literature, the proportionality factors of $b = 4.6851$, $c = 2.3849$ and $a = 1.2107$ ensures 95% of efficiency in the case of Gaussian noise. We refer the reader to Comport [2005] for a detailed explanation. For most choices of $\rho(.)$ or $\psi(.)$, no close-form solution exists. In that case, the solution can be iteratively determined using the iteratively re-weighted least-square algorithm.

Iteratively Reweighted Least-Mean Square (IRLS) is used to estimate the weighted least-square solution to equation A.2 as

$$\hat{\xi}^{k+1} = \left(\mathbf{J}^T [\mathbf{W}^{-1}]^k \mathbf{J}\right)^{-1} \mathbf{J}^T [\mathbf{W}^{-1}]^k \mathbf{r}_i \tag{A.3}$$

where $\mathbf{W} = diag(\omega_1, \ldots, \omega_N)$ is a diagonal matrix of weight coefficients. Coefficients for common M-Estimators are provided in Table A.1.

Solving for the M-Estimator instead of the ordinary least-square provides robustness to outliers, which proves to be important for real-applications, especially when considering

Function	Huber	Tukey	Cauchy
$\rho(u)$	$\frac{1}{2}u^2, \quad \lvert u\rvert \leq a$ $\frac{a}{2}(2\lvert u\rvert - a), \quad \lvert u\rvert > a$	$\frac{b^2}{6}(1-(1-(\frac{u}{b})^2)^3), \quad \lvert u\rvert \leq b$ $\frac{b^2}{6}, \quad \lvert u\rvert > b$	$\frac{c^2}{2}\log\left(1+\left(\frac{u}{c}\right)^2\right)$
$\psi(u)$	$u, \quad \lvert u\rvert \leq a$ $a\frac{u}{\lvert u\rvert}, \quad \lvert u\rvert > a$	$-\frac{u}{\tau}(1-(\frac{u}{b})^2)^2 \quad \lvert u\rvert \leq b$ $0 \quad \lvert u\rvert > b$	$\frac{u}{1+(\frac{u}{c})^2}$
$\omega(u)$	$1, \quad \lvert u\rvert \leq a$ $a\frac{1}{u}, \quad \lvert u\rvert > a$	$(1-(\frac{u}{b})^2)^2, \quad \lvert u\rvert \leq b$ $0, \quad \lvert u\rvert > b$	$\frac{1}{1+\left(\frac{u}{c}\right)^2}$

Table A.1 Various robust cost functions and their corresponding influence and weight functions. Functions $\rho(.)$ corresponds to the robust probability distribution, $psi(.)$ is the partial derivative of $\rho(.)$ *w.r.t.* the parameters r, and $\omega(.)$ is the corresponding weight factor for an IRLS implementation.

data measured from vision sensors that can easily break assumptions of the modelled cost function.

A.2 Optimization on the special euclidean manifold

A common problem consists of finding a rigid-body transformation *w.r.t.* to observed data. SLAM algorithm use visual cues, along with a camera projection model to track the 6D-pose of the camera; the Hand-Eye calibration method proposed in Chapter 2 uses robot and sensor motions to obtain the transformation between the kinematic chain's joints. In recent years, Lie algebra, and in particular its special euclidean group $\mathbb{SE}(3)$ has become a de-facto standard for formulating least-mean square problems involving rigid-body transformations. In this section we will briefly summarize this representation, and its advantages. We refer the reader to the excellent book by Ma et al. [2003] for an in-depth analysis.

A.2.1 Special Euclidean Group

An object is considered rigid if and only if the distance between any points is constant. Two points with coordinates $\mathbf{x}(t) \in \mathbb{R}^3$ and $\mathbf{y}(t) \in \mathbb{R}^3$ must satisfy:

$$\exists c \in \mathbb{R}, \forall t \in \mathbb{R} \qquad ||\mathbf{x}(t) - \mathbf{y}(t)|| = c \tag{A.4}$$

A rigid body motion is a family of transformation that describes how the coordinates of every point on the object changes as a function of time. As the distances between all points of a rigid body object are constant (Eq. A.4), a rigid body motion can be defined with respect to one unique point lying on the object.

A convenient way of representing rigid body motions is *Special Euclidean group*, belonging to the broader category of Lie groups. Denoted by $\mathbb{SE}(3)$, it can be represented by the set of homogeneous matrices $\overline{\mathbf{T}}$ composed of a rotation and translation, represented respectively by $\mathbf{R} \in \mathbb{SO}(3)$ and $\mathbf{t} \in \mathbb{R}^3$. All rigid body motions can be described as an element of the following set:

$$\begin{aligned} \mathbb{SE}(3) &= \left\{ \overline{\mathbf{T}} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix} \;\middle|\; \mathbf{R} \in \mathbb{SO}(3), \mathbf{t} \in \mathbb{R}^{3\times 1} \right\} \subset \mathbb{R}^{4\times 4} \\ \mathbb{SO}(3) &= \left\{ \mathbf{R} \in \mathbb{R}^{3x3} \;\middle|\; \mathbf{R}^T\mathbf{R} = \mathbf{I}_3, \det(\mathbf{R}) = 1 \right\} \end{aligned} \tag{A.5}$$

where $\overline{\mathbf{T}}(t)$ is called the homogeneous matrix. Homogeneous coordinates unify computations of rotation and translation in a single matrix form. The coordinates of a point $\mathbf{x_0} \in \mathbb{R}^3$ expressed in a frame (*e.g.* world frame) can be expressed relative to a current frame (camera) at time t as

$$\overline{\mathbf{x}}(t) = \overline{\mathbf{T}}(t)\overline{\mathbf{x}_0} = \mathbf{R}(t)\mathbf{x}_0 + \mathbf{t}(t) \tag{A.6}$$

This representation, while intuitive isn't optimal. Indeed, in matrix form, there are 12 free parameters (9 for the rotation matrix, 3 for the translation), while only 6 are needed to fully represent a 3D rigid body motion (3 for rotation, 3 for translation). To respect the structure of the special euclidean group, these parameters cannot be chosen arbitrarily, and the additional computational complexity of optimizing 12 parameters instead of 6 is non-negligible. Any element of the special euclidean group $\mathbb{SE}(3)$ may be represented in its associated algebra $\mathfrak{se}(3)$, which provides such a compact minimal representation.

A.2.2 Special euclidean algebra

In $\mathbb{SO}(3)$ the rotation matrix is represented with 9 non-free parameters: they must respect the constraint that $\mathbf{R}^T\mathbf{R} = \mathbf{I}_3$. This imposes 6 independent constraints on these 9 parameters, hence the dimension of the rotation space is 3, and 6 parameters are in fact redundant.

The space of all skew-symmetric matrices, also called tangent space at the identity of the matrix group $\mathbb{SO}(3)$ achieves such a minimal representation

$$\mathfrak{so}(3) = \left\{ [\omega]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \quad | \quad \omega \in \mathbb{R}^3 \right\} \tag{A.7}$$

An element of $\omega \in \mathfrak{so}(3)$ is related to its rotation matrix by

$$\mathbf{R}(t) = e^{[\omega t]_x t} = \mathbf{I}_3 + [\omega t]_\times + \frac{([\omega t]_\times)^2}{2!} + \ldots + \frac{([\omega t]_\times)^n}{n!} \tag{A.8}$$

The special euclidean group also has an associated Lie algebra where $se(3)$ is called the tangent space of the matrix group $\mathbb{SE}(3)$, of the form

$$\mathfrak{se}(3) = \{ [\xi]_\wedge = \begin{bmatrix} [\omega]_\times & \upsilon \\ 0 & 0 \end{bmatrix} | \omega \in \mathfrak{so}(3), \upsilon \in \mathbb{R}^{3\times 1} \} \tag{A.9}$$

An element of $\xi \in \mathfrak{se}(3)$ is related to its transformation matrix by

$$\mathbf{T}(t) = e^{[\xi t]_\wedge} = \begin{bmatrix} e^{[\omega t]_\times} & (I - e^{[\omega t]_\times})[\omega]_\times \upsilon + \omega\omega^T \upsilon t \\ 0 & 1 \end{bmatrix} \tag{A.10}$$

The computation of this exponential map is usually achieved with the Rodrigues' formula.

