N° d'ordre: D.U: 2260
EDSPIC: 568

THÈSE

présentée devant

L'UNIVERSITÉ BLAISE PASCAL - CLERMONT II

pour obtenir le grade de

DOCTEUR D'UNIVERSITÉ
Spécialité: Vision et Robotique

par

Erol ÖZGÜR

Équipe d'accueil : ISPR de l'Institut Pascal
École Doctorale : Sciences Pour l'Ingénieur

Titre de la thèse :

From Lines To Dynamics of Parallel Robots

13 Juillet 2012

COMPOSITION DU JURY

M.	Grigore	GOGU	Président
M.	Jean-Pierre	MERLET	Rapporteur
M.	Jacques	GANGLOFF	Rapporteur
M.	Sébastien	BRIOT	Examinateur
M.	Philippe	MARTINET	Co-encadrant
M.	Nicolas	ANDREFF	Directeur de thèse

à Bérengère

Remerciements

Nicolas ANDREFF, sous votre direction, je suis devenu plus compétent. Vous êtes un très bon directeur de thèse. J’ai eu beaucoup de plaisir à travailler avec vous. Merci.

Philippe MARTINET, vos jugements et votre soutien m’ont porté jusqu’au bout de cette thèse. Merci.

Je remercie le jury : Grigore GOGU pour avoir présidé le jury et évalué le manuscrit ; Jean-Pierre MERLET et Jacques GANGLOFF pour avoir rapporté cette thèse et Sébastien BRIOT pour l’avoir examinée.

Redwan DAHMOUCHE, Nicolas BOUTON, Laurent MALATERRE, Tej DALLEJ, Datta RAMADASAN et Pierre LEBRALY, je vous remercie pour l’aide que vous m’avez apportée.

Mustafa ÜNEL, hocam, bana öğretiğiniz herşey bu tezin gerçekleşmesinde faydalı oldu. İyi bir akademisyen olabilmem için verdiğiniz tüm emeklere teşekkür ederim.

Enfin, je remercie le projet ANR VIRAGO qui a financé cette thèse.

Contents

Introduction **1**

1 State of The Art **5**
1.1 Parallel Robots . . . 5
1.1.1 Definition . . . 5
1.1.2 Joint Types and Graphical Layout . . . 6
1.1.3 Classification . . . 7
1.1.3.1 Kinematic Classification . . . 7
1.1.3.2 Architectural Classification . . . 8
1.1.4 Compared with Serial Robots . . . 8
1.1.4.1 Advantages . . . 8
1.1.4.2 Disadvantages . . . 9
1.1.5 The Duality of Parallel and Serial Robots . . . 9
1.1.6 Metrological Redundancy . . . 10
1.2 Modeling . . . 11
1.2.1 Kinematic Modeling . . . 11
1.2.1.1 The Zeroth-Order Kinematic Models . . . 11
1.2.1.2 The First-Order Kinematic Models . . . 13
1.2.1.3 Singularities . . . 15
1.2.2 Dynamic Modeling . . . 16
1.2.3 Dynamic Modeling Methods of Serial Robots . . . 17
1.2.3.1 Euler-Lagrange Method . . . 17
1.2.3.2 Newton-Euler Method / Luh-Walker-Paul's Algorithm . . . 18
1.2.3.3 d'Alembert Method / Principle of Virtual Work . . . 18
1.2.3.4 Kane's Method . . . 18
1.2.4 Dynamic Modeling Methods of Parallel Robots . . . 19
1.2.4.1 Khalil's Method . . . 19
1.3 Control . . . 19
1.3.1 Kinematic Control . . . 20
1.3.2 Dynamic Control . . . 23
1.4 Identification . . . 25
1.4.1 Geometric Identification . . . 25
1.4.2 Dynamic Identification . . . 27

1.5 MICMAC . . . 28
1.5.1 Introduction . . . 28
1.5.2 Modular MICMAC . . . 29
1.5.3 Integrated MICMAC . . . 29
1.5.4 Complementary Background . . . 30
1.5.4.1 Vision . . . 31
1.5.4.2 Lines and Robotic Legs . . . 31
1.6 Thesis Objective . . . 36

2 Modeling **37**
2.1 Introduction . . . 37
2.2 Motivation and Objective . . . 37
2.3 Discussions on the Inspiring Works . . . 38
2.3.1 Kane's Method . . . 38
2.3.2 Khalil's Method . . . 39
2.3.3 Tsai's Method . . . 39
2.4 Methodology . . . 40
2.4.1 The Descriptive Language of a Robot . . . 40
2.4.2 A Mathematical Language for a Robot . . . 40
2.4.2.1 Variables . . . 40
2.4.2.2 Constant Parameters . . . 41
2.4.2.3 Motion Space . . . 42
2.4.3 Proposed Modeling Formulation . . . 42
2.5 New Construction « Kinematic Element » Definition . . . 44
2.5.1 Construction Primitives . . . 45
2.5.1.1 Joints . . . 45
2.5.1.2 Links . . . 45
2.5.2 Geometric Representation of a « Kinematic Element » . . . 46
2.5.3 Dynamic Representation of a « Kinematic Element » . . . 49
2.6 Kinematics of a « Kinematic Element » . . . 49
2.6.1 Positions . . . 49
2.6.2 Rotational Velocity and Acceleration . . . 50
2.6.3 Translational Velocity and Acceleration . . . 51
2.7 Dynamics of a « Kinematic Element » . . . 52
2.7.1 Active Forces and Torques . . . 52
2.7.1.1 Forces and Torques of Actuators . . . 52
2.7.1.2 Force of Gravity . . . 52
2.7.2 Reactive Forces and Torques . . . 52
2.7.2.1 Inertial Forces and Torques . . . 52
2.7.2.2 Frictional Forces and Torques . . . 53
2.8 Physical Formation of a Parallel Robot . . . 54
2.8.1 A Base Platform . . . 54
2.8.2 A Kinematic Leg . . . 54
2.8.3 Nacelle . . . 54

2.8.4 A Parallel Robot . . . 55
2.9 Distribution of Nacelle Dynamics . . . 55
2.9.1 Moving Platform Distribution . . . 55
2.9.2 Nacelle Distribution . . . 57
2.10 Sate Variables and Motion Basis of a Parallel Robot . . . 58
2.10.1 State Variables . . . 58
2.10.2 Motion Basis . . . 58
2.11 Kinematics of a Parallel Robot . . . 59
2.11.1 Mass Centers . . . 59
2.11.2 Velocities . . . 60
2.11.2.1 Translational Velocity . . . 60
2.11.2.2 Rotational Velocity . . . 60
2.11.3 Accelerations . . . 61
2.11.3.1 Translational Acceleration . . . 61
2.11.3.2 Rotational Acceleration . . . 61
2.12 Kinematic Constraints of a Parallel Robot . . . 61
2.12.1 Configuration Constraints . . . 61
2.12.2 Motion Constraints . . . 62
2.13 Kinematic Coordinates of a Parallel Robot . . . 63
2.14 Dynamic Coordinates of a Parallel Robot . . . 65
2.14.1 Listing the Active and Reactive Forces . . . 65
2.14.2 Computing Dynamic Coordinates . . . 66
2.15 Dynamic Constraints of a Parallel Robot . . . 67
2.16 Linear Solution for the Inverse Dynamics . . . 68
2.17 A Global View to the Proposed Methodology . . . 69
2.17.1 Compared to/with Khalil's, Kane's and Tsai's Methods . . . 70
2.17.2 Originality . . . 70
2.18 Applied to the Quattro Parallel Robot . . . 70
2.18.1 State Variables . . . 71
2.18.2 Kinematics . . . 73
2.18.3 Kinematic Constraints . . . 75
2.18.4 Kinematic Coordinates . . . 77
2.18.5 Dynamic Coordinates . . . 77
2.18.5.1 Listing Active and Reactive Forces . . . 77
2.18.6 Dynamic Constraints . . . 80
2.18.7 Inverse Dynamics . . . 80
2.19 Conclusions . . . 82

3 Control **83**
3.1 Introduction . . . 83
3.2 High-Speed Integrated Dynamic MICMAC Observer . . . 84
3.2.1 Differential Edge Kinematics of a Cylindrical Kinematic Element . . . 85
3.2.1.1 Notation . . . 85
3.2.1.2 Differential Edge Kinematics . . . 86

3.2.2 High-Speed Dynamic State Observer via Sequential Visual Sensing . . 87
3.2.2.1 Motivation for Sequential Visual Sensing 87
3.2.2.2 Algorithm: « Single-Iteration Virtual Visual Servoing » . . . 89
3.2.2.3 Notation 89
3.2.2.4 Spatiotemporal Reference Signal 90
3.2.2.5 Sequential Postures Error 90
3.2.2.6 Approximated Edge Evolution Model 92
3.2.2.7 Visual Servoing Control Law 94
3.2.2.8 Virtual Parallel Robot Dynamic State Update 95
3.2.2.9 Predicting Future Sub-Image Location 97
3.2.3 Applied to the Quattro Parallel Robot 98
3.2.3.1 On the Observability of the Legs 98
3.2.3.2 End-Effector Pose Representation 99
3.2.3.3 Validation By Simulations 99
3.2.3.4 Conclusions 104
3.3 Sensor-Based Computed-Torque Control based on Sequential Leg Observations 105
3.3.1 Control Variable 105
3.3.2 Versatile Control Law 106
3.3.3 Variations Upon the Control Space 108
3.3.4 Validation By Simulations 114
3.3.4.1 Feedback Sensing 114
3.3.4.2 Noises for Robustness Test 118
3.3.4.3 Performance Metric 118
3.3.4.4 End-Effector Computed-Torque Control (EE-CTC) 118
3.3.4.5 Results 119
3.3.4.6 Conclusions 123
3.4 Conclusions 124

4 Experiments **125**
4.1 High-Speed Integrated Dynamic MICMAC Observer 125
4.1.1 Test-Bed Setup 125
4.1.2 Experimental Scenario 126
4.1.3 Calibration 127
4.1.3.1 Camera-Quattro Parallel Robot Calibration from Leg Edges . 127
4.1.3.2 Coarse to Fine Calibration 129
4.1.4 Reference Trajectory 130
4.1.5 Data Collection 131
4.1.6 Off-Line Dynamic State Computation 132
4.1.7 Comparison 134
4.2 Inverse Dynamic Model 139
4.2.1 Validation of the Inverse Dynamic Model 139
4.2.2 Inverse Dynamic Model with High-Speed Dynamic State Observer . . 143
4.3 Conclusions 145

5 Conclusions and Perspectives **147**
5.1 Conclusions . . . 147
5.2 Perspectives . . . 148
5.2.1 Control-Oriented Linear Dynamic Modeling . . . 148
5.2.2 Vision-Based High-Speed Dynamic State Observer . . . 149
5.2.3 We do need VISION and DYNAMIC CONTROL . . . 150
5.2.4 From MICMAC to RODAP . . . 151

A Modeling: Applied to Parallel Robots **153**
A.1 The Gough-Stewart Robot . . . 153
A.1.1 Geometry and Notation . . . 153
A.1.2 State Variables . . . 154
A.1.3 Kinematics . . . 155
A.1.4 Kinematic Constraints . . . 156
A.1.5 Kinematic Coordinates . . . 157
A.1.6 Dynamic Coordinates . . . 157
A.1.7 Dynamic Constraints . . . 159
A.2 The Delta Robot . . . 160
A.2.1 Geometry and Notation . . . 160
A.2.2 State Variables . . . 161
A.2.3 Kinematics . . . 162
A.2.4 Kinematic Constraints . . . 163
A.2.5 Kinematic Coordinates . . . 164
A.2.6 Dynamic Coordinates . . . 164
A.2.7 Dynamic Constraints . . . 166
A.3 The 3RRR Robot . . . 167
A.3.1 Geometry and Notation . . . 167
A.3.2 State Variables . . . 168
A.3.3 Kinematics . . . 169
A.3.4 Kinematic Constraints . . . 169
A.3.5 Kinematic Coordinates . . . 171
A.3.6 Dynamic Coordinates . . . 171
A.3.7 Dynamic Constraints . . . 172
A.4 The Orthoglide Robot . . . 173
A.4.1 Geometry and Notation . . . 173
A.4.2 State Variables . . . 174
A.4.3 Kinematics . . . 175
A.4.4 Kinematic Constraints . . . 175
A.4.5 Kinematic Coordinates . . . 176
A.4.6 Dynamic Coordinates . . . 176
A.4.7 Dynamic Constraints . . . 178

References **180**

List of figures **191**

Abstract **198**

Introduction

Problem

In an industrial environment, mechanisms are assessed by two main criteria: quality and throughput. The ability of a mechanism to operate at a high speed certainly improves throughput, but what about quality? As a rule of thumb, the industry always demands better quality and more throughput, even though these two criteria conflict. Unfortunately with serial robots, there is a limit to the amount we can improve the quality and throughput.

In order to overcome this limit, parallel robots are designed to be faster and more accurate than serial robots. Today, they are being used more and more in industry. Although parallel robots are theoretically more skillful, they are structurally very complex to manipulate. Furthermore, parallel robots are mostly modeled and controlled with approaches adopted from serial robots, which is certainly not the best way. There are plenty of specific controls for mobile robots, for humanoids, etc., so why should parallel robots be controlled like serial robots?

Consequently, in order to push parallel robots to their limits, new methods are a must. So, we are faced with the problem of how to control parallel robots at high speed. We can divide this problem into 3 sub-sections:

1. How can parallel robots be simply and accurately modeled?
2. How can the dynamic state of a high speed parallel robot be measured?
3. What is the appropriate control space for better performance?

Objectives

Thus, the first objective of this thesis is to develop an appropriate control-oriented modeling approach for parallel robots. In modeling, we must always strive to obtain the simplest, most accurate and most applicable solution possible. The second objective of this thesis is to measure the posture and the velocity of a parallel robot at high speed with an off-the-shelf sensor(s). The third objective of this thesis is to explore the existing control spaces and to propose new ones in order to improve the control of parallel robots.

Contributions

In this thesis, the key contributions are built upon the orientations of the legs of a parallel robot. Figure 1 simply illustrates the global observation of the legs by a camera. The main

contributions are as follows:

- Initially, we pushed the modeling of parallel robots by means of leg orientations one step further. We moved it from kinematics to dynamics. This keeps the dynamic model of a parallel robot simple, clear, and linear. What is more this means that one can write the dynamic model of any complex parallel robot from beginning till end with just pen and paper.
- Secondly, we proposed a new approach for estimating the dynamic state of a parallel robot at high speed. We achieved this using only the partial visual contours of the legs. These visual contours were measured from the sequentially grabbed small sub-images of the legs during the motion of the parallel robot.

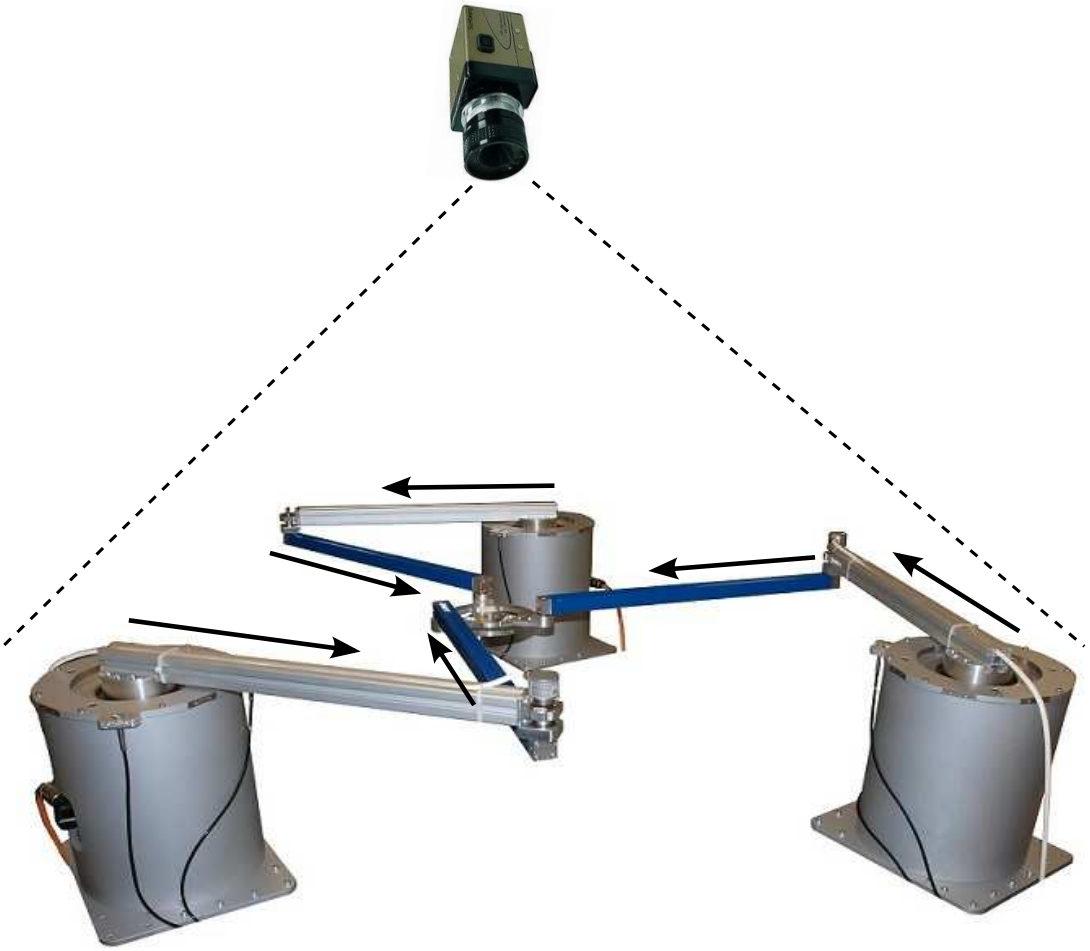


Figure 1 – A camera is observing the leg directions of a parallel robot. Leg orientation vectors unify modeling and control into a single linear control-oriented framework.

We published the following papers throughout this thesis:

1. “Linear Dynamic Modeling of Parallel Kinematic Manipulators from Observable Kinematic Elements", Int. Journal of Mechanism and Machine Theory, 2013;
2. “High Speed Parallel Kinematic Manipulator State Estimation from Legs Observation", IROS 2013;
3. “Dynamic Control of the Quattro Robot by the Leg Edges", ICRA 2011;
4. “Vector-Based Dynamic Modeling and Control of the Quattro Parallel Robot by means of Leg Orientations", ICRA 2010;
5. “On the Adequation of Dynamic Modeling and Control of Parallel Kinematic Manipulators", IMSD 2010.

Outline of the Thesis

The rest of this thesis proceeds as follows: Chapter 1 surveys the state-of-the-art works on parallel robots and discusses the MICMAC project; Chapter 2 outlines a linear framework for the kinematic and dynamic modeling of parallel robots based on leg orientations; Chapter 3 uses kinematic control to estimate the dynamic state (posture and velocity) of a parallel robot at high speed, and defines a versatile sensor-based computed-torque control law; Chapter 4 experimentally validates these new theoretical approaches; Finally, in the last Chapter, I conclude my thesis and offer some future possibilities.

Chapter 1

State of The Art

This chapter gives background to the evolution of the thesis so that the reader can be equipped for the next. The state-of-the-art topics are focused on modeling, control, and identification of parallel robots. We also discuss some new extensions.

1.1 Parallel Robots

1.1.1 Definition

A parallel robot can be conceptually imagined as Sir Newton's hand holding a red apple with his fingertips (see Fig. 1.1). The palm of the hand forms the base platform of the parallel robot, and the fingers act like serial robots attached to this base platform all cooperatively manipulating the red apple. Here in the parallel robot, the red apple represents a moving platform that might have a large load.

Figure 1.1 – A metaphor for a parallel robot concept: a hand holding a red apple with its fingertips. The palm of the hand forms the base platform. Fingers represent the kinematic chains. The red apple is either the moving platform or the moving platform with a large heavy load.

In a parallel robot, these serial robots are called kinematic chains and they usually contain a single motorized joint while the rest of the joints are passive. Each of these motorized joints are generally located either at the base platform (i.e., the hand's knuckle joints) or at the very first of joint locations close to the base platform. These kinematic chains are connected to the moving platform all together, and consequently they form a closed-loop mechanism. The structures of these kinematic chains are usually identical and their placements are symmetric, but they can be also different like the fingers of a hand. The number of these kinematic chains must be equal to or greater than two, "2", so that we can call it a *parallel robot*. Here, "parallel" does not imply that the kinematic chains are aligned as parallel lines, but it means rather that these kinematic chains work together to achieve a task. A parallel robot shows better dynamic performances than a serial robot in terms of speed and accuracy while manipulating both large and heavy loads [Mer00, Gog08]. Clearly, it would be difficult to eat the red apple with one finger. Figure 1.2 shows the well-known two examples of parallel robots: the Gough-Stewart platform [GW62, Ste65] and the Delta robot [Cla88, Cla91].

Figure 1.2 – The Stewart platform (left) and the Delta parallel robot (right).

1.1.2 Joint Types and Graphical Layout

The primitive joints with different degrees of freedom (dof) [KD02] are as follows:

– *Prismatic*: Slides on an axis (1 dof). It is noted by (**P**).
– *Revolute*: Rotates around an axis (1 dof). It is noted by (**R**).
– *Spherical (ball)*: Rotates around three axes (3 dof). It is a ball and noted by (**S**).

Different combinations of these joints create different types:

– *Universal*: It is composed of two revolute joints that allow two rotations (2 dof). It is noted by (**U**).
– *Spherical*: It can also be composed of three revolute joints that allow three rotations (3 dof). It is noted again by (**S**).

– *Parallelogram*: It is composed of four bars that are connected end to end by revolute joints [Cla88]. These four bars form a parallelogram shape. A parallelogram keeps an output link at fixed orientation with respect to an input link. It allows translation in three axes (3 dof) on a sphere. It is noted by (**Pa**).

A graphical layout of a robot demonstrates the positions of the actuators, of the joints and of the kinematic elements [Pie91, Kru03]. A graphical layout is composed of:

– Two bars representing a base and a moving platform.
– Boxes representing the joints. Each box has a symbol indicating the type of joint. If the joint is actuated and contains a sensor (e.g., motor encoder) then the symbol is underlined.
– Lines representing the bodies.

Figure 1.3 shows the joint-oriented graphical layouts of the parallel robots in Fig. 1.2.

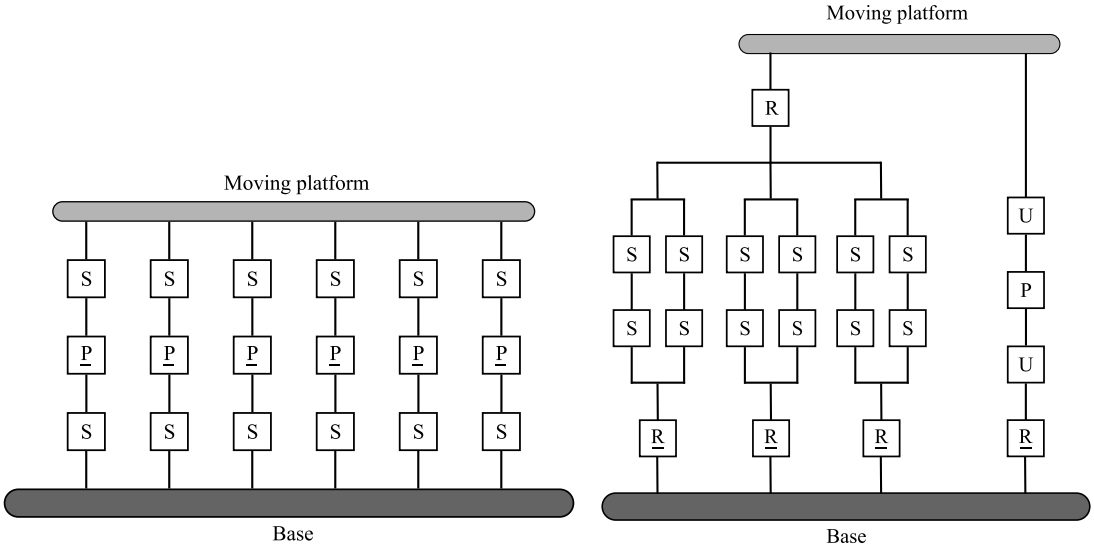


Figure 1.3 – The Gough-Stewart platform (left) and the Delta parallel robot (right) joint-oriented graphical layouts.

1.1.3 Classification

1.1.3.1 Kinematic Classification

Parallel robots are classified into three groups [Mer00] based on their motion capacities:

– *Mechanism for translation*: The Speed-R-Man robot [RLN92], the Orthoglide robot [WC00].
– *Mechanism for rotation*: The Agile Eye [GH94].
– *Mechanism for translation and rotation*: The Gough-Stewart robot [GW62, Ste65], the Delta robot [Cla88, Cla91], the Quattro robot, the 3RRR robot [GA88], the T3R1 parallel robot [Gog02].

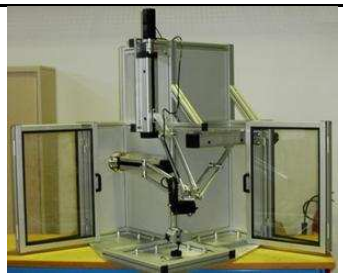
1.1.3.2 Architectural Classification

Parallel robots are also classified into three groups [GLZ+02, Ren03] based on their structure:

- Group 1: It contains a prismatic joint between the two bodies of each kinematic leg.
- Group 2: It contains a prismatic joint between the base and each kinematic leg.
- Group 3: It lacks prismatic joints.

Table 1.1 shows examples of these three groups of robots.

Table 1.1 – Architectural Classification.

Group 1	*Group* 2	*Group* 3
		
The Gough − Stewart robot	*The Orthoglide robot*	*The Quattro robot*

1.1.4 Compared with Serial Robots

1.1.4.1 Advantages

Parallel robots are, in theory, better than serial robots:

Lighter: Parallel robots usually have lighter kinematic legs because the heavy actuators are often mounted on the base platform rather than inside the kinematic legs.

Faster: Parallel robots move faster and consume less energy than the serial robots, since their kinematic legs are lighter [Cla89, Mer00, TZR99]. Consequently, for a given energetic expenditure, the acceleration of a parallel robot's end-effector is greater than a serial robot's end-effector.

Stronger: Against a disturbing force, a parallel robot's end-effector, which is supported by many kinematic legs in a closed-loop form, shows more rigidity than a serial robot's end-effector, which is supported through a single long and heavy kinematic leg [Mer00]. The longer and heavier the kinematic leg is, the weaker it becomes. In addition, the disturbing force applies a compression for the kinematic elements of a parallel robot, while for the kinematic elements of a serial robot it applies a torsion.

More Accurate: Unlike serial robots, parallel robots can be more accurate. This is because errors in the assembly of the mechanism and errors made when controlling the mechanism average out rather than accumulating [Mer00].

1.1.4.2 Disadvantages

In practice, however, parallel robots suffer from:

Limited Workspace & Singularity: Even though they are larger in size they have limited workspace [Mer00]. This is because of the closed-loop connections between the kinematic legs. This workspace also gets smaller by the presence of singularities [PK98], which makes them harder to settle for a given pose than a serial robot. Moreover, in these singularity positions, they completely lose their stiffness and become shaky.