A.2.3 Derivatives on the special euclidean algebra

Computing jacobians of cost-function dependent on $\mathfrak{se}(3)$ element often requires obtaining the derivative of the exponential map $\dfrac{\partial e^{[\xi]_\wedge}}{\partial \xi}$. For an arbitrary value of ξ, an analytical formula is hard to obtain. Fortunately deriving the exponential map around the identity leads to a simple Jacobian composed of the generators of the lie algebra.

$$\mathbf{J}_0 = \left.\frac{\partial e^{\xi}}{\partial \xi}\right|_{\xi=0} = \begin{array}{c} \\ r_{11} \\ r_{12} \\ r_{13} \\ t_x \\ r_{21} \\ r_{22} \\ r_{23} \\ t_y \\ r_{31} \\ r_{32} \\ r_{33} \\ t_z \end{array} \begin{array}{c} \begin{array}{cccccc} v_x & v_y & v_z & \omega_x & \omega_y & \omega_z \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array} \tag{A.11}$$

In the context of iterative least-square minimization, it is generally possible to express the cost function such that the Jacobian is expressed at identity. Consider a cost-function $f(\mathbf{T})$ where $\mathbf{T} \in \mathbb{SE}(3)$. At iteration k of the minimization, it can be expressed as either $f(\mathbf{T}^k) = f(\mathbf{T}^{k-1} e^{[\xi]_\wedge})$, or $f(\mathbf{T}^k) = f(e^{[\xi]_\wedge} \mathbf{T}^{k-1})$, where ξ is the optimization shift-variable close to the identity-vector of the algebra. Choosing pre or pos-multiplication depends on the cost-function, and may provide efficiency advantages by allowing to pre-compute the Jacobian in some cases. A detailed example of this principle is described with the proposed Hand-Eye cost-function (Equation 2.2) and its analytical jacobian computation (Equation 2.5). Robot-Eye formulation (Equation 2.7), *D6DSLAM* cost function (Equation 1.23), and other cost-functions presented in this thesis also rely on this principle.

Appendix B

Iterative Closest Point

The Iterative Closest Point (ICP) method was first introduced by Besl et McKay [1992] in 1992, and has since become a well-known and widely used algorithm for registering two sets of closely related cloud of points (in 2D or 3D) under Euclidean transformations. The concept is simple, the algorithm iteratively finds the transformation that minimizes a distance between both point sets. To do so, it alternates between two phases (i) finding the closest points between a reference cloud $\mathscr{P}^*$ and a registration candidate $\mathscr{P}$ and (ii) solving a least-square minimization problem that finds an euclidean transformation $\widehat{\mathbf{T}} \in \mathbb{SE}(3)$ minimizing the distance between both clouds.

$$\hat{\boldsymbol{\xi}} = \underset{\boldsymbol{\xi}}{\operatorname{argmin}} \quad f(\mathscr{P}*, \mathscr{P}, \boldsymbol{\xi}) \tag{B.1}$$

where f is an error function whose value is low when the pointclouds are correctly registered. Some common error functions will be detailed further in the following sections.

In the literature, several choices of representation can be found for the minimization vector $\boldsymbol{\xi}$ corresponding to the pose $\widehat{\mathbf{T}}$. In Besl et McKay [1992], a quaternion and translation vector are used. In this thesis, we choose to use a special euclidean algebra as the minimal representation for the pose between the clouds. All methods presented in this section are provided as an open-source C++ library[1].

The ICP algorithm can be defined as follow:

[1] https://github.com/arntanguy/icp

Algorithm 1 Iterative Closest Point Algorithm

INPUT: reference cloud $\mathscr{P}^*$, current cloud $\mathscr{P}$, initial guess $\hat{\mathbf{T}}$

1: Find the closest points in the reference cloud $\mathscr{P}^*$ for each point in the current cloud $\mathscr{P}$. This is typically done by building a kd-tree composed of the points of $\mathscr{P}^*$, and searching for the nearest neighbour of each point of $\mathscr{P}$ in it.
2: Find the transformation $\boldsymbol{\xi}$ that minimizes the error f between the point matches (eq. B.1).
3: Transform all points of the current cloud using the obtained transformation $\boldsymbol{\xi}$: $\forall \mathbf{P} \in \mathscr{P}^*, \quad \mathbf{P} = \mathbf{T}(\boldsymbol{\xi})\hat{\mathbf{T}}\mathbf{P}$.
4: Iterate until the convergence criterion is reached (error function close to zero, variation in translation and rotation too small...).

B.1 Common ICP Formulations

Point to Point ICP One common error function for ICP is to use the euclidean distance between each matching pair of points as the error function. First, we define an error vector for a pair of matching points $\mathbf{P}_i \in \mathbb{R}^4$ and $\mathbf{P}_i^* \in \mathbb{R}^4$ and $\mathbf{e_i} = [e_x, e_y, e_z]^T \in \mathbb{R}^3$ as

$$\begin{aligned} f: \quad (\mathbb{R}^4, \mathbb{R}^4, \mathfrak{se}(3)) &\to \mathbb{R}^3 \\ (\mathbf{P_i}, \mathbf{P_i}^*, \boldsymbol{\xi}) &\mapsto \mathbf{e}_i = \mathbf{P_i} - \mathbf{T}(\boldsymbol{\xi})\hat{\mathbf{T}}\mathbf{P_i}^* \end{aligned} \tag{B.2}$$

The error Jacobian can be expressed as

$$\mathbf{J_i} = \frac{\partial \mathbf{e_i}}{\partial \boldsymbol{\xi}} = \frac{\partial e^{\boldsymbol{\xi}}}{\partial \boldsymbol{\xi}} \hat{\mathbf{T}}\mathbf{P_i}^* = [\mathbf{I}_3 \quad -[\hat{\mathbf{T}}\mathbf{P_i}^*]_\times] \in \mathbb{R}^{3\times 6} \tag{B.3}$$

$$\mathbf{e_i} = \mathbf{J_i}\boldsymbol{\xi} \in \mathbb{R}^{3\times 1} \tag{B.4}$$

Solving the equation for $\boldsymbol{\xi}$:

$$\boldsymbol{\xi} = \begin{bmatrix} \upsilon \\ \omega \end{bmatrix} = -\mathbf{J_i}^+ \mathbf{e_i} \in \mathbb{R}^{6\times 1} \tag{B.5}$$

, where $\mathbf{J}^+ = (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J} \in \mathbb{R}^{6\times 6}$ is the pseudo-inverse of the Jacobian. Finally, the pose is updated as

$$\hat{\mathbf{T}} = \mathbf{e_i}^{\xi}\hat{\mathbf{T}} \tag{B.6}$$

$\hat{\mathbf{T}}$ is re-computed in each iteration of the equation B.2 until convergence is reached (variation of the error low, small change in rotational and translational error...).