Modeling: Their modeling is difficult because of the complex closed-loop structures and because of the existing passive joints [Mer00]. The clearances and assembly errors in the passive joints decrease the precision of the positioning of the end-effector. Calibration cannot solve this totally.

Control: It is hard to compensate for the loss of accuracy in modeling, since the passive joints are not motorized. Furthermore, as a result of the coupling among the kinematic legs, the motion of the end-effector in the Cartesian space imposes a highly non-linear dynamic behavior on the parallel robot [DC99]. In industrial environments, parallel robots are usually controlled by single axis linear controllers. These controllers can take into account neither the influences of the kinematic legs on one another nor the non-linear behavior of the dynamics.

In a high-speed trajectory tracking task, consequently, parallel robots will loose accuracy [DH06, TDH04].

1.1.5 The Duality of Parallel and Serial Robots

The fact that parallel robots use passive joints means that, in some respects, they behave oppositely to serial robots [WH91, Bru99].

– *From a configurational point of view:*

A serial robot: The active joint values and the analytical forward kinematic model define uniquely the state of the serial robot [KD02]. There are usually several solutions to the inverse kinematic problem. These solutions may not have a closed-form.

A parallel robot: The end-effector pose and the analytical inverse kinematic model define uniquely the state of the parallel robot [Mer00] (except some of the parallel robots, e.g., $3\underline{R}RR$, $3R\underline{P}R$). The analytical inverse kinematic model is usually expressed in terms of the active joint values. Nevertheless, writing it in terms of the end-effector pose is more pertinent [DAMM06, DC99]. Unlike serial robots, the forward kinematic model of a parallel robot does not always have a closed-form solution and it needs to be estimated numerically. Unfortunately, this may yield many solutions [Mer90, Hus94] and there are not very many parallel robots that are deliberately designed to have closed-form forward kinematic models [Gog04, WC00]. By mechanical construction, one can reduce this set of solutions (e.g., the Delta robot, the Quattro robot).

– *From a control point of view:*
  It is harder to master the behavior of the end-effector of a parallel robot than a serial robot. This is because of the two conflicting properties of a parallel robot: (i) the lack of a closed-form solution for a forward kinematic model that represents the actual mechanics of a parallel robot (i.e., manufacturing and assembly errors, joint clearances and backlashes, flexibilities); (ii) the end-effector can be moved only by controlling the actuators built in the base platform.
– *From a sensing point of view:*
  The previous two points of views (should) induce naturally this third one, *duality in sensing*.

  Serial robots: Since active joints represent fully the configuration, using only the motor encoders for sensing is adequate for control of serial robots.

  Parallel robots: The full configuration of parallel robots, however, cannot be expressed easily by the active joints (except a few, e.g., the Orthoglide, the Delta). On the other hand, an end-effector pose can represent the full configuration of most of the parallel robots. Thus, sensing the end-effector pose of a parallel robot rather than its active joints is more adequate [DAMM06] for control.

1.1.6 Metrological Redundancy

One of the easiest and the fastest and consequently preferred way to measure information from parallel robots is to use again the motor encoders as it is often done in serial robots. But this simplifies neither modeling (e.g., forward kinematics problem) nor the control of parallel robots as discussed above. Thus, it seems that different sensing techniques can affect fundamentally the performance of parallel robots.

Therefore, use of extra sensors (or so-called *metrological redundancy*) in modeling and control of parallel robots has appeared in the literature as complementary to actuation redundancy where extra motors added in the structure [MPC03], [MH11], [NCP12].

In [COB93], [BA95], [BTKL99] and [PCG99], redundant position sensors and the motorized joint encoders were used together to solve forward kinematic problems of parallel robots.

It is well known that the forward kinematics of the Gough-Stewart platform has 40 possible solutions [Hus94]. But if one adds sensors to the passive joints, then the solution may become straightforward (e.g., Gough-S. + length + direction of the legs).

Alternately to joint sensing, one can use exteroceptive sensing, such as vision [AMM05], [DAMM06]. In [AMM05], vision observed the mechanism legs and replaced advantageously redundant position sensors by delivering the internal state of the mechanism in the Cartesian frame. This approach was then used to servo visually the leg directions of the Gough-Stewart platform rather than its Cartesian pose. Thus, this approach proposed an original vision-based kinematic modeling and control method of parallel robots based on observation of their legs.

In [DAMM06], the Cartesian space control of the Gough-Stewart platform was performed using only vision (without using active joint positions). This was done by directly measuring the Cartesian end-effector pose of the Gough-Stewart platform by a camera observing a pattern fastened to the moving platform. Thus, the forward kinematic problem of the Gough-Stewart platform was completely removed and replaced by a computer vision system. This computer

vision system gives a unique solution and makes the control of the Gough-Stewart type mechanisms simpler than they are.

In [MCKP02], the potential of redundant sensors was also shown on a H4 parallel robot. An optical encoder and a vision system observed together the end-effector to yield a full pose information of the H4 parallel robot. Then, this measured end-effector pose was fused with the active joint positions. These fused redundant measurements enclose the H4 mechanism at both ends (from base and end-effector). Consequently, internal mechanical errors were compensated and the control accuracy was enhanced.

1.2 Modeling

The geometry, kinematics, mass distribution, and acceleration of a robot define its behavior.

1.2.1 Kinematic Modeling

It would be possible to adapt one of the existing methods for the kinematic modeling of serial robots [DH55, KK86, GCB96] for use with parallel robots. However, it would be preferable to use a method that takes into account the closed-loop constraints of a parallel robot [Kru03, Viv04].

1.2.1.1 The Zeroth-Order Kinematic Models

These models describe the static relations between the configuration variables such as the tool (end-effector) pose $\mathbb{X}$ and the articular positions $\mathbf{q}$ of the robot:

Inverse Kinematic Model (IKM) This model describes the motor positions ($\mathbf{q}$), given the end-effector pose ($\mathbb{X}$) and the geometric parameters (ξ_{geo}) of the robot. For serial robots, IKM might offer several solutions:

$$\mathbf{q}_i \;=\; M^0_{I_s}(\,\mathbb{X},\,\xi_{geo}\,)\,, \quad i = 1, \ldots, n \geqslant 2 \tag{1.1}$$

On the other hand, for parallel robots except a few (e.g., 3RRR), IKM yields a unique solution:

$$\mathbf{q} \;=\; M^0_{I_p}(\,\mathbb{X},\,\xi_{geo}\,) \tag{1.2}$$

where $M^0_{I_s}$ and $M^0_{I_p}$ are the inverse kinematic models of a serial robot and of a parallel robot, respectively. The super-right script denotes the order of the kinematic models.

Closed-form expressibility: The IKM of a parallel robot can be expressed easily in a closed form. But the IKM of a serial robot can be expressed hardly in a closed form.

Forward Kinematic Model (FKM) This model describes the pose of the end-effector ($\mathbb{X}$), given the positions of the motors ($\mathbf{q}$) and the geometric parameters of the robot (ξ_{geo}). For serial robots, FKM yields a unique solution:

$$\mathbb{X} \;=\; M^0_{F_s}(\,\mathbf{q},\,\xi_{geo}\,) \tag{1.3}$$

where $\mathbb{X}$ is a column-array representation of the end-effector pose. On the other hand, for parallel robots, FKM might offer several solutions:

$$\mathbb{X}_i \;=\; M^0_{F_p}(\,\mathbf{q},\,\xi_{geo}\,)\,, \quad i = 1, \ldots, n \geqslant 2 \tag{1.4}$$

where $M^0_{F_s}$ and $M^0_{F_p}$ are the forward kinematic models of a serial robot and of a parallel robot, respectively.

Closed-form expressibility: The FKM of a serial robot can be expressed in a closed form, but the formulation of FKM of a parallel robot is more complicated (except a few, e.g., the Orthoglide, the Isoglide4-T3R1, GauntryTau, Delta-like). Use of additional sensors can either complete the missing information of the passive joints of a parallel robot to express its FKM in a closed form [BTKL99] or eliminate the FKM by measuring directly the end-effector pose $\mathbb{X}$ of a parallel robot [DAMM06].

Implicit Kinematic Model (ImplKM) [And06] Since previously mentioned forward and inverse kinematic models of robots might not exist as injective mappings (i.e., single input-single output), one would prefer a formulation of the kinematics under the form of an implicit kinematic model. This model combines the previous models into one implicit model, given the relations between the end-effector pose, the motor positions and the geometric parameters. It is a holonomic constraint for any relevant state of the variables:

$$M^0_{\phi}(\,\mathbb{X},\,\mathbf{q},\,\xi_{geo}\,) \;=\; 0 \tag{1.5}$$

where M^0_{ϕ} is the implicit kinematic model of a robot.

Redundant Implicit Kinematic Model (RImplKM) Here, we propose a more generic model by deduction from the implicit model (1.5). Assuming that the robot is equipped with different type(s) of sensor(s) providing redundant measurements $\mathbf{r}$, we rewrite (1.5) as follows:

$$M^0_{\phi R}(\,\mathbb{X},\,\mathbf{q},\,\mathbf{r},\,\xi_{sensor},\,\xi_{geo}\,) = 0 \tag{1.6}$$

where ξ_{sensor} is the parameter vector of the sensor(s) and $M^0_{\phi R}$ is the redundant implicit kinematic model. Equation (1.6) can be written in a more compact form by assembling $\mathbf{q}$ and $\mathbf{r}$ in a single measurement vector $\mathbf{s}$ as follows:

$$M^0_{\phi R}(\,\mathbb{X},\,\mathbf{s},\,\xi_{sensor},\,\xi_{geo}\,) = 0 \tag{1.7}$$

where $\mathbf{s}$ contains all the signals of the sensors. The sensor(s) can be proprioceptive (e.g., motor encoders), or exteroceptive (e.g., a camera), or a combination of these. Consequently, this can allow $\mathbf{s}$ to be chosen as bijective (minimal) or surjective (redundant) to the end-effector pose. For example, the possible choices for $\mathbf{s}$ may be as follows:

$$\mathbf{s} \in \{\,\mathbf{q},\,\{\mathbf{q},\,\rho\},\,\{\underline{\mathbf{x}}_1, \underline{\mathbf{x}}_2,\,\ldots,\,\underline{\mathbf{x}}_k\},\;\ldots\;,\;\ldots\,\} \tag{1.8}$$

where ρ is a variable set completing the motor positions for a unique solution of the end-effector pose, and where $\{\underline{\mathbf{x}}_1, \underline{\mathbf{x}}_2,\,\ldots,\,\underline{\mathbf{x}}_k\}$ is the set of unit vectors showing the 3D directions of the bodies in a robot, and so on. The rest of this thesis will extensively discuss this issue.

1.2.1.2 The First-Order Kinematic Models

The time differentiation of the previous kinematic models gives the differential kinematic models relating the Cartesian velocity of the end-effector pose to the speed of the motors, or more precisely to the velocity of the sensor signal $\mathbf{s}$:

Inverse Differential Kinematic Model (IDKM) This model gives the speed of motors, given the end-effector pose and velocity, and as well as the geometric parameters of the robot:

$$\dot{\mathbf{q}} = \frac{\partial M_I^0(\mathbb{X}, \xi_{geo})}{\partial \mathbb{X}} \dot{\mathbb{X}} \tag{1.9}$$

$$\dot{\mathbf{q}} = \frac{\partial M_I^0(\mathbb{X}, \xi_{geo})}{\partial \mathbb{X}} L_{\mathbb{X}} \boldsymbol{\zeta} \quad or \quad \dot{\mathbf{q}} = M_I^1(\mathbb{X}, \xi_{geo}) L_{\mathbb{X}} \boldsymbol{\zeta} \tag{1.10}$$

where M_I^1 is the inverse differential kinematic model of a robot. This model is necessary for all the controls taking explicitly into account the non-linear couplings between the joints.

Closed-form expressibility: The IDKM of a parallel robot can be usually expressed in a closed form. On the other hand, expressing IDKM of a serial robot is more complicated.

Forward Differential Kinematic Model (FDKM) This model gives the velocity of the end-effector pose, given the positions and speeds of the motors and as well as the geometric parameters of the robot:

$$\dot{\mathbb{X}} = \frac{\partial M_F^0(\mathbf{q}, \xi_{geo})}{\partial \mathbf{q}} \dot{\mathbf{q}} \tag{1.11}$$

where $\mathbb{X}$ is a chosen representation for the end-effector pose. The previous forward differential kinematic model can be associated to the kinematic twist $\boldsymbol{\zeta}$, which defines the instantaneous motion of the end-effector with respect to the base of the robot. $\boldsymbol{\zeta}$ is composed of a translational velocity $\boldsymbol{v}$ and a rotational velocity $\boldsymbol{\omega}$. The chosen $\dot{\mathbb{X}}$ can be expressed in terms of a representation dependant matrix $L_{\mathbb{X}}$ relating the partial derivative of the pose to the kinematic twist $\boldsymbol{\zeta}$ [Ang97]:

$$\dot{\mathbb{X}} = L_{\mathbb{X}} \boldsymbol{\zeta} = L_{\mathbb{X}} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\omega} \end{bmatrix} \tag{1.12}$$

Then, one can rewrite (1.11) with the kinematic twist $\boldsymbol{\zeta}$ as follows:

$$\boldsymbol{\zeta} = L_{\mathbb{X}}^{-1} \frac{\partial M_F^0(\mathbf{q}, \xi_{geo})}{\partial \mathbf{q}} \dot{\mathbf{q}} \quad or \quad \boldsymbol{\zeta} = L_{\mathbb{X}}^{-1} M_F^1(\mathbf{q}, \xi_{geo}) \dot{\mathbf{q}} \tag{1.13}$$

where M_F^1 is the forward differential kinematic model of a robot. This model can be used for both simulation and prediction.