All of the above method has been derived for optimization of the pose given a single point. In practice, we want to perform the optimization algorithm on pointclouds. Doing so is simple, all we need to do is stack up the Jacobian and errors for each point into a big matrix and vector, and perform the minimization step (eq. B.5) using these. Consider two pointclouds $\mathscr{P}^*$ and $\mathscr{P}$ defined as

$$\mathscr{P} = \{\mathbf{P_1}, \mathbf{P_2}, \cdots, \mathbf{P_n}\} \in \mathbb{R}^{4\times n} \tag{B.7}$$

$$\mathscr{P}^* = \{\mathbf{P_1}^*, \mathbf{P_2}^*, \cdots, \mathbf{P_{n^*}}^*\} \in \mathbb{R}^{4\times n^*} \tag{B.8}$$

We then find all matching pairs of points between the two pointclouds (by using a nearest neighbour search on a kd-tree for instance). The correspondence set can be expressed as

$$\mathscr{P}_m = \{(\mathbf{p_1^*}, \mathbf{P}_{\phi(\mathbf{1})}), \cdots, (\mathbf{p_{n^*}^*}, \mathbf{P}_{\phi(\mathbf{n})})\} \tag{B.9}$$

where $\phi(i)$ is a function matching the index a point i from the reference cloud into the current cloud. Using the point matches from the set $\mathscr{P}_m$, the stacked jacobian and error is now expressed as

$$\mathbf{J} = \begin{bmatrix} \mathbf{J_1} \\ \mathbf{J_2} \\ \vdots \\ \mathbf{J_{n^*}} \end{bmatrix} \in \mathbb{R}^{3n\times 1} \quad \mathbf{e} = \begin{bmatrix} \mathbf{e_1} \\ \mathbf{e_2} \\ \vdots \\ \mathbf{e_{n^*}} \end{bmatrix} \in \mathbb{R}^{3n\times 6} \tag{B.10}$$

where

$$\mathbf{e_i} = f(\mathbf{p_i^*}, \mathbf{p}_{\phi(\mathbf{i})}, \xi) = \mathbf{p}_{\phi(\mathbf{i})} - \mathbf{T}(\xi)\hat{\mathbf{T}}\mathbf{p_i^*} \tag{B.11}$$

The minimization is then computed using equation B.5 and B.6.

$$\xi = \begin{bmatrix} \upsilon \\ \omega \end{bmatrix} = -\mathbf{J}^+\mathbf{e} \in \mathbb{R}^{6\times 1} \tag{B.12}$$

B.1.1 Point to Plane ICP

While point to point ICP computes the error directly as the euclidean distance between pairs of points, point to plane ICP concerns itself with the distance between a point and the surface tangent to a matching point on the other cloud. In effect, the euclidean distance between

pairs of points is computed, and then projected along the normal of either the reference point or the current point.

Using the current normal

$$\begin{aligned} f: \quad (\mathbb{R}^4, \mathbb{R}^4, \mathbb{R}^4, \mathfrak{se}(3)) &\to \mathbb{R} \\ (\mathbf{n_i}, \mathbf{P_i}, \mathbf{P_i}^*, \boldsymbol{\xi}) &\mapsto e_i = \mathbf{n_i}^T \Pi_0(\mathbf{P}_{\phi(\mathbf{i})} - \mathbf{T}(\xi)\hat{\mathbf{T}}\mathbf{P}_\mathbf{i}^*) \end{aligned} \tag{B.13}$$

Notice that the error is now a scalar instead of a vector as was the case for point to point ICP. This is due to the fact that the error is projected along the normal direction. As the normal is a constant with respect to the pose, computing the new jacobian is simple

$$\mathbf{J}_{\mathbf{i_{p2plane}}} = \mathbf{n_i}^T \mathbf{J}_{\mathbf{i_{p2point}}} = \begin{bmatrix} n_{i_x} \\ n_{i_y} \\ n_{i_z} \\ y_{e_i} n_{i_z} - z_{e_i} n_{i_y} \\ z_{e_i} n_{i_x} - x_{e_i} n_{i_z} \\ y_{e_i} n_{i_y} - y_{e_i} n_{i_x} \end{bmatrix}^T \in \mathbb{R}^{1\times 6} \tag{B.14}$$

Since we know have the Jacobian and error for a single point correspondence, we can now, similarly to the point to point method, define a stacked Jacobian and error, and perform the minimization using equation B.12 and B.6 applied with the above matrices.

Using the reference normal Using the reference normal presents one main advantage *w.r.t.* using the current normal: accuracy. In many systems, such as SLAM, the reference pointcloud is iteratively refined as new keyframes are added, thus providing a better normal estimation. In that case, the error can be defined as

$$\begin{aligned} f: \quad (\mathbb{R}^4, \mathbb{R}^4, \mathbb{R}^4, \mathfrak{se}(3)) &\to \mathbb{R} \\ (\mathbf{n_i}, \mathbf{P_i}, \mathbf{P_i}^*, \boldsymbol{\xi}) &\mapsto e_i = \mathbf{R}(\xi)\hat{\mathbf{R}}\mathbf{n}_\mathbf{i}^{*T} \Pi_0(\mathbf{P}_{\phi(\mathbf{i})} - \mathbf{T}(\xi)\hat{\mathbf{T}}\mathbf{P}_\mathbf{i}^*) \end{aligned} \tag{B.15}$$

Appendix C

List of software

In order to increase the reach and usefulness of this work, efforts were made to provide ready-to-use C++ libraries and applications for the main subjects studied in this thesis. Unfortunately not all of the work presented here could be made publicly available as of yet.

RobCalib

Shorthand for *Robot Calibraion*, this library implements both online Hand-Eye and Robot-Eye calibration with dense visual SLAM. It was used extensively in this thesis, in particular to provide Hand-Eye calibration for HRP4 RGB-D sensor, MOCAP markers frame on the robot, and on a dataset recorded with a hand-held RGB-D sensor. It can be used as a library to provide online calibration, or through sample ROS-based applications are provided to provide ready-to-use calibration tools. For implementation details, please refer to Section 2.6.1.
https://github.com/arntanguy/robcalib

RBVis - Robot Visualization

This library allows to render a model of the robot from any desired perspective. This was developed and used to generate the self-occlusion masks presented in Section3.2.2, but is suitable for any robot supported by RBDyn.
https://gite.lirmm.fr/atanguy/rbvis

Iterative Closest Point Library

The ICP method described in Section **??** and the robustness improvements of Section 3.2.3 were implemented an published as an open-source C++ library. This library has been extensively used in this thesis for all ICP-registration applications, such as the results depicted by Figure 3.6, 3.8, 3.5. It was also integrated into the *choreonoid* application and used during the *DARPA Robotics Challenge*.
https://github.com/arntanguy/icp

Uniform Mesh Sampling

The uniform mesh sampling method described by Section 3.1.1 was implemented as an open-source C++ library. This library was heavily inspired by the great python library *pyntcloud* David de la Iglesia [2017]. The main advantage of this sampling method is that it guarantees a uniform distribution of points regardless of the arrangement of triangles in the mesh.
https://github.com/arntanguy/mesh_sampling

Savitzky-Golay Filtering based on Gram Polynomials

Savitzky-Golay is a widely described, and very popular method for filtering signals. Yet, to our knowledge, no good open-source C++ implementation was available. We provide a generic and efficient implementation based on Gram polynomials. This library has been extensively used in this thesis to filter noisy 6D-pose and velocities obtained from various odometry methods, and by my colleagues at *LIRMM* in their own projects.
https://github.com/arntanguy/gram_savitzky_golay

Multi-Contact QP Control Framework (mc_rtc)

mc_rtc is a framework for QP-based multi-contact control of humanoid robots. This is the basis upon which all control method were built and implemented. Substantial improvements were made during this thesis, such as

Closed-Loop Implementation The method described in Chapter 3 was integrated within the framework and can be used with any supported robot.

State Estimation Various state-estimation method were implemented to estimate the attitude of the floating base from SLAM or VICON trackers and to estimate CoP and ZMP from force sensors measurements.

Trajectory Task defines end-effector trajectories that follow bezier curves passing through user specified waypoints. This was used to provide smooth end-effector motions for entering the A400M Aircraft.

LookAt Task to look at and track surface in the environment with the vision sensor.

Various Contributions

Besides the aforementioned list of software, during the course of this thesis, I was brought to interact and make some minor contributions to many libraries and applications, such as *PX2M*, *Roboptim*, *Sophus*, *ElasticFusion*, *ORB2-SLAM*, *Rviz*, and more.

References

[Abe et al. 2007] ABE, Yeuhi ; DA SILVA, Marco ; POPOVIĆ, Jovan: Multiobjective control with frictional contacts. In: *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* Eurographics Association (Veranst.), 2007, S. 249–258

[Aloimonos et al. 1988] ALOIMONOS, John ; WEISS, Isaac ; BANDYOPADHYAY, Amit: Active vision. In: *International journal of computer vision* 1 (1988), Nr. 4, S. 333–356

[Atkeson 2015] ATKESON, Christopher G.: *What Happened at the DARPA Robotics Challenge?* https://www.cs.cmu.edu/~cga/drc/events. 2015

[Audras et al. 2011] AUDRAS, Cedric ; COMPORT, A ; MEILLAND, Maxime ; RIVES, Patrick: Real-time dense appearance-based SLAM for RGB-D sensors. In: *Australasian Conf. on Robotics and Automation* Bd. 2, 2011, S. 2

[Bailey et Durrant-Whyte 2006] BAILEY, Tim ; DURRANT-WHYTE, Hugh: Simultaneous localization and mapping (SLAM): Part II. In: *IEEE Robotics and Automation Magazine* 13 (2006), Nr. 3, S. 108–117

[Baudouin et al. 2011] BAUDOUIN, Léo ; PERRIN, Nicolas ; MOULARD, Thomas ; LAMIRAUX, Florent ; STASSE, Olivier ; YOSHIDA, Eiichi: Real-time replanning using 3d environment for humanoid robot. In: *11th IEEE-RAS International Conference on Humanoid Robots (Humanoids)* IEEE (Veranst.), 2011, S. 584–589

[Benallegue et Lamiraux 2015] BENALLEGUE, Mehdi ; LAMIRAUX, Florent: Estimation and stabilization of humanoid flexibility deformation using only inertial measurement units and contact information. In: *International Journal of Humanoid Robotics* 12 (2015), Nr. 03, S. 1550025

[Besl et McKay 1992] BESL, P. ; MCKAY, N.: A Method for Registration of 3-D Shapes. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* Bd. 14, 1992, S. 239–256

[Birbach et al. 2012] BIRBACH, Oliver ; BÄUML, Berthold ; FRESE, Udo: Automatic and self-contained calibration of a multi-sensorial humanoid's upper body. In: *Proceedings - IEEE International Conference on Robotics and Automation* (2012), S. 3103–3108. – ISBN 9781467314039

[Blanco 2010] BLANCO, Jl: A tutorial on se (3) transformation parameterizations and on-manifold optimization. In: *University of Malaga, Tech. Rep* (2010)

[Bouyarmane et Kheddar 2011a] BOUYARMANE, Karim ; KHEDDAR, Abderrahmane: Multi-contact stances planning for multiple agents. In: *IEEE International Conference on Robotics and Automation*, 2011, S. 5246–5253

[Bouyarmane et Kheddar 2011b] BOUYARMANE, Karim ; KHEDDAR, Abderrahmane: Using a multi-objective controller to synthesize simulated humanoid robot motion with changing contact configurations. In: *IEEE International Conference on Intelligent Robots and Systems*, 2011, S. 4414–4419

[Bouyarmane et Kheddar 2012] BOUYARMANE, Karim ; KHEDDAR, Abderrahmane: Humanoid robot locomotion and manipulation step planning. In: *Advanced Robotics* (2012), July/September

[Bouyarmane et al. 2012] BOUYARMANE, Karim ; VAILLANT, Joris ; KEITH, François ; KHEDDAR, Abderrahmane: Exploring humanoid robots locomotion capabilities in virtual disaster response scenarios. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)* IEEE (Veranst.), 2012, S. 337–342

[Bretl 2006] BRETL, Timothy: Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem. In: *The International Journal of Robotics Research* 25 (2006), Nr. 4, S. 317–342

[Bridson 2007] BRIDSON, Robert: Fast Poisson disk sampling in arbitrary dimensions. In: *SIGGRAPH sketches*, 2007, S. 22

[Brossette et al. 2013] BROSSETTE, Stanislas ; VAILLANT, Joris ; KEITH, François ; ESCANDE, Adrien ; KHEDDAR, Abderrahmane: Point-Cloud Multi-Contact Planning for Humanoids: Preliminary Results. In: *IEEE Conference on Robotics Automation and Mechatronics*. Manila, Philippines, 12-15 November 2013, S. 19–24

[Cadena et al. 2016] CADENA, C. ; CARLONE, L. ; CARRILLO, H. ; LATIF, Y. ; SCARAMUZZA, D. ; NEIRA, J. ; REID, I. ; LEONARD, J.J.: Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age. In: *IEEE Transactions on Robotics* 32 (2016), Nr. 6, S. 1309–1332

[Caron et al. 2018a] CARON, Stéphane ; ESCANDE, Adrien ; LANARI, Leonardo ; MALLEIN, Bastien: Capturability-based Analysis, Optimization and Control of 3D Bipedal Walking. In: *arXiv preprint arXiv:1801.07022* (2018)

[Caron et Kheddar 2017] CARON, Stéphane ; KHEDDAR, Abderrahmane: Dynamic walking over rough terrains by nonlinear predictive control of the floating-base inverted pendulum. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* IEEE (Veranst.), 2017, S. 5017–5024

[Caron et al. 2018b] CARON, Stéphane ; KHEDDAR, Abderrahmane ; TEMPIER, Olivier: *Stair Climbing Stabilization of the HRP-4 Humanoid Robot using Whole-body Admittance Control*. September 2018. – URL https://hal.archives-ouvertes.fr/hal-01875387. – working paper or preprint

[Caron et al. 2015] CARON, Stéphane ; PHAM, Quang-Cuong ; NAKAMURA, Yoshihiko: Stability of surface contacts for humanoid robots: Closed-form formulae of the contact wrench cone for rectangular support areas. In: *IEEE International Conference on Robotics and Automation (ICRA)* IEEE (Veranst.), 2015, S. 5107–5112

[Caron et al. 2017] CARON, Stéphane ; PHAM, Quang-Cuong ; NAKAMURA, Yoshihiko: Zmp support areas for multicontact mobility under frictional constraints. In: *IEEE Transactions on Robotics* 33 (2017), Nr. 1, S. 67–80

[Caron 2018] CARON, Stéphane: *Equations of motion*. https://scaron.info/teaching/equations-of-motion.html. 2018