Closed-form expressibility: The FDKM of a parallel robot cannot be expressed easily in a closed form. The FDKM is computed usually by inverting numerically the IDKM of a parallel robot. On the other hand, the FDKM of a serial robot can be expressed in a closed form.

Implicit Differential Kinematic Model (ImplDKM) [And06] The differentiation of (1.5) yields the complete implicit differential kinematic model:

$$\frac{\partial\, M^0_\phi(\,\mathbb{X},\,\mathbf{q},\,\xi_{geo}\,)}{\partial\,\mathbb{X}}\;\dot{\mathbb{X}}\;+\;\frac{\partial\, M^0_\phi(\,\mathbb{X},\,\mathbf{q},\,\xi_{geo}\,)}{\partial\,\mathbf{q}}\;\dot{\mathbf{q}}\;=\;\mathbf{0} \tag{1.14}$$

or equivalently,

$$M^1_{\phi(\,\mathbb{X})}\,\dot{\mathbb{X}}\;+\;M^1_{\phi(\mathbf{q})}\,\dot{\mathbf{q}}\;=\;\mathbf{0} \tag{1.15}$$

In the case of a flexible robot, one must add the term relating the variations of the geometric parameters:

$$\frac{\partial\, M^0_\phi(\,\mathbb{X},\,\mathbf{q},\,\xi_{geo}\,)}{\partial\,\mathbb{X}}\;\dot{\mathbb{X}}+\frac{\partial\, M^0_\phi(\,\mathbb{X},\,\mathbf{q},\,\xi_{geo}\,)}{\partial\,\mathbf{q}}\;\dot{\mathbf{q}}+\frac{\partial\, M^0_\phi(\,\mathbb{X},\,\mathbf{q},\,\xi_{geo}\,)}{\partial\,\xi_{geo}}\;\dot{\xi}_{geo}\;=\;\mathbf{0} \tag{1.16}$$

or equivalently,

$$M^1_{\phi(\,\mathbb{X})}\,\dot{\mathbb{X}}\;+\;M^1_{\phi(\mathbf{q})}\,\dot{\mathbf{q}}\;+\;M^1_{\phi(\xi_{geo})}\,\dot{\xi}_{geo}\;=\;\mathbf{0} \tag{1.17}$$

This model has 3 matrices: the differential Cartesian kinematic matrix $M^1_{\phi(\,\mathbb{X})}$; the differential articular kinematic matrix $M^1_{\phi(\mathbf{q})}$; and the sensibility matrix $M^1_{\phi(\xi_{geo})}$. Rewriting (1.15) and (1.17) in a matrix-vector form:

$$\left[\;M^1_{\phi(\,\mathbb{X})}\quad M^1_{\phi(\mathbf{q})}\;\right]\left[\;\begin{array}{c}\dot{\mathbb{X}}\\ \dot{\mathbf{q}}\end{array}\;\right]\;=\;\mathbf{0} \tag{1.18}$$

$$\left[\;M^1_{\phi(\,\mathbb{X})}\quad M^1_{\phi(\mathbf{q})}\quad M^1_{\phi(\xi_{geo})}\;\right]\left[\;\begin{array}{c}\dot{\mathbb{X}}\\ \dot{\mathbf{q}}\\ \dot{\xi}_{geo}\end{array}\;\right]\;=\;\mathbf{0} \tag{1.19}$$

we get the implicit differential kinematic model matrix M^1_ϕ and the flexible implicit differential kinematic model matrix $M^1_{\phi_f}$:

$$M^1_\phi\;=\;\left[\;M^1_{\phi(\,\mathbb{X})}\quad M^1_{\phi(\mathbf{q})}\;\right] \tag{1.20}$$

$$M^1_{\phi_f}\;=\;\left[\;M^1_{\phi(\,\mathbb{X})}\quad M^1_{\phi(\mathbf{q})}\quad M^1_{\phi(\xi_{geo})}\;\right] \tag{1.21}$$

For a rigid robot, one can write from (1.18) the forward (M^1_F) and the inverse (M^1_I) differential kinematic models as follows:

$$M^1_F\;=\;-\left(\,M^1_{\phi(\,\mathbb{X})}\right)^{\dagger}\,M^1_{\phi(\mathbf{q})}\,,\qquad M^1_I\;=\;-\left(\,M^1_{\phi(\mathbf{q})}\right)^{\dagger}\,M^1_{\phi(\,\mathbb{X})} \tag{1.22}$$

provided that $M^1_{\phi(\,\mathbb{X})}$ and $M^1_{\phi(\mathbf{q})}$ have full rank.

Redundant Implicit Differential Kinematic Model (RImplDKM) Similarly to the previous part (ImplDKM), the differentiation of (1.7) yields the complete redundant implicit differential kinematic model:

$$M^1_{\phi R(\,\mathbb{X})}\,\dot{\mathbb{X}}\;+\;M^1_{\phi R(\mathbf{s})}\,\dot{\mathbf{s}}\;=\;\mathbf{0} \tag{1.23}$$

or equivalently,

$$\left[\begin{array}{cc} M^1_{\phi R(\mathbb{X})} & M^1_{\phi R(\mathbf{s})} \end{array} \right] \left[\begin{array}{c} \dot{\mathbb{X}} \\ \dot{\mathbf{s}} \end{array} \right] = \mathbf{0} \tag{1.24}$$

where $M^1_{\phi R(\mathbb{X})}$ and $M^1_{\phi R(\mathbf{s})}$ are the differential Cartesian kinematic matrix and the differential sensor(s) signal kinematic matrix, respectively. In the case of a flexible robot equipped with dynamic sensors (e.g., a moving and/or zooming camera), the terms relating the variations of the geometric parameters of the robot and the variations of the sensor parameters must be taken into account. Thus, the differentiation of (1.7) yields:

$$M^1_{\phi R(\mathbb{X})}\,\dot{\mathbb{X}} + M^1_{\phi R(\mathbf{s})}\,\dot{\mathbf{s}} + M^1_{\phi R(\xi_{sensor})}\,\dot{\xi}_{sensor} + M^1_{\phi R(\xi_{geo})}\,\dot{\xi}_{geo} = \mathbf{0} \tag{1.25}$$

or equivalently,

$$\left[\begin{array}{cccc} M^1_{\phi R(\mathbb{X})} & M^1_{\phi R(\mathbf{s})} & M^1_{\phi R(\xi_{sensor})} & M^1_{\phi R(\xi_{geo})} \end{array} \right] \left[\begin{array}{c} \dot{\mathbb{X}} \\ \dot{\mathbf{s}} \\ \dot{\xi}_{sensor} \\ \dot{\xi}_{geo} \end{array} \right] = \mathbf{0} \tag{1.26}$$

where $M^1_{\phi R(\xi_{sensor})}$ and $M^1_{\phi R(\xi_{geo})}$ are the differential sensor kinematic matrix and the sensibility matrix, respectively.

1.2.1.3 Singularities

Equation (1.17) yields three type of singularities [And06]:

Articular Singularity: It happens when the differential articular kinematic matrix is singular:

$$\{(\mathbb{X},\, \mathbf{q},\, \xi_{geo}) | M^1_{\phi(\mathbf{q})}(\mathbb{X},\, \mathbf{q},\, \xi_{geo})\,\dot{\mathbf{q}} = \mathbf{0}\} \tag{1.27}$$

and as well as where a motion of the active articular joints neither changes the end-effector pose nor deforms the robot.

Cartesian Singularity: It happens when the differential Cartesian kinematic matrix is singular:

$$\{(\mathbb{X},\, \mathbf{q},\, \xi_{geo}) | M^1_{\phi(\mathbb{X})}(\mathbb{X},\, \mathbf{q},\, \xi_{geo})\,\dot{\mathbb{X}} = \mathbf{0}\} \tag{1.28}$$

and as well as where a motion of the end-effector neither changes the active articular positions nor deforms the robot.

Sensibility Singularity: It happens when the sensibility matrix is singular:

$$\{(\mathbb{X},\, \mathbf{q},\, \xi_{geo}) | M^1_{\phi(\xi_{geo})}(\mathbb{X},\, \mathbf{q},\, \xi_{geo})\,\dot{\xi}_{geo} = \mathbf{0}\} \tag{1.29}$$

In these configurations, deformations or variations in the geometric parameters of the robot do not have any effect on the kinematic behavior of the robot. Reconfigurable robots are not considered here.

In the rigid body assumption, the articular and cartesian singularities correspond respectively to the well known serial and parallel singularities of parallel robots [GA90, CW98].

1.2.2 Dynamic Modeling

A dynamic model relates the active forces acting on a robot to the accelerations they cause, or the other way around. These active forces can be both moments in rotation and forces in translation.

Inverse Dynamic Model (IDM) An inverse dynamic model computes the torques and/or forces that the actuators of the robot must deliver to make the end-effector move in a certain way. It is used for control of robot motions and forces:

$$\mathbf{\Gamma} = A(\mathbf{s})\,\ddot{\mathbf{s}} + \mathbf{h}(\mathbf{s}, \dot{\mathbf{s}}) + \mathbf{\Gamma}_f \tag{1.30}$$

where $\mathbf{\Gamma}$ is the force vector of the actuators, A is the inertia matrix of the robot, $\mathbf{h}$ is the vector of the centrifugal, the Coriolis and the gravity forces, $\mathbf{\Gamma}_f$ is the vector of the friction forces, and $\mathbf{s}$ is the state variable vector of the robot. In the case of a serial robot, the inverse dynamic model is usually written in terms of the joint positions $\mathbf{q}$, because the joint positions uniquely define the posture of a serial robot:

$$\mathbf{\Gamma} = A(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{\Gamma}_f \tag{1.31}$$

or equivalently,

$$\mathbf{\Gamma} = IDM(\ddot{\mathbf{q}}, \dot{\mathbf{q}}, \mathbf{q}, \xi_{geo}, \xi_{dyn}) \tag{1.32}$$

where ξ_{dyn} denotes the dynamic parameters of the robot. Since the methods for modeling of parallel robots are adopted from serial robots, the inverse dynamic model of a parallel robot is generally written in terms of the articular positions $\mathbf{q}$ [KD02] too. It is, however, preferable to write the inverse dynamic model of a parallel robot with the end-effector pose $\mathbb{X}$, since in most cases it is the end-effector pose which defines uniquely the posture of a parallel robot:

$$\mathbf{\Gamma} = IDM(\ddot{\mathbb{X}}, \dot{\mathbb{X}}, \mathbb{X}, \xi_{geo}, \xi_{dyn}) \tag{1.33}$$

This inverse dynamic model is used for control purposes and ideally it should be as precise and as simple as possible.

Forward Dynamic Model (FDM) A forward dynamic model computes the accelerations of the state variables for given forces, positions and velocities. It is also known as a *direct dynamic model*. It is mainly used for simulation. A forward dynamic model is linearly extracted from (1.30) as follows:

$$\ddot{\mathbf{s}} = A^{-1}(\mathbf{s})\,(\mathbf{\Gamma} - \mathbf{h}(\mathbf{s}, \dot{\mathbf{s}}) - \mathbf{\Gamma}_f) \tag{1.34}$$

or, equivalently, it can be noted as follows:

$$\ddot{\mathbf{s}} = FDM(\mathbf{\Gamma}, \dot{\mathbf{s}}, \mathbf{s}, \xi_{geo}, \xi_{dyn}) \tag{1.35}$$

For serial robots, (1.35) is again written in terms of the joint positions $\mathbf{q}$, since the joint positions define uniquely the posture of a serial robot:

$$\ddot{\mathbf{q}} = FDM(\mathbf{\Gamma}, \dot{\mathbf{q}}, \mathbf{q}, \xi_{geo}, \xi_{dyn}) \tag{1.36}$$

And for parallel robots, (1.35) is again written in terms of the end-effector pose $\mathbb{X}$, since in most cases the end-effector pose defines uniquely the posture of a parallel robot:

$$\ddot{\mathbb{X}} \, = \, FDM(\, \mathbf{\Gamma}, \, \dot{\mathbb{X}}, \, \mathbb{X}, \, \xi_{geo}, \, \xi_{dyn} \,) \tag{1.37}$$

Implicit Dynamic Model (ImplDM) Here, we introduce an implicit dynamic model which actually corresponds to the equations of motion (EoM) of a robot. We note it as follows:

$$\mathbf{0} \, = \, ImplDM(\, \mathbf{\Gamma}, \ddot{\mathbf{s}}, \, \dot{\mathbf{s}}, \, \mathbf{s}, \, \xi_{sensor}, \, \xi_{geo}, \, \xi_{dyn} \,) \tag{1.38}$$

where $\mathbf{s}$ is the set of the measured variables which can express uniquely the state of a robot. The definition of $\mathbf{s}$ is similar to (1.8), and here we augment it with the end-effector pose $\mathbb{X}$ of a robot:

$$\mathbf{s} \in \{ \, \mathbb{X}, \, \mathbf{q}, \, \{\mathbf{q}, \, \rho\}, \, \{\underline{\mathbf{x}}_1, \underline{\mathbf{x}}_2, \, \ldots, \, \underline{\mathbf{x}}_k\}, \, \ldots \, , \, \ldots \, \} \tag{1.39}$$

where again ρ is a complementary set of variables enriching the motor positions for the unique solution of a robot posture, and $\{\underline{\mathbf{x}}_1, \underline{\mathbf{x}}_2, \, \ldots, \, \underline{\mathbf{x}}_k\}$ is the set of unit vectors showing the 3D directions of the bodies in a robot, and so on.

1.2.3 Dynamic Modeling Methods of Serial Robots

Many methods exist for dynamic modeling of serial robots, such as the Euler-Lagrange method, the recursive Newton-Euler method, the d'Alembert method and Kane's method. Now, we shall describe briefly these methods.

1.2.3.1 Euler-Lagrange Method

The Euler-Lagrange method [Lag87] exploits the principle of conservation of energy in a mechanism to derive the dynamic equations. The equations are in analytic and closed-form. The Euler-Lagrange equations are obtained by differentiating the Lagrangian function:

$$L(\mathbf{q} \, , \, \dot{\mathbf{q}}) \, = \, T(\mathbf{q} \, , \, \dot{\mathbf{q}}) \, - \, V(\mathbf{q}) \tag{1.40}$$

which yields:

$$\frac{d}{dt} \left(\frac{\partial \, L}{\partial \, \dot{\mathbf{q}}} \right) \, - \, \frac{\partial \, L}{\partial \, \mathbf{q}} \, = \, \frac{d}{dt} \left(\frac{\partial \, T}{\partial \, \dot{\mathbf{q}}} \right) - \left(\frac{\partial \, T}{\partial \, \mathbf{q}} \, - \, \frac{\partial \, V}{\partial \, \mathbf{q}} \right) \, = \, \mathbf{\Gamma} \tag{1.41}$$

where T and U are the kinematic and potential energies. This method is good for study of the dynamic properties and analysis of control schemes. However, since it is an energy-based method, it is geometrically less intuitive and it is reported to be computationally inefficient.

1.2.3.2 Newton-Euler Method / Luh-Walker-Paul's Algorithm

The Newton-Euler method [Pau81, LWP80a] exploits the balance of forces and torques in a mechanism to derive the dynamic equations. The equations are in numeric and recursive-form. This method uses the Newtonian equations of motion:

$$\mathbf{f} = m\,\mathbf{a}\,, \qquad \boldsymbol{\tau} = \mathcal{I}^T\,\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\mathcal{I}^T\,\boldsymbol{\omega}) \tag{1.42}$$

where $\mathbf{f}$, $\boldsymbol{\tau}$, m, $\mathbf{a}$, $\boldsymbol{\omega}$ and $\mathcal{I}$ are the linear momentum, the angular momentum, the mass, the linear acceleration, the angular velocity and the inertia of the body, respectively. The method calculates the dynamics through two loops:

- *Forward Loop*: moves from base to end to evaluate the velocities and the accelerations.
- *Backward Loop*: moves from end to base to compute the forces and torques.

It is systematic and efficient for real time implementation of the control schemes.

1.2.3.3 d'Alembert Method / Principle of Virtual Work

This method exploits the principle of conservation of virtual work [lRd43] in a mechanism to derive the dynamic equations. It states that the sum of differences in work, resulting from either virtual forces acting through a real displacement or real forces acting through a virtual displacement, is zero. The displacement is *infinitesimal* and is *consistent with constraints* on the system:

$$0 = \sum_i \left(\mathbf{f}_i - m_i\,\mathbf{a}_i\right)^T\ \delta\mathbf{x}_i \tag{1.43}$$

where $\mathbf{f}$ is an applied force, m is the mass of a particle, $\mathbf{a}$ is the acceleration of a particle, $\delta\mathbf{x}$ is an infinitesimal displacement consistent with the constraints, and i enumerates a particular particle in the system.

1.2.3.4 Kane's Method

Kane's method [KL85] actually has the Lagrange form of d'Alembert principle and offers many advantages, while obtaining the equations of motion of a system. It needs neither the use of energy functions nor consequently their differentiation problem. It uses the generalized forces where the non-contributing forces are directly eliminated by projection. It allows the choice of different variables other than the generalized coordinates, which can have a significant effect on the resulting equations of motion. This method is also more useful for multi-body systems. Now, we introduce briefly the basic equations in Kane's method on which this work is based. Further details will be given in Chapter 2.

Let $\{f^*_{u_r}, f_{u_r}\}|_{r=1}^n$ be respectively the *generalized inertia forces* and *generalized active forces* for a system with n degrees of freedom, and be given as:

$$f^*_{u_r} = \sum_{k=1}^{p} \left(\left(\frac{\partial \mathbf{v}_{c_k}}{\partial u_r}\right)^T \mathbf{f}_{in_k} + \left(\frac{\partial \omega_k}{\partial u_r}\right)^T \boldsymbol{\tau}_{in_k} \right)\ ,\ \ r = 1, \ldots, n \tag{1.44}$$

$$f_{u_r} = \sum_{k=1}^{p} \left(\left(\frac{\partial \mathbf{v}_{c_k}}{\partial u_r}\right)^T \mathbf{f}_k + \left(\frac{\partial \omega_k}{\partial u_r}\right)^T \boldsymbol{\tau}_k \right)\ ,\quad r = 1, \ldots, n \tag{1.45}$$

where p is the number of rigid bodies, u_r is a *generalized speed*, $\{\frac{\partial \mathbf{v}_{c_k}}{\partial u_r}\,,\,\frac{\partial \omega_k}{\partial u_r}\}$ are so-called partial linear and angular velocities, $\{\mathbf{f}_{in_k}, \boldsymbol{\tau}_{in_k}\}$ are the *inertia force* and the *inertia torque* generated by the accelerated masses and inertias acting on the k^{th} body, $\{\mathbf{f}_k, \boldsymbol{\tau}_k\}$ are the *resultant force* and the *resultant torque* that are equivalent to a set of contact and distance forces acting on the k^{th} body. In order to have the equations of motion, namely Kane's dynamical equations, one just needs to add the generalized inertia and active forces and equate them to zero:

$$f^*_{u_r} + f_{u_r} = 0\,, \quad r = 1, \ldots, n \tag{1.46}$$

1.2.4 Dynamic Modeling Methods of Parallel Robots

All the methods for serial robots can be adapted to parallel robots. In addition to those methods, there exist one method designed for parallel robots in particular. Now, we explain briefly this method.

1.2.4.1 Khalil's Method

Khalil proposed [KI04] to obtain the equations of motion of a parallel robot by extending the systematic approach of the modeling of a serial robot. This approach proceeds as follows: (i) Firstly, each of the kinematic legs of a parallel robot is considered as an independent serial robot and the inverse dynamic model of this kinematic leg is written using one of the methods proposed for modeling of serial robots; (ii) Then, the equilibrium of all the efforts (torques and forces) applied on the moving platform is calculated. These efforts come from each of the kinematic legs, from the acceleration of the moving platform and from the external forces (weight, contact, etc.); (iii) Finally, this total effort collected on the moving platform is projected onto the active joints.

This approach is strongly intuitive for handling of the kinematic constraints, because it allows each leg to contribute for the total effort on the moving platform:

$$\boldsymbol{\Gamma} \;=\; FDKM^T_{robot} \left(\mathcal{W}_{platform} \;+\; \sum_{i=1}^{N_{legs}} \left(J_i^T \; IDKM^T_{leg(i)} \; IDM_{leg(i)} \right) \right) \tag{1.47}$$

where $FDKM_{robot}$ is the forward differential kinematic model of the parallel robot, $\mathcal{W}_{platform}$ is the wrench vector for the dynamics of the moving platform, J_i is the Jacobian matrix relating the velocity of the terminal point of the i^{th} leg to the end-effector velocity, $IDKM_{leg(i)}$ is the inverse differential kinematic model of the i^{th} leg, $IDM_{leg(i)}$ is the inverse dynamic model of the i^{th} leg. The $IDM_{leg(i)}$ can be computed with any of the existing methods for serial robots.

1.3 Control

Motion control is concerned with moving the end-effector of the robot to the desired position with a desired velocity profile. Motion control is usually performed by feeding back the state of the robot, which is called either closed-loop control or feedback control. Closed-loop motion control of robots is either at kinematic level or dynamic level. In kinematic control

the active forces and masses are not taken into account. However, when a high-speed motion is considered, these forces and masses severely perturb the movement of the robot. And at this point a dynamic control, which can handle these perturbations, is required. Closed-loop control is performed either directly in a sensor space or in an augmented space supported by some auxiliary models. In the next two subsections, we will present kinematic and dynamic controls.

1.3.1 Kinematic Control

Some of the well known sensor-based kinematic controls are joint-space kinematic control, Cartesian-space kinematic control, and visual servoing. Control in the joint space is a sensor-space control where the error is regulated over the measurements supplied by the motor encoders. In the case of Cartesian-space control, two scenarios are possible: (i) if the Cartesian end-effector pose is measured directly by a sensor, then it can be considered as sensor-space control; (ii) if the Cartesian end-effector pose is computed with the help of an additional algorithm, then it can be considered as a model-space control. Lastly, visual servoing is a specific sensor-space control where the sensor is a camera and the error is defined with some extracted image features.

In a kinematic control, the control input for a robot is the velocity of the articular positions $\dot{\mathbf{q}}$, and the observed output is usually the articular positions $\mathbf{q}$. Therefore, the robot is generally considered as an integrator. Figure 1.4 shows the block representation of a robot in a kinematic control.

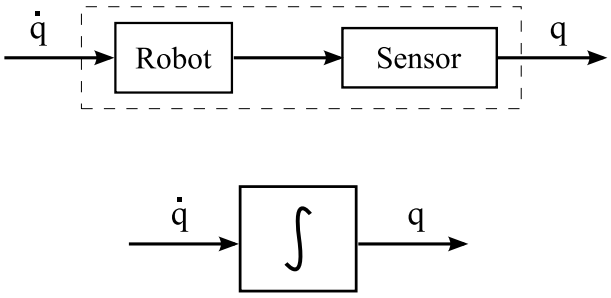


Figure 1.4 – Robot model is assumed to be an integrator in a kinematic control.

In a sensor-space kinematic control [SLE91] (see Fig. 1.5), let $\mathbb{X}$ be a representation of the pose of the end-effector of the robot at time t and let $\mathbf{s}$ be a sensor signal depending on $\mathbb{X}$ and as well as on t:

$$\mathbf{s} = \mathbf{s}(\mathbb{X}(t),\, t) \tag{1.48}$$

then, the error function $\mathbf{e}$ for the task is given as below:

$$\mathbf{e} = \mathbf{e}(\mathbb{X}(t),\, t) = C\,(\,\mathbf{s}(\mathbb{X}(t),\, t) - \mathbf{s}^*(t)\,) \tag{1.49}$$

where C is a combination matrix of the current sensor signal $\mathbf{s}$ and the desired value of the sensor signal $\mathbf{s}^*(t)$. Afterwards, the control law that will minimize the error is built upon the

following assumptions: (i) the sensor signal $\mathbf{s}(\mathbb{X}(t),\, t)$ is observable during the task; (ii) the variations of the matrix C are negligible, $\dot{C} \ll I$, where I is an identity matrix; (iii) the reference trajectory $\mathbf{s}^*(t)$ corresponds to reachable poses; and (iv) the sensor signal $\mathbf{s}(\mathbb{X}(t),\, t)$ is an injective mapping from $SE(3)$ to the sensor space.

Taking the previous assumptions into consideration, the differentiation of the error function yields:

$$\dot{\mathbf{e}} \,=\, C\, \dot{\mathbf{s}} \,-\, C\, \dot{\mathbf{s}}^* \tag{1.50}$$

where the variation in the sensor signal $\mathbf{s}$ can be expressed as a function of the relative kinematic twist ($\boldsymbol{\zeta}$) of the robot pose with respect to the sensor, and of the observed scene (e.g., static or dynamic properties of shapes, colors, etc.):

$$\dot{\mathbf{s}} \,=\, \frac{\partial\, \mathbf{s}}{\partial\, \mathbb{X}}\, \dot{\mathbb{X}} \,+\, \frac{\partial\, \mathbf{s}}{\partial\, t} \,=\, L_s\, \boldsymbol{\zeta} \,+\, \frac{\partial\, \mathbf{s}}{\partial\, t} \tag{1.51}$$

with L_s $\left(= \frac{\partial\, \mathbf{s}}{\partial\, \mathbb{X}}\, L_{\mathbb{X}}\right)$ the interaction matrix that relates the variations of the sensor signal $\mathbf{s}$ to the relative kinematic twist $\boldsymbol{\zeta}$ of the end-effector of the robot, and $L_{\mathbb{X}}$ the matrix that relates the variations of the pose of the end-effector $\mathbb{X}$ of the robot to the relative kinematic twist $\boldsymbol{\zeta}$. Then, the differentiation of the error function (1.50) can be rewritten as follows:

$$\dot{\mathbf{e}} \,=\, C\, L_s\, \boldsymbol{\zeta} \,+\, C\, \frac{\partial\, \mathbf{s}}{\partial\, t} \,-\, C\, \dot{\mathbf{s}}^* \tag{1.52}$$

Assuming that the robot is a pure integrator, an exponential convergence for the error is imposed by setting its derivative to be equal to $\dot{\mathbf{e}} \,=\, -\lambda\, \mathbf{e}$ with $\lambda \,>\, 0$. Hence, one extracts the kinematic twist $\boldsymbol{\zeta}$ as below:

$$\boldsymbol{\zeta} \,=\, (\, C\, L_s\,)^{-1} \left(-\lambda\, \mathbf{e} \,-\, C\, \frac{\partial\, \mathbf{s}}{\partial\, t} \,+\, C\, \dot{\mathbf{s}}^* \right) \tag{1.53}$$

and by using $C \,=\, \widehat{L_s}^{\dagger}$ as the pseudo-inverse of an estimated L_s, the kinematic twist $\boldsymbol{\zeta}$ is calculated as follows:

$$\boldsymbol{\zeta} \,=\, (\, \widehat{L_s}^{\dagger}\, L_s\,)^{-1} \left(-\lambda\, \mathbf{e} \,-\, \widehat{L_s}^{\dagger}\, \frac{\partial\, \mathbf{s}}{\partial\, t} \,+\, \widehat{L_s}^{\dagger}\, \dot{\mathbf{s}}^* \right) \tag{1.54}$$

Since the real interaction matrix L_s is not known, its estimation $\widehat{L_s}$ should be as accurate as possible so that the multiplication $\widehat{L_s}^{\dagger}\, L_s$ can be assumed to yield the identity matrix and so that the so-called pseudo-control vector becomes:

$$\boldsymbol{\zeta} \,=\, -\lambda\, \mathbf{e} \,-\, \widehat{L_s}^{\dagger}\, \frac{\partial\, \mathbf{s}}{\partial\, t} \,+\, \widehat{L_s}^{\dagger}\, \dot{\mathbf{s}}^* \tag{1.55}$$

Assuming that

$$\frac{\partial\, \mathbf{s}}{\partial\, t} \,=\, \mathbf{0}\,, \qquad \dot{\mathbf{s}}^* \,=\, \mathbf{0} \tag{1.56}$$

the pseudo-control vector $\boldsymbol{\zeta}$ in (1.55) gives:

$$\boldsymbol{\zeta} \,=\, -\lambda\, \mathbf{e} \tag{1.57}$$

The stability of the pseudo-control vector $\boldsymbol{\zeta}$ in (1.57) can be analyzed by substituting it into the error dynamics in (1.52) with the assumptions in (1.56):

$$\dot{\mathbf{e}} = -\lambda \left(\widehat{L_s}^{\dagger} \, L_s \right) \mathbf{e} \tag{1.58}$$

if the multiplication $(\widehat{L_s}^{\dagger} \, L_s)$ is a positive-definite matrix, then the pseudo-control law is stable in the sense of Lyapunov. The last step is the conversion of the pseudo-control law to the actual control input $\dot{\mathbf{q}}$ of the robot:

$$\dot{\mathbf{q}} = \widehat{M_I^1} \, \boldsymbol{\zeta} \tag{1.59}$$

where $\widehat{M_I^1}$ is an approximate inverse differential kinematic model (IDKM) that relates end-effector kinematic twist (pseudo-control law) to the velocities of the active joints (control input). And again for stability, the multiplication of the approximated IDKM with the forward differential kinematic model M_F^1 should also be a positive-definite matrix ($\widehat{M_I^1} \, M_F^1 > 0$).