[Carpentier et al. 2016] CARPENTIER, Justin ; TONNEAU, Steve ; NAVEAU, Maximilien ; STASSE, Olivier ; MANSARD, Nicolas: A versatile and efficient pattern generator for generalized legged locomotion. In: *IEEE International Conference on Robotics and Automation (ICRA)* IEEE (Veranst.), 2016, S. 3555–3561

[Chasles 1831] CHASLES, M: Note sur les propriétés générales du système de deux corps semblables entre eux, placés d'une manière quelconque dans l'espace; et sur le déplacement fini, ou infiniment petit d'un corps solide libre. In: *Bulletin des Sciences Mathématiques de Firussac* XJV (1831), S. 321–326

[Chen 1991] CHEN, H.H.: A screw motion approach to uniqueness analysis of head-eye geometry. In: *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (1991), S. 145–151

[Chretien et al. 2016] CHRETIEN, Benjamin ; ESCANDE, Adrien ; KHEDDAR, Abderrahmane: GPU robot motion planning using semi-infinite nonlinear programming. In: *IEEE Transactions on Parallel and Distributed Systems* (2016)

[Cisneros et al. 2015] CISNEROS, Rafael ; KAJITA, Shuuji ; SAKAGUCHI, Takeshi ; NAKAOKA, Shin'ichiro ; MORISAWA, Mitsuharu ; KANEKO, Kenji ; KANEHIRO, Fumio: Task-level teleoperated manipulation for the HRP-2Kai humanoid robot. In: *IEEE-RAS 15th International Conference on Humanoid Robots*, 2015, S. 1102–1108

[Comport 2005] COMPORT, Andrew I.: *Towards a computer imagination: Robust real-time 3D tracking of rigid and articulated objects for augmented reality and robotics*, Université de Rennes I, Dissertation, 2005. – 291 S. – URL http://www.i3s.unice.fr/{~}comport/publications/2005{_}these{_}comport.pdf

[Comport et al. 2007] COMPORT, Andrew I. ; MALIS, Ezio ; RIVES, Patrick: Accurate Quadrifocal Tracking for Robust 3D Visual Odometry. In: *IEEE International Conference on Robotics and Automation (ICRA)* Citeseer (Veranst.), 2007, S. 40–45

[Curless et Levoy 1996] CURLESS, Brian ; LEVOY, Marc: A volumetric method for building complex models from range images. In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* ACM (Veranst.), 1996, S. 303–312

[Dame et Marchand 2010] DAME, Amaury ; MARCHAND, Eric: Accurate Real-time Tracking Using Mutual Information. In: *IEEE International Symposium on Mixed and Augmented Reality* (2010), S. 47–56. ISBN 978-1-4244-9343-2

[David de la Iglesia 2017] DAVID DE LA IGLESIA: *3D point cloud generation from 3D triangular mesh*. https://medium.com/@daviddelaiglesiacastro/3d-point-cloud-generation-from-3d-triangular-mesh-bbb602ecf238. 2017

[De Simone et al. 2017] DE SIMONE, Daniele ; SCIANCA, Nicola ; FERRARI, Paolo ; LANARI, Leonardo ; ORIOLO, Giuseppe: MPC-based humanoid pursuit-evasion in the presence of obstacles. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, S. 5245–5250

[Durrant-Whyte et Bailey 2006] DURRANT-WHYTE, Hugh ; BAILEY, Tim: Simultaneous localization and mapping: part I. In: *IEEE robotics and automation magazine* 13 (2006), Nr. 2, S. 99–110

[Engel et al. 2015] ENGEL, J. ; STUECKLER, J. ; CREMERS, D.: Large-Scale Direct SLAM with Stereo Cameras. In: *International Conference on Intelligent Robots and Systems*, September 2015

[Engel et al. 2014] ENGEL, Jakob ; SCHÖPS, Thomas ; CREMERS, Daniel: LSD-SLAM: Large-scale direct monocular SLAM. In: *European Conference on Computer Vision* Springer (Veranst.), 2014, S. 834–849

[Escande et Kheddar 2009] ESCANDE, Adrien ; KHEDDAR, Abderrahmane: Contact planning for acyclic motion with tasks constraints. In: *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on* IEEE (Veranst.), 2009, S. 435–440

[Escande et al. 2013] ESCANDE, Adrien ; KHEDDAR, Abderrahmane ; MIOSSEC, Sylvain: Planning contact points for humanoid robots. In: *Robotics and Autonomous Systems* 61 (2013), Nr. 5, S. 428–442

[Escande et al. 2014] ESCANDE, Adrien ; MANSARD, Nicolas ; WIEBER, Pierre-Brice: Hierarchical quadratic programming: Fast online humanoid-robot motion generation. In: *The International Journal of Robotics Research* 33 (2014), Nr. 7, S. 1006–1028

[Fallon et al. 2014] FALLON, Maurice F. ; ANTONE, Matthew ; ROY, Nicholas ; TELLER, Seth: Drift-free humanoid state estimation fusing kinematic, inertial and lidar sensing. In: *14th IEEE-RAS International Conference on Humanoid Robots (Humanoids)* IEEE (Veranst.), 2014, S. 112–119

[Fallon et al. 2015] FALLON, Maurice F. ; MARION, Pat ; DEITS, Robin ; WHELAN, Thomas ; ANTONE, Matthew ; MCDONALD, John ; TEDRAKE, Russ: Continuous humanoid locomotion over uneven terrain using stereo fusion. In: *IEEE-RAS International Conference on Humanoid Robots*. Seoul, Korea, 3-5 November 2015, S. 881–888

[Featherstone 2014] FEATHERSTONE, Roy: *Rigid body dynamics algorithms*. Springer, 2014

[Feng et al. 2016] FENG, Siyuan ; XINJILEFU, X ; ATKESON, Christopher G. ; KIM, Joohyung: Robust dynamic walking using online foot step optimization. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* IEEE (Veranst.), 2016, S. 5373–5378

[Gorry 1990] GORRY, Peter A.: General Least-Squares Smoothing and Differentiation by the Convolution (Savitzky-Golay) Method. In: *Analytical Chemistry* 62 (1990), Nr. 6, S. 570–573

[Gramkow 2001] GRAMKOW, Claus: On averaging rotations. In: *Journal of Mathematical Imaging and Vision* 15 (2001), Nr. 1-2, S. 7–16

[Heller et al. 2014] HELLER, Jan ; HENRION, Didier ; PAJDLA, Tomas: Hand-eye and robot-world calibration by global polynomial optimization. In: *Proceedings - IEEE International Conference on Robotics and Automation* (2014), S. 3157–3164

[Herdt et al. 2010] HERDT, A. ; PERRIN, N. ; WIEBER, P. B.: Walking without thinking about it. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, S. 190–195. – ISSN 2153-0858

[Hersch et al. 2008] HERSCH, Micha ; SAUSER, Eric ; BILLARD, Aude: Online Learning of the Body Schema. In: *International Journal of Humanoid Robotics* 05 (2008), Nr. 02, S. 161–181. – ISSN 0219-8436

[Huber 2011] HUBER, Peter J.: Robust statistics. In: *International Encyclopedia of Statistical Science*. Springer, 2011, S. 1248–1251

[Huber et al. 1964] HUBER, Peter J. et al.: Robust estimation of a location parameter. In: *The annals of mathematical statistics* 35 (1964), Nr. 1, S. 73–101

[Innmann et al. 2016] INNMANN, Matthias ; ZOLLHÖFER, Michael ; NIESSNER, Matthias ; THEOBALT, Christian ; STAMMINGER, Marc: VolumeDeform: Real-time volumetric non-rigid reconstruction. In: *European Conference on Computer Vision* Springer (Veranst.), 2016, S. 362–379

[Ireta 2018] IRETA, Fernando: *Global Pose Estimation and Tracking for RGB-D Localization and 3D Mapping*, UNIVERSITÉ DE NICE SOPHIA ANTIPOLIS, Dissertation, 2018

[Ireta Munoz et Comport 2017] IRETA MUNOZ, Fernando I. ; COMPORT, Andrew I.: Global Point-to-hyperplane ICP: Local and Global Pose Estimation by Fusing Color and Depth. In: *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*. Daegu, South Korea, 2017

[Johnson et al. 2015] JOHNSON, Matthew ; SHREWSBURY, Brandon ; BERTRAND, Sylvain ; WU, Tingfan ; DURAN, Daniel ; FLOYD, Marshall ; ABELES, Peter ; STEPHEN, Douglas ; MERTINS, Nathan ; LESMAN, Alex ; CARFF, John ; RIFENBURGH, William ; KAVETI, Pushyami ; STRAATMAN, Wessel ; SMITH, Jesper ; GRIFFIOEN, Maarten ; LAYTON, Brooke ; DE BOER, Tomas ; KOOLEN, Twan ; NEUHAUS, Peter ; PRATT, Jerry: Team IHMC's lessons learned from the DARPA robotics challenge trials. In: *Journal of Field Robotics* 32 (2015), Nr. 2, S. 192–208

[Kajita et al. 2014] KAJITA, Shuuji ; HIRUKAWA, Hirohisa ; HARADA, Kensuke ; YOKOI, Kazuhito: *Introduction to Humanoid Robotics*. Bd. 101. 2014. – ISBN 364254536X

[Kajita et al. 2006] KAJITA, Shuuji ; MORISAWA, Mitsuharu ; HARADA, Kensuke ; KANEKO, Kenji ; KANEHIRO, Fumio ; FUJIWARA, Kiyoshi ; HIRUKAWA, Hirohisa: Biped walking pattern generator allowing auxiliary zmp control. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, S. 2993–2999

[Kanehira et al. 2002] KANEHIRA, Noriyuki ; KAWASAKI, TU ; OHTA, Shigehiko ; ISMUMI, T ; KAWADA, Tadahiro ; KANEHIRO, Fumio ; KAJITA, Shuuji ; KANEKO, Kenji: Design and experiments of advanced leg module (HRP-2L) for humanoid robot (HRP-2) development. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems* Bd. 3 IEEE (Veranst.), 2002, S. 2455–2460

[Kaneko et al. 2011] KANEKO, Kenji ; KANEHIRO, Fumio ; MORISAWA, Mitsuharu ; AKACHI, Kazuhiko ; MIYAMORI, Go ; HAYASHI, Atsushi ; KANEHIRA, Noriyuki: Humanoid robot hrp-4-humanoid robotics platform with lightweight and slim body. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* IEEE (Veranst.), 2011, S. 4400–4407

[Kaneko et al. 2015] KANEKO, Kenji ; MORISAWA, Mitsuharu ; KAJITA, Shuuji ; NAKAOKA, Shin'ichiro ; SAKAGUCHI, Takeshi ; CISNEROS, Rafael ; KANEHIRO, Fumio: Humanoid robot HRP-2Kai—Improvement of HRP-2 towards disaster response tasks. In: *IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)* IEEE (Veranst.), 2015, S. 132–139

[Kastner et al. 2015] KASTNER, Tobias ; ROFER, Thomas ; LAUE, Tim: Automatic robot calibration for the NAO. In: *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)* 8992 (2015), S. 233–244. – ISBN 9783319186146

[Kerl et al. 2013] KERL, Christian ; STURM, Jurgen ; CREMERS, Daniel: Dense visual SLAM for RGB-D cameras. In: *IEEE International Conference on Intelligent Robots and Systems*, 2013, S. 2100–2106

[Ma et al. 2003] MA, Yi ; SOATTO, Stefano ; KOSECKA, Jana ; SASTRY, S. S.: *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag, 2003. – ISBN 0387008934

[Mansard et al. 2009] MANSARD, Nicolas ; KHATIB, Oussama ; KHEDDAR, Abderrahmane: A unified approach to integrate unilateral constraints in the stack of tasks. In: *IEEE Transactions on Robotics* 25 (2009), Nr. 3, S. 670–685

[Martinez-Cantin et al. 2010] MARTINEZ-CANTIN, Ruben ; LOPES, Manuel ; MONTESANO, Luis: Body schema acquisition through active learning. In: *Proceedings - IEEE International Conference on Robotics and Automation* (2010), S. 1860–1866. – ISBN 9781424450381

[Maye et al. 2016] MAYE, Jérôme ; SOMMER, Hannes ; AGAMENNONI, Gabriel ; SIEGWART, Roland ; FURGALE, Paul: Online self-calibration for robotic systems. In: *The International Journal of Robotics Research* 35 (2016), Nr. 4, S. 357–380

[Meilland et al. 2013a] MEILLAND, Maxime ; BARAT, Christian ; COMPORT, Andrew: 3d high dynamic range dense visual slam and its application to real-time object re-lighting. In: *IEEE International Symposium on Mixed and Augmented Reality*, 2013, S. 143–152

[Meilland et Comport 2013a] MEILLAND, Maxime ; COMPORT, Andrew I.: On unifying key-frame and voxel-based dense visual SLAM at large scales. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Tokyo, Japan, 3-7 November 2013, S. 3677–3683

[Meilland et Comport 2013b] MEILLAND, Maxime ; COMPORT, Andrew I.: Super-resolution 3D tracking and mapping. In: *IEEE International Conference on Robotics and Automation*, 6-10 May 2013, S. 5717–5723

[Meilland et al. 2013b] MEILLAND, Maxime ; DRUMMOND, Tom ; COMPORT, Andrew I.: A unified rolling shutter and motion blur model for 3D visual registration. In: *IEEE International Conference on Computer Vision* (2013), S. 2016–2023

[Montemerlo et al. 2002] MONTEMERLO, Michael ; THRUN, Sebastian ; KOLLER, Daphne ; WEGBREIT, Ben et al.: FastSLAM: A factored solution to the simultaneous localization and mapping problem. In: *Aaai/iaai* 593598 (2002)

[Moulard et al. 2013] MOULARD, Thomas ; LAMIRAUX, Florent ; BOUYARMANE, Karim ; YOSHIDA, Eiichi: Roboptim: an optimization framework for robotics. In: *Robomec*, 2013

[Mur-Artal et al. 2015] MUR-ARTAL, Raul ; MONTIEL, Jose Maria M. ; TARDOS, Juan D.: ORB-SLAM: a versatile and accurate monocular SLAM system. In: *IEEE Transactions on Robotics* 31 (2015), Nr. 5, S. 1147–1163

[Murray et al. 1994] MURRAY, Richard M. ; SASTRY, S. S. ; ZEXIANG, Li: *A Mathematical Introduction to Robotic Manipulation*. 1st. Boca Raton, FL, USA : CRC Press, Inc., 1994

[Naveau et al. 2017] NAVEAU, Maximilien ; KUDRUSS, Manuel ; STASSE, Olivier ; KIRCHES, Christian ; MOMBAUR, Katja ; SOUÈRES, Philippe: A reactive walking pattern generator based on nonlinear model predictive control. In: *IEEE Robotics and Automation Letters* 2 (2017), Nr. 1, S. 10–17