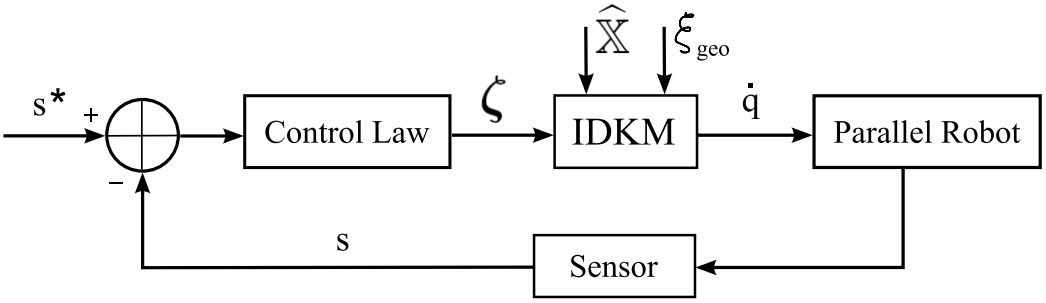


Figure 1.5 – Sensor-space kinematic control. $\mathbf{s}$ is the sensor signal, $\boldsymbol{\zeta}$ is the pseudo-control law, IDKM is the inverse differential kinematic model computed with an estimated pose of the end-effector $\widehat{\mathbb{X}}$, and $\dot{\mathbf{q}}$ is the actual control input of the parallel robot.

When the regulation of the sensor signal $\mathbf{s}$ is not enough to achieve the task, one calls for a model-space control approach (see Fig. 1.6) to enrich the sensor signal $\mathbf{s}$ with the known geometry of the scene (robot geometry, object geometry, etc.) so that the task can be accomplished. This model can be the forward kinematic model (FKM) of the robot, an algorithm for computing the relative pose of the end-effector of the robot, and etc.

In order to perform the kinematic control, the inverse differential kinematic model needs the estimated end-effector pose $\widehat{\mathbb{X}}$. The estimated end-effector pose $\widehat{\mathbb{X}}$ can be obtained from one of these options:

- $\widehat{\mathbb{X}}$ can be sometimes directly measured (laser tracking, specific motions).
- $\widehat{\mathbb{X}} = \mathbb{X}^*(t)$ can be chosen as equal to the current desired end-effector pose.
- $\widehat{\mathbb{X}} = \underset{\mathbb{X}}{\operatorname{argmin}} \| \mathbf{s} - \mathbf{s}(\mathbb{X}, t) \|$ can be computed by minimizing the error between the measured sensor signal $\mathbf{s}$ (which is an image of the current end-effector pose) and the initial guess of the sensor signal through the previously known end-effector pose.
- $\widehat{\mathbb{X}} = \underset{\mathbb{X}}{\operatorname{argmin}} \| \mathbf{q} - IKM(\mathbb{X}, \xi_{geo}) \|$ can be computed by minimizing the error between the measured motor positions $\mathbf{q}$ (which is the map of the current end-effector pose via

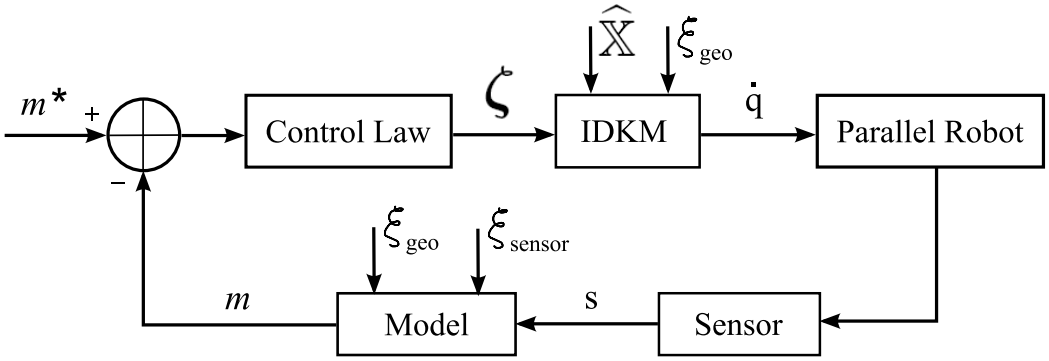


Figure 1.6 – Model-space kinematic control. $\mathbf{s}$ is the sensor signal, $\boldsymbol{\zeta}$ is the pseudo-control law and IDKM is the inverse differential kinematic model computed with an estimated pose of the end-effector $\widehat{\mathbb{X}}$, and $\dot{\mathbf{q}}$ is the actual control input of the parallel robot.

IKM) and the initial estimate of the motor positions through the previously known end-effector pose.

or one can use another convenient method.

1.3.2 Dynamic Control

The *computed-torque control* method governs the dynamic behavior of a robot based on its inverse dynamic model [LWP80b, Pau81] (see Fig. 1.8). It computes the required actuator forces which correspond to the current dynamic state of the robot. It decouples the non-linear dynamic behavior of the robot and linearizes its control. Here, we recall this method which will be later employed for control purposes. In a dynamic control, the control input is the acceleration, and the observed output is usually the position of the robot. Therefore, a robot in a dynamic control is considered as a double integrator. Figure 1.7 shows the block representation of a robot in a dynamic control.

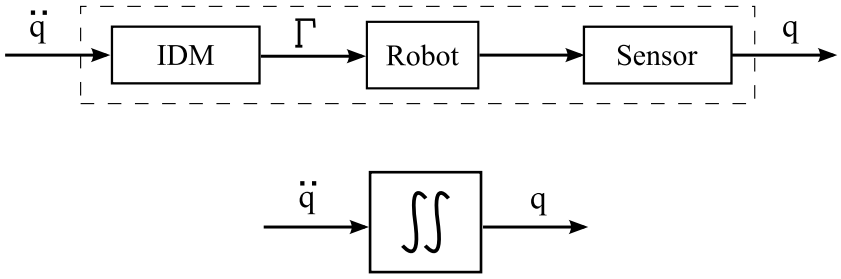


Figure 1.7 – Robot model is assumed to be a double integrator in a dynamic control.

The computed-torque control method can be simply illustrated using the Lagrange formulation of the inverse dynamic model which is expressed by means of joint positions $\mathbf{q}$, brought

from (1.31), as follows:

$$IDM: \ \boldsymbol{\Gamma} \ = \ A(\mathbf{q})\,\ddot{\mathbf{q}} \ + \ \mathbf{h}(\mathbf{q},\,\dot{\mathbf{q}}) \ + \ \boldsymbol{\Gamma}_f \tag{1.60}$$

where again $\boldsymbol{\Gamma}$ is the force vector of the actuators, A is the inertia matrix of the robot, $\mathbf{h}$ is the vector of the centrifugal, the Coriolis and the gravity forces, $\boldsymbol{\Gamma}_f$ is the vector of the friction forces.

In an ideal case, the control law $\mathbf{u}$ is equal to the acceleration $\ddot{\mathbf{q}}$ of the sensor signal. The control law $\mathbf{u}$ is usually obtained with a proportional-derivative control term $\mathbf{u}_{_{PD}}$ and with a feed-forward term of the desired signal acceleration $\mathbf{u}_{_{ff}}$:

$$\mathbf{u} \ = \ \mathbf{u}_{_{ff}} \ + \ \mathbf{u}_{_{PD}} \tag{1.61}$$

where

$$\mathbf{u}_{_{ff}} \ = \ \ddot{\mathbf{q}}^{*}\,, \qquad \mathbf{u}_{_{PD}} \ = \ K_d\,\dot{\mathbf{e}} + K_p\,\mathbf{e} \tag{1.62}$$

and where $\ddot{\mathbf{q}}^{*}$ is the desired acceleration, $K_p\,,\ K_d$ are the proportional and derivative positive control gains, $\mathbf{e} = \mathbf{q} - \mathbf{q}^{*}$ is the error between the measured signal and the desired reference signal.

Knowing that in ideal case $\mathbf{u} \ = \ \ddot{\mathbf{q}}$, the behavior of the error is characterized by the following second order form:

$$\ddot{\mathbf{e}} \ + \ K_d\,\dot{\mathbf{e}} \ + \ K_p\,\mathbf{e} \ = \ \mathbf{0} \tag{1.63}$$

In the error dynamics, the oscillation and damping can be regulated by tuning the control gains as follows:

$$K_p \ = \ \omega^2\,, \qquad K_d \ = \ 2\,\zeta\,\omega \tag{1.64}$$

with ζ a fixed damping ratio (usually between 0.9 and 1) and ω a cut-off frequency which is fixed to the highest value with respect to the mechanical resonance frequency.

Actually, the behavior of the real robot can be written in the following form [KD02]:

$$\boldsymbol{\Gamma} \ = \ (\widehat{A}(\mathbf{q}) \ + \ \widetilde{A})\,\ddot{\mathbf{q}} \ + \ (\widehat{\mathbf{h}}(\mathbf{q},\,\dot{\mathbf{q}}) \ + \ \widetilde{\mathbf{h}}) \ + \ (\widehat{\boldsymbol{\Gamma}}_f + \widetilde{\boldsymbol{\Gamma}}_f) \tag{1.65}$$

where hats $(\widehat{\cdot})$ are for the estimations and where tildes $(\widetilde{\cdot})$ are for the errors of the estimations. Since we calculate the dynamics of the robot with the estimated models, the inverse dynamic model is rewritten as below:

$$\boldsymbol{\Gamma} \ = \ \widehat{A}(\mathbf{q})\,\mathbf{u} \ + \ \widehat{\mathbf{h}}(\mathbf{q},\,\dot{\mathbf{q}}) \ + \ \widehat{\boldsymbol{\Gamma}}_f \tag{1.66}$$

This dynamic model will perturb the system due to the errors in the modeling. If it is necessary to improve the control of the robot, then one must identify the model parameters well [OP01, SGT+97]. If this does not solve the problem, either robust control methods [BHC00, HBS00, VPP03, LSCH03, BAP98, YOKN98] or adaptive control methods [Lam93] can be used. Figures 1.8 and 1.9 show the joint-space computed-torque control and the Cartesian-space computed-torque control block diagrams, respectively. The joint-space computed-torque control is better suited for serial robots, while the Cartesian-space computed-torque control is better suited for parallel robots.

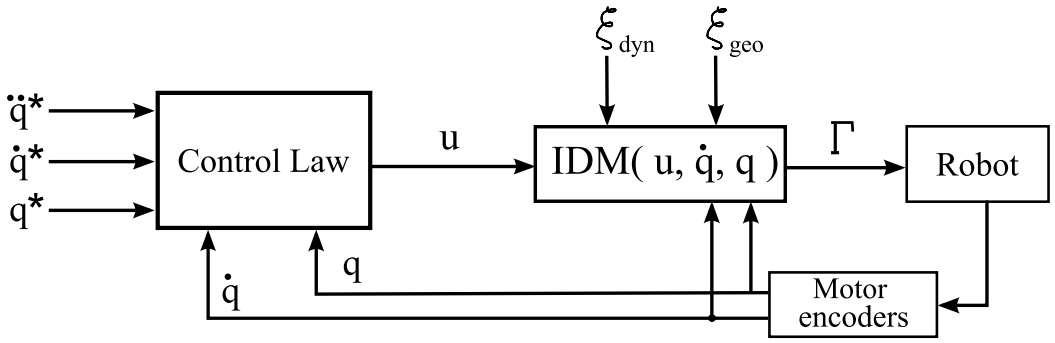


Figure 1.8 – Joint-space computed-torque control. $\mathbf{q}$ is the motor positions vector and $\mathbf{u} = \ddot{\mathbf{q}}$ is the control signal. This control is more convenient for serial robots.