[Newcombe et al. 2015] NEWCOMBE, Richard A. ; FOX, Dieter ; SEITZ, Steven M.: Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, S. 343–352

[Newcombe et al. 2011a] NEWCOMBE, Richard A. ; IZADI, Shahram ; HILLIGES, Otmar ; MOLYNEAUX, David ; KIM, David ; DAVISON, Andrew J. ; KOHI, Pushmeet ; SHOTTON, Jamie ; HODGES, Steve ; FITZGIBBON, Andrew: KinectFusion: Real-time dense surface mapping and tracking. In: *10th IEEE international symposium on Mixed and augmented reality (ISMAR)* IEEE (Veranst.), 2011, S. 127–136

[Newcombe et al. 2011b] NEWCOMBE, Richard A. ; LOVEGROVE, Steven J. ; DAVISON, Andrew J.: DTAM: Dense Tracking and Mapping in Real-time. In: *Proceedings of the 2011 International Conference on Computer Vision*. Washington, DC, USA : IEEE Computer Society, 2011, S. 2320–2327. – ISBN 978-1-4577-1101-5

[Park et Martin 1994] PARK, Frank C. ; MARTIN, Bryan J.: Robot Sensor Calibration: Solving AX = XB on the Euclidean Group. In: *IEEE Transactions on Robotics and Automation* 10 (1994), Nr. 5, S. 717–721

[Pradeep et al. 2014] PRADEEP, Vijay ; KONOLIGE, Kurt ; BERGER, Eric: Calibrating a multi-arm multi-sensor robot: A bundle adjustment approach. In: *Springer Tracts in Advanced Robotics* Bd. 79, 2014, S. 211–225. – ISBN 9783642285714

[Roncone et al. 2014] RONCONE, Alessandro ; HOFFMANN, Matej ; PATTACINI, Ugo ; METTA, Giorgio: Automatic kinematic chain calibration using artificial skin: Self-touch in the iCub humanoid robot. In: *Proceedings - IEEE International Conference on Robotics and Automation* (2014), S. 2305–2312. – ISBN 9781479936847

[Roth et Vona 2009] ROTH, Henry ; VONA, Marsette: Moving Volume KinectFusion. In: *IEEE-RAS International Conference on Humanoid Robots* Bd. 9, 2009, S. 62–67

[Rusu et al. 2009] RUSU, Radu B. ; BLODOW, Nico ; BEETZ, Michael: Fast point feature histograms (FPFH) for 3D registration. In: *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on* IEEE (Veranst.), 2009, S. 3212–3217

[Salas-Moreno et al. 2014] SALAS-MORENO, Renato F. ; GLOCKEN, Ben ; KELLY, Paul H. ; DAVISON, Andrew J.: Dense planar SLAM. In: *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)* IEEE (Veranst.), 2014, S. 157–164

[Samy et Kheddar 2015] SAMY, Vincent ; KHEDDAR, Abderrahmane: Falls control using posture reshaping and active compliance. In: *IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)* IEEE (Veranst.), 2015, S. 908–913

[Savitzky et Golay 1964] SAVITZKY, Abraham ; GOLAY, Marcel J.: Smoothing and differentiation of data by simplified least squares procedures. In: *Analytical chemistry* 36 (1964), Nr. 8, S. 1627–1639

[Scianca et al. 2016] SCIANCA, Nicola ; COGNETTI, Marco ; DE SIMONE, Daniele ; LANARI, Leonardo ; ORIOLO, Giuseppe: Intrinsically stable MPC for humanoid gait generation. In: *16th IEEE-RAS Int. Conf. on Humanoid Robots*, 2016, S. 101–108

[Scona et al. 2017] SCONA, Raluca ; NOBILI, Simona ; PETILLOT, Yvan R. ; FALLON, Maurice: Direct visual SLAM fusing proprioception for a humanoid robot. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* IEEE (Veranst.), 2017, S. 1419–1426

[Segal et al. 2009] SEGAL, Aleksandr ; HAEHNEL, Dirk ; THRUN, Sebastian: Generalized-icp. In: *Robotics: science and systems* Bd. 2, 2009, S. 435

[Spong et Vidyasagar 2004] SPONG, Mark W. ; VIDYASAGAR, Mathukumalli: *Robot dynamics and control*. John Wiley & Sons, 2004

[Stasse et al. 2006] STASSE, Olivier ; DAVISON, Andrew J. ; SELLAOUTI, Ramzi ; YOKOI, Kazuhito: Real-time 3d slam for humanoid robot considering pattern generator information. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, S. 348–355

[Stasse et al. 2009] STASSE, Olivier ; VERRELST, Bjorn ; VANDERBORGHT, Bram ; YOKOI, Kazuhito: Strategies for humanoid robots to dynamically walk over large obstacles. In: *IEEE Transactions on Robotics* 25 (2009), Nr. 4, S. 960–967

[Straub et al. 2017] STRAUB, Julian ; CAMPBELL, Trevor ; HOW, Jonathan P. ; FISHER III, John W.: Efficient Global Point Cloud Alignment using Bayesian Nonparametric Mixtures. In: *CVPR*, URL https://arxiv.org/abs/1603.04868, 2017

[Taketomi et al. 2017] TAKETOMI, Takafumi ; UCHIYAMA, Hideaki ; IKEDA, Sei: Visual SLAM algorithms: a survey from 2010 to 2016. In: *IPSJ Transactions on Computer Vision and Applications* 9 (2017), Nr. 1, S. 16

[Tanguy et al. 2018a] TANGUY, Arnaud ; DE SIMONE, Daniele ; COMPORT, Andrew I. ; ORIOLO, Giuseppe ; KHEDDAR, Abderrahmane: *Closed-loop MPC with Dense Visual SLAM-Stability through Reactive Stepping*. September 2018. – URL https://hal.archives-ouvertes.fr/hal-01883725. – Preprint - Submitted to the 2019 *IEEE International Conference on Robotics and Autonmation (ICRA)*

[Tanguy et al. 2016] TANGUY, Arnaud ; GERGONDET, Pierre ; COMPORT, Andrew I. ; KHEDDAR, Abderrahmane: Closed-loop RGB-D SLAM Multi-Contact Control for humanoid robots. In: *IEEE/SICE International Symposium on System Integration (SII)*. Sapporo, Japan, December 2016, S. 51–57. – URL https://hal.archives-ouvertes.fr/hal-01568048

[Tanguy et al. 2018b] TANGUY, Arnaud ; KHEDDAR, Abderrahmane ; COMPORT, Andrew I.: Online eye-robot self-calibration. In: *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*. Brisbane, France : IEEE, May 2018. – URL https://hal.archives-ouvertes.fr/hal-01883715

[Tonneau et al. 2018] TONNEAU, Steve ; DEL PRETE, Andrea ; PETTRÉ, Julien ; PARK, Chonhyon ; MANOCHA, Dinesh ; MANSARD, Nicolas: An efficient acyclic contact planner for multiped robots. In: *IEEE Transactions on Robotics* (2018)

[Tsai et Lenz 1989] TSAI, Roger Y. ; LENZ, Reimar K.: A New Technique for Fully Autonomous and Efficient 3D Robotics Hand/Eye Calibration. In: *IEEE Transactions on Robotics and Automation* 5 (1989), Nr. 3, S. 345–358

[Tykkälä et al. 2011] TYKKÄLÄ, Tommi ; AUDRAS, Cédric ; COMPORT, Andrew I.: Direct iterative closest point for real-time visual odometry. In: *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)* IEEE (Veranst.), 2011, S. 2050–2056

[Vaillant et al. 2016] VAILLANT, Joris ; KHEDDAR, Abderrahmane ; AUDREN, Hervé ; KEITH, François ; BROSSETTE, Stanislas ; ESCANDE, Adrien ; BOUYARMANE, Karim ; KANEKO, Kenji ; MORISAWA, Mitsuharu ; GERGONDET, Pierre ; YOSHIDA, Eiichi ; KAJITA, Suuji ; KANEHIRO, Fumio: Multi-contact vertical ladder climbing with an HRP-2 humanoid. In: *Autonomous Robots* 40 (2016), Nr. 3, S. 561–580

[Villa et Wieber 2017] VILLA, N. A. ; WIEBER, P.: Model predictive control of biped walking with bounded uncertainties. In: *IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, Nov 2017, S. 836–841. – ISSN 2164-0580

[Viola et Wells III 1997] VIOLA, Paul ; WELLS III, William M.: Alignment by maximization of mutual information. In: *International journal of computer vision* 24 (1997), Nr. 2, S. 137–154

[Whelan et al. 2012] WHELAN, T. ; KAESS, M. ; FALLON, M.F. ; JOHANNSSON, H. ; LEONARD, J.J. ; MCDONALD, J.B.: Kintinuous: Spatially Extended KinectFusion. In: *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*. Sydney, Australia, Jul 2012

[Whelan et al. 2015a] WHELAN, Thomas ; KAESS, Michael ; JOHANNSSON, Hordur ; FALLON, Maurice ; LEONARD, John J. ; MCDONALD, John: Real-time large-scale dense RGB-D SLAM with volumetric fusion. In: *The International Journal of Robotics Research* 34 (2015), Nr. 4-5, S. 598–626

[Whelan et al. 2013] WHELAN, Thomas ; KAESS, Michael ; LEONARD, John J. ; MCDONALD, John: Deformation-based loop closure for large scale dense RGB-D SLAM. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* IEEE (Veranst.), 2013, S. 548–555

[Whelan et al. 2015b] WHELAN, Thomas ; LEUTENEGGER, Stefan ; SALAS-MORENO, R ; GLOCKER, Ben ; DAVISON, Andrew: ElasticFusion: Dense SLAM without a pose graph. In: *International Journal of Robotics Research (IJRR)*, 2015

[Wieber 2005] WIEBER, Pierre-Brice: Some comments on the structure of the dynamics of articulated motion. In: *Fast Motions in Biomechanics and Robotics*, 2005

[Xinjilefu et al. 2015] XINJILEFU, X ; FENG, Siyuan ; ATKESON, Christopher G.: Center of mass estimator for humanoids and its application in modelling error compensation, fall detection and prevention. In: *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on* IEEE (Veranst.), 2015, S. 67–73

[Yang et al. 2013] YANG, Jiaolong ; LI, Hongdong ; JIA, Yunde: Go-icp: Solving 3d registration efficiently and globally optimally. In: *IEEE International Conference on Computer Vision*, 2013, S. 1457–1464

[Yokoi et al. 2004] YOKOI, Kazuhito ; KANEHIRO, Fumio ; KANEKO, Kenji ; KAJITA, Shuuji ; FUJIWARA, Kiyoshi ; HIRUKAWA, Hirohisa: Experimental study of humanoid robot HRP-1S. In: *The International Journal of Robotics Research* 23 (2004), Nr. 4-5, S. 351–362

[Zhou et al. 2016] ZHOU, Haoyin ; NI, Kai ; ZHOU, Qian ; ZHANG, Tao: An SfM Algorithm With Good Convergence That Addresses Outliers for Realizing Mono-SLAM. In: *IEEE Transactions on Industrial Informatics* 12 (2016), Nr. 2, S. 515–523

Personal papers

Tanguy et al. 2016 TANGUY, Arnaud ; GERGONDET, Pierre ; COMPORT, Andrew I. ; KHEDDAR, Abderrahmane: Closed-loop RGB-D SLAM Multi-Contact Control for humanoid robots. In: *IEEE/SICE International Symposium on System Integration (SII)*. Sapporo, Japan, December 2016, S. 51–57. – URL https://hal.archives-ouvertes.fr/hal-01568048

Tanguy et al. 2018b TANGUY, Arnaud ; KHEDDAR, Abderrahmane ; COMPORT, Andrew I.: Online eye-robot self-calibration. In: *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*. Brisbane, France : IEEE, May 2018. – URL https://hal.archives-ouvertes.fr/hal-01883715

Tanguy et al. 2018a TANGUY, Arnaud ; DE SIMONE, Daniele ; COMPORT, Andrew I. ; ORIOLO, Giuseppe ; KHEDDAR, Abderrahmane: *Closed-loop MPC with Dense Visual SLAM-Stability through Reactive Stepping*. September 2018. – URL https://hal.archives-ouvertes.fr/hal-01883725. – Preprint - Submitted to the 2019 *IEEE International Conference on Robotics and Autonmation (ICRA)*

List of figures

List of tables

Liste des symboles

$. \times .$	Cartesian product
$. \wedge .$	Cross product
$.^{+}$	Upper Bound of variable .
$.^{-}$	Lower Bound of variable .
$.^{-1}$	Inverse of .
$.^{T}$	Transpose of .
$\frac{\partial a}{\partial b}$	Derivative of a with respect to b
$\hat{v}$	3D skew-symmetric matrix such that $\hat{v}u = v \wedge u$
$\mathbf{a} \cdot \mathbf{b}$	Dot product between a and b
$\nabla .$	Gradient operator
$\vec{v}, \mathbf{v}$	Denotes a vector
$\mathbf{x}$	Optimization variable (usually in $\mathfrak{se}(3)$)

Superscripts

f	Variable describing the external forces
q	Variable describing the robot's joint configuration
R	3D Rotation
t	3D translation vector
Jac_i	Jacobian of Body i

${}^{B}X_{A}$ 3D transformation from frame A to frame B

B_i Body i

F_i Frame i

J_i Joint i

n_B Number of bodies

n_J Number of Joints

T_i Task i

X_i^J Transformation from the reference frame of joint i to the reference frame of its successor body

X_i^x Static transformation of joint i

Subscripts

$\mathbb{R}$ The Real space

$\mathbb{R}^+$ The set of positive Reals

$\mathbb{SE}(3)$ Special Euclidean Group (Rotations and Translations)

$\mathfrak{se}(3)$ The Lie algebra manifold associated with $\mathbb{SE}(3)$

$\mathbb{SO}(3)$ Special Orthogonal Group (Rotations)

$\mathfrak{so}(3)$ The Lie algebra manifold associated with $\mathbb{SO}(3)$

Other Symbols

$\mathbf{0}_n$ Matrix zero of dimension n

$\mathbf{1}_n$ Matrix identity of dimension n

Acronyms / Abbreviations

D6DSLAM SLAM software of Meilland et Comport [2013a]

SLAM Simultaneous Localization and Mapping

s.t. subject to

w.r.t. with respect to

DoF Degrees of freedom

FK Forward Kinematics

ICP Iterative Closest Point

IK Inverse Kinematics

PG Posture Generation

QP Quadratic Problem

RX, RY, RZ Rotations around $\vec{x}$, $\vec{y}$, and $\vec{z}$

TX, TY, TZ Translations along $\vec{x}$, $\vec{y}$, and $\vec{z}$