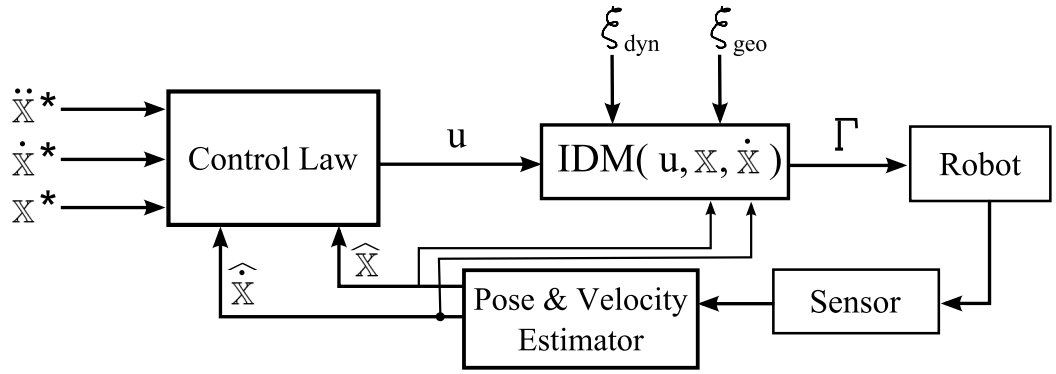


Figure 1.9 – Cartesian-space computed-torque control. $\mathbb{X}$ is the end-effector pose and $\mathbf{u} = \ddot{\mathbb{X}}$ is the control signal. This control is more convenient for parallel robots.

1.4 Identification

Identification process looks for the most accurate values of model parameters to imitate better the real robot behavior. A model of a real robot is usually built on its CAD model assuming that the dimensions and the assembly of the robot are perfect. In reality, however, there are always imperfections in the fabrication and assembly of the pieces, which mean that the model is of a lower quality than its expected accuracy. Consequently, a model identification is a necessary step for better control of robots. There are two types of model parameters to be identified: geometric and dynamic parameters of the robot.

1.4.1 Geometric Identification

A good identification of the geometric parameters ξ_{geo} of the robot improves the precision of the estimated end-effector pose via FKM. Geometric parameters are usually obtained through a non-linear minimization. There are two steps that should be taken into consideration

before the identification:

- *Configuration decision*: Firstly, the number (n) of configurations (static states) should be large enough to span the workspace of the robot as much as possible.
- *Measurements*: Afterwards, the corresponding measurement signals of these configurations should be collected $\{\, \mathbf{s}_i^m,\, \bar{\mathbf{s}}_i^m \,\}$ where $i = 1, \ldots, n$. For instance $\mathbf{s}_i^m = \mathbb{X}_i^m$ and $\bar{\mathbf{s}}_i^m = \mathbf{q}_i^m$ or vice versa. Each of the pairs $\{\, \mathbf{s}_i^m,\, \xi_{geo} \,\}$ and $\{\, \bar{\mathbf{s}}_i^m,\, \xi_{geo} \,\}$ should express a static state of the robot.

Once the initial steps are completed, the ξ_{geo} is computed (see Fig 1.10) by minimizing the following error function:

$$\widehat{\xi}_{geo} = \arg\min_{\xi_{geo}} \mathbf{error}\,(\mathbf{s}_i^m,\, \mathbf{s}_i^e)\,, \quad i = 1, \ldots, n \tag{1.67}$$

where $\mathbf{s}^e$ is the estimated signal given by the model:

$$\mathbf{s}_i^e = Model(\,\bar{\mathbf{s}}_i^m,\, \widehat{\xi}_{geo}\,) \tag{1.68}$$

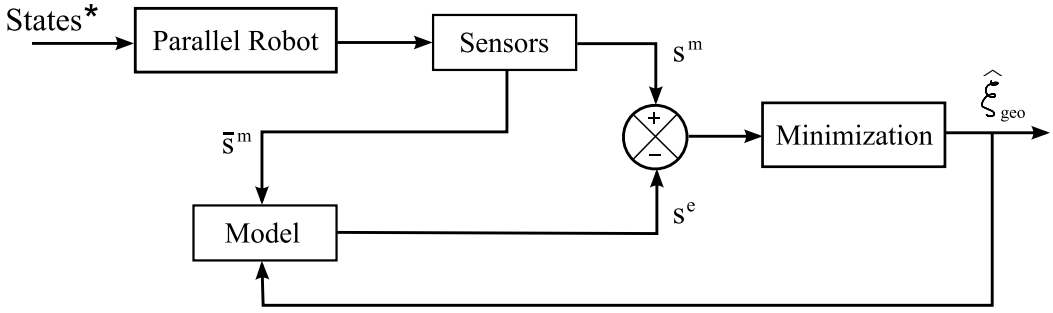


Figure 1.10 – Geometric parameter identification. $\{\, \mathbf{s}^m,\, \bar{\mathbf{s}}^m \,\}$ and $\mathbf{s}^e$ are the measured and the estimated signals, respectively. The ξ_{geo} is the geometric parameters vector. The box "sensors" can contain different sensors such as motor encoders, laser-tracker, camera, etc.

For a serial robot, the forward kinematic model (FKM) is analytically defined and it is usually used as a model for minimization through the measured end-effector pose and the articular positions:

$$\bar{\mathbf{s}}_i^m = \mathbf{q}_i^m\,, \qquad \mathbf{s}_i^m = \mathbb{X}_i^m\,, \qquad Model : FKM(\,\mathbf{q}_i^m,\, \widehat{\xi}_{geo}\,) = \mathbb{X}_i^e \tag{1.69}$$

$$\widehat{\xi}_{geo} = \arg\min_{\xi_{geo}} \sum_{i}^{n} \| \mathbb{X}_i^m - \mathbb{X}_i^e \|^2 \tag{1.70}$$

On the other hand, for a parallel robot, the inverse kinematic model (IKM) is analytically defined and it is usually chosen as a model for minimization [BK99b] through the measured end-effector pose and the articular positions:

$$\bar{\mathbf{s}}_i^m = \mathbb{X}_i^m\,, \qquad \mathbf{s}_i^m = \mathbf{q}_i^m\,, \qquad Model : IKM(\,\mathbb{X}_i^m,\, \widehat{\xi}_{geo}\,) = \mathbf{q}_i^e \tag{1.71}$$

$$\widehat{\xi}_{geo} = \arg\min_{\xi_{geo}} \sum_{i}^{n} \| \mathbf{q}_i^m - \mathbf{q}_i^e) \|^2 \tag{1.72}$$

Geometric identification methods can be classified in three groups: self-calibration methods, constrained-motion methods, and external-calibration methods. In self-calibration methods [YCYL02, HBP+05, HL95], either extra sensors are added to the passive joints or an extra passive chain is added to the mechanism. On the other hand, constrained-motion methods do not need extra sensors [RR01b, BK01, RR01a, CP04, RFS08]. Constrained-motion methods either decrease number of degrees of freedom of the moving platform or the mobility of any joint (e.g., fixing the length of a leg of the Gough-Stewart platform). And, external-calibration methods use external measurement devices, such as theodolite [WLR86], inclinometres [BK99a, RPR+06], vision [RALD06, RVA+06], laser-tracker [KAS+98, NBHW00, MTW03], and the coordinate measuring machine [Dan03, CYH06, YH05]. The minimization algorithms can vary from a classical least square method [KD02] to an interval analysis method [Dan99, DACP06].

1.4.2 Dynamic Identification

The computed-torque control has to use an inverse dynamic model (IDM) based on the dynamic parameters ξ_{dyn} and previously identified geometric parameters ξ_{geo} of the robot. That is to say, well identified dynamic parameters ξ_{dyn} improve the trajectory tracking performance of the robot. There are three steps that should be taken into consideration before the identification:

- *Trajectory design*: It is important to design a trajectory which spans the workspace with different velocities so that it can excite all the parameters in the model [GK92].
- *Fast sensor(s):* Since the identification process requires exciting (high frequency) trajectories, the sensors must be able to take measurements quickly and accurately.
- *Velocity and acceleration measurements*: Most of the robots are equipped with sensors which are reasonably precise for position information. However, it is somewhat difficult to obtain precise velocity and acceleration information. Usually, they are computed by successive numerical differentiation of the position signal which introduces high frequency noises. This, especially, makes the acceleration signal impractical. So, the position signal should be passed through a low-pass filter before the numerical differentiations [KD02, Gue03, Viv04, GP01].

Dynamic parameters can be obtained either through minimization methods (e.g., non-linear constrained optimization [FDM07], interval analysis [PRV03]) similar to geometric identification (see Fig 1.11) or through solving the simple linear system [Aea04] given below:

$$\mathbf{s}^m = W \left(\bar{\mathbf{s}}^m, \dot{\bar{\mathbf{s}}}^m, \langle \ddot{\bar{\mathbf{s}}}^m \rangle, \xi_{geo} \right) \ \xi_{dyn} \tag{1.73}$$

where $\langle \cdot \rangle$ implies that the associated variable is optional, W is an observation matrix, function of the measured signals $\{ \bar{\mathbf{s}}^m, \dot{\bar{\mathbf{s}}}^m, \langle \ddot{\bar{\mathbf{s}}}^m \rangle \}$ and previously identified geometric parameters ξ_{geo}. And $\mathbf{s}^m$ is an output signal measured by another sensor. Linear system (1.73) is written from a $Model$ of the robot which is noted as follows:

$$\mathbf{s}^e = Model(\bar{\mathbf{s}}^m, \dot{\bar{\mathbf{s}}}^m, \langle \ddot{\bar{\mathbf{s}}}^m \rangle, \xi_{geo}, \widehat{\xi}_{dyn}) \tag{1.74}$$

where $\mathbf{s}^e$ is the estimated output signal corresponding to the measured output signal $\mathbf{s}^m$. This $Model$ can be either an inverse dynamic model (IDM), or an energy model (EM), or a power model (PM) [KD02]. If it is an IDM, then (1.73) takes the form below:

$$\mathbf{\Gamma}^m \;=\; W_{IDM}\left(\,\overline{\mathbf{s}}^{\,m},\,\dot{\overline{\mathbf{s}}}^{\,m},\,\ddot{\overline{\mathbf{s}}}^{\,m},\,\xi_{geo}\,\right)\;\xi_{dyn} \tag{1.75}$$

where $\mathbf{\Gamma}^m$ is the measured actuators' forces. If it is an energy model or a power model, this time the acceleration signal does not appear in the observation matrix:

$$\int (\mathbf{\Gamma}^m)^T\;\dot{\mathbf{q}} \;=\; W_{EM}\left(\,\overline{\mathbf{s}}^{\,m},\,\dot{\overline{\mathbf{s}}}^{\,m},\,\xi_{geo}\,\right)\;\xi_{dyn} \tag{1.76}$$

$$(\mathbf{\Gamma}^m)^T\;\dot{\mathbf{q}} \;=\; W_{PM}\left(\,\overline{\mathbf{s}}^{\,m},\,\dot{\overline{\mathbf{s}}}^{\,m},\,\xi_{geo}\,\right)\;\xi_{dyn} \tag{1.77}$$

For a parallel robot, it may be more pertinent to perform the identification through the end-effector pose measurements $\{\,\mathbb{X}^m,\,\dot{\mathbb{X}}^m,\,\langle\,\ddot{\mathbb{X}}^m\,\rangle\,\}$:

$$\mathbf{s}^m \;=\; W(\,\mathbb{X}^m,\,\dot{\mathbb{X}}^m,\,\langle\,\ddot{\mathbb{X}}^m\,\rangle,\,\xi_{geo}\,)\;\xi_{dyn} \tag{1.78}$$

because the end-effector pose can be obtained by vision (an exteroceptive sensor) and because it is analytically simpler to write the models from the Cartesian space to articular space.

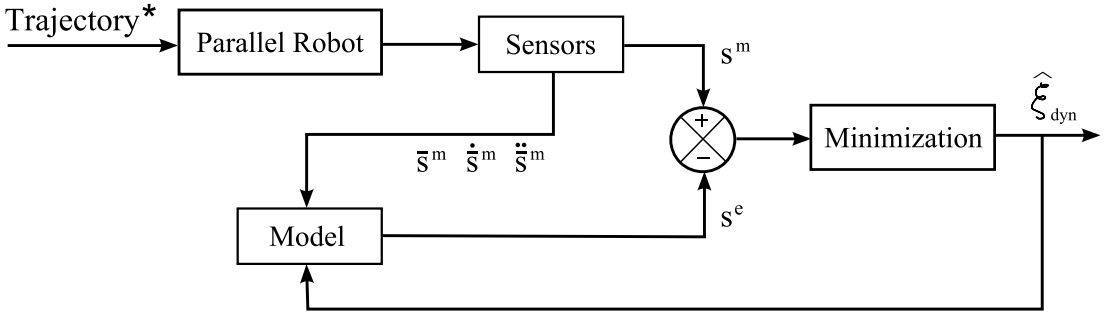


Figure 1.11 – Dynamic parameter identification. $\{\,\mathbf{s}^m,\,\overline{\mathbf{s}}^{\,m},\,\dot{\overline{\mathbf{s}}}^{\,m},\,\ddot{\overline{\mathbf{s}}}^{\,m}\,\}$ and $\mathbf{s}^e$ are the measured and the estimated signals, respectively. ξ_{dyn} is the dynamic parameters vector.

1.5 MICMAC

1.5.1 Introduction

MICMAC is the acronym of «Modélisation, Identification et Commande des MAchines Complexes», but it also fits to the English translation «Modeling, Identification and Control of complex MAChines». MICMAC is one of the research topics of the ROSACE (RObotique et Systèmes Autonomes ComplexEs) team at the laboratory of LASMEA/ISPR of Pascal Institute. It is focused on the *high-speed vision-based modeling, identification and control of parallel robots*.

The MICMAC project first started in kinematics, then progressed to dynamics. It did so in two parts: firstly *modular* MICMAC and secondly *integrated* MICMAC. The modeling, control, and identification modules of parallel robots have been often treated separately so far by being adapted from the customary approaches of the serial robots. It can be, however, difficult to apply these modules for parallel robots as they do not have sensors in their passive joints. The modular MICMAC makes these processes easier by using vision sensor(s) to complete the missing Cartesian geometry of the mechanism.

This is, however, still not enough for efficient control of parallel robots. That is because parallel robots are a lot faster and much more architecturally complex than serial robots. Furthermore, parallel robots work in duality, unlike serial robots. Consequently, it would be better to develop new methodologies and sensors for precise and simple control of parallel robots. On the other hand, the integrated MICMAC is concerned with the optimization of modeling, control and identification modules regarding the performance of a mechanism.

In next two subsections, we introduce briefly the modular MICMAC and the integrated MICMAC approaches.

1.5.2 Modular MICMAC

The following people have made important contributions to the modular MICMAC. Their works are discussed in chronological order: (i) In the first part of his Ph.D. thesis [Ren03], P. Renaud worked on geometric identification of parallel robots based on observation of the end-effector pose by vision; (ii) In the first part of his Ph.D. thesis [Dal07], T. Dallej worked on kinematic control of parallel robots based on observation of the end-effector pose by vision; (iii) In his Ph.D. thesis [Pac08], F. Paccot worked on dynamic modeling and Cartesian control of parallel robots based on observation of the end-effector pose by fast vision. But the modeling and visual feedback were neither optimal and nor fast enough; and (iv) In his Ph.D. thesis [Dah10], R. Dahmouche worked on fast sensor-space (2D visual servoing) dynamic state estimation of parallel robots based on observation of the end-effector pose with sequential acquisition, and its integration into Paccot's work [Pac08].

Moreover, P. Renaud and T. Dallej have evolved their works, in the rest of their theses, and contributed to the integrated MICMAC part, too.

1.5.3 Integrated MICMAC

In [And06], N. Andreff proposed to observe visually the legs of parallel robots for efficient modeling, control, and identification purposes. Observation of the legs unifies modeling, control, and identification modules into a single control-oriented framework. In this control-oriented framework, almost all the expressions are merged through the measurements of the leg directions. These measurements of the leg directions replace many long expressions, thus yielding simplified and more precise models regarding the sensor accuracy. It is shown that it is possible to regulate the Cartesian space too while the leg directions are controlled. In the case of a parallel robot, when the directions of the legs are followed, we find a unique solution for the end-effector pose defining the state. The most attractive side of this approach that the 3D leg directions can be directly calculated from 2D image measurements.

So why not use the leg directions (3D lines) for control of parallel robots? Andreff's main objective in [And06] was to build: « *a vision-based control-oriented framework for parallel robots based on their leg observations* » at kinematic and dynamic levels.

On the way to the main objective, the following mid-objectives are accomplished:

- Integration of the modeling, control, and identification modules at the kinematic level based on observation of the leg edges [Ren03, Dal07, RALD06].
- Proposition of a global framework for the kinematic control of the parallel robots based on observation of the leg edges [DAM11].

The following objectives remained:

- To identify the parallel robot parameters (geometric and dynamic) based on observation of the leg edges.
- To estimate the dynamic state (position and velocity) of parallel robots at high speed from the observation of the leg edges.
- To integrate modeling, control and identification modules at the dynamic level based on observation of the leg edges.
- To propose a vision-based global framework for dynamic control of parallel robots based on observation of the leg edges.

1.5.4 Complementary Background

In this section, we recall some background information which forms the fundamental basics of the MICMAC project in complement to the above state-of-the-art. Before giving these basics, in Figure 1.5.4, we outline the evolution of the MICMAC project up to now and we highlight the rest of the main objectives once more. These basics will play a crucial role in the solution of the remaining objectives.

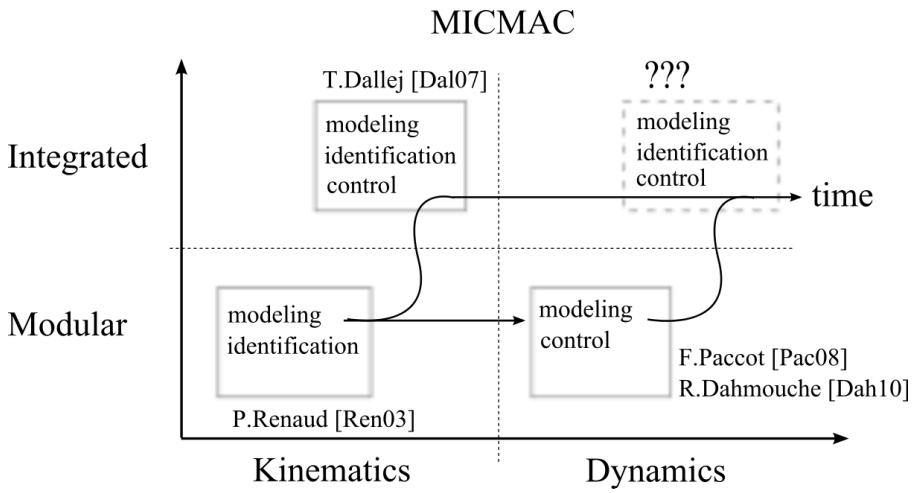


Figure 1.12 – State of the MICMAC art with the unreached objectives (upper-right box).

1.5.4.1 Vision

Let ${}^c\mathbf{P} = [{}^cX, {}^cY, {}^cZ]^T$ be a point defined in the camera frame $\mathcal{F}_c$. Then, its projection ${}^c\mathbf{p} = [{}^cx, {}^cy, 1]^T$ on the plane ${}^cZ = 1$ (see Fig. 1.13) is defined by [Fau93]:

$$ {}^c\mathbf{p} \; \alpha \; {}^c\mathbf{P} \tag{1.79} $$

where α means « proportional to ». Then, the point ${}^c\mathbf{p}$ is converted from normalized coordinates to pixel coordinates ${}^{im}\mathbf{p} = [u, v, 1]^T$ with the intrinsic camera matrix K:

$$ {}^{im}\mathbf{p} \; = \; K \; {}^c\mathbf{p} \tag{1.80} $$

Assuming that the pixels in the camera sensor are rectangular and are perfectly aligned, the intrinsic matrix K is given as below:

$$ K \; = \; \begin{bmatrix} f_u & 0 & u_o \\ 0 & f_v & v_o \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{matrix} f_u > 0 \\ f_v > 0 \end{matrix} \tag{1.81} $$

where (f_u, f_v) are the effective focal lengths (in pixel units) of the camera and where (u_o, v_o) are the coordinates (in pixel units) of the image center. Consequently, the perspective projection model is as follows:

$$ {}^{im}\mathbf{p} \; \alpha \; K \; {}^c\mathbf{P} \tag{1.82} $$

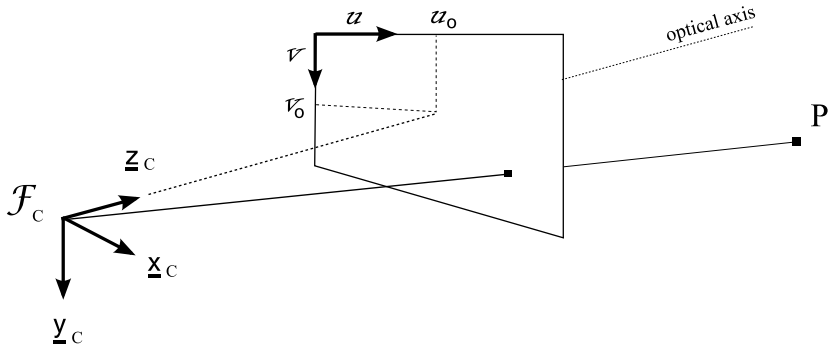


Figure 1.13 – Perspective projection of a point $\mathbf{P}$ onto the image plane.

1.5.4.2 Lines and Robotic Legs

Here in this section, we revise a 3D line representation, its perspective projection, and we discuss how lines can be adapted for the sensing and representation of the robotic legs.

1.5.4.2.1 A 3D Line and Its Projection: A 3D line L can be uniquely represented in the Euclidian space by Plücker coordinates [Plu65, SK52]:

$$(L) : (\underline{\mathbf{x}},\, \mathbf{n}) \tag{1.83}$$

where $\underline{\mathbf{x}}$ is the unit vector showing the direction of the 3D line L, and $\mathbf{n}$ is a normal vector to the plane, which contains the line L and passes through the origin of the reference frame (e.g., a camera frame). This plane is called the *interpretation plane*. The normal vector $\mathbf{n}$ can be expressed using any 3D point $\mathbf{P}$ located on the line L:

$$\mathbf{n} = \mathbf{P} \times \underline{\mathbf{x}} \tag{1.84}$$

In [AEH02], it was suggested that the line L can be represented by decomposing the normal vector $\mathbf{n}$ to the depth n of the line L from the origin of the reference frame and to the unit normal vector $\underline{\mathbf{n}}$ of the interpretation plane ($\mathbf{n} = n\,\underline{\mathbf{n}}$):

$$(L) : (\underline{\mathbf{x}},\, \underline{\mathbf{n}},\, n) \tag{1.85}$$

If we want to express a different line, $\widetilde{L}$, lying on the same interpretation plane, the previous representation can be rewritten as follows [And99]:

$$(\widetilde{L}) : (\underline{\widetilde{\mathbf{x}}},\, \underline{\mathbf{n}},\, \widetilde{n}) \tag{1.86}$$

where $\underline{\widetilde{\mathbf{x}}}$ and $\widetilde{n}$ are the direction and the depth of the line $\widetilde{L}$, respectively. Since $\widetilde{L}$ is on the same interpretation plane, the unit normal $\underline{\mathbf{n}}$ does not change.

Now, consider the intersection of the interpretation plane with the image plane. This gives a unique projection line ℓ (see Fig. 1.14) corresponding to the perspective projections of all the 3D lines (except the line which is orthogonal to the image plane) lying on the interpretation plane. Since ℓ itself is also on the interpretation plane, then it too will be expressed with the same $\underline{\mathbf{n}}$:

$$(\ell) : (\underline{\mathbf{x}}_\ell,\, \underline{\mathbf{n}},\, n_\ell) \tag{1.87}$$

where $\underline{\mathbf{x}}_\ell$ and n_ℓ are the direction and the depth of the projection line ℓ on the image plane. Consequently, in the representation of the projection line ℓ of any 3D line L, we lose the direction $\underline{\mathbf{x}}$ and the depth n but we still keep in hand the plane unit normal vector $\underline{\mathbf{n}}$. This means that it should be possible to recover the unit normal vector $\underline{\mathbf{n}}$ from the image. Now, the question is: How should one go about doing so?

On the image plane, we observe the pixel points $\{\, {}^{im}\mathbf{p}_i,\ i = 1, \ldots, m \geq 2 \,\}$ lying on the projection line ℓ. Any of these points holds the line equation:

$${}^{im}\mathbf{p}_i{}^T\ {}^{im}\underline{\mathbf{n}} = 0 \tag{1.88}$$

The solution for ${}^{im}\underline{\mathbf{n}}$ (in pixel coordinates) is in the null space of the linear system written from (1.88):

$$\begin{bmatrix} {}^{im}\mathbf{p}_1{}^T \\ \vdots \\ {}^{im}\mathbf{p}_m{}^T \end{bmatrix} \tag{1.89}$$

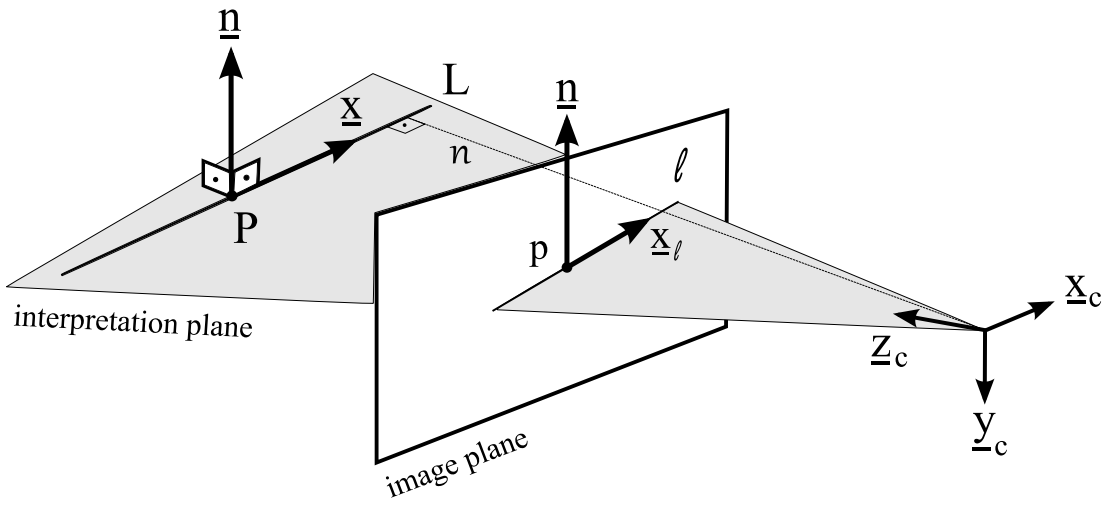


Figure 1.14 – 3D Line representation and its projection.

The above system can be solved for ${}^{im}\underline{\mathbf{n}}$ either with QR decomposition [HJ85] (algebraic, better for real-time considerations) or with SVD [HJ85] (iterative but more precise). Afterwards, substituting (1.80) into (1.88) yields:

$$ {}^{c}\mathbf{p}_i{}^{T}\ K^{T}\ {}^{im}\underline{\mathbf{n}} = 0 \tag{1.90} $$

and knowing that the line equation can be written in any space as long as the variables are also expressed in the same space:

$$ {}^{c}\mathbf{p}_i{}^{T}\ {}^{c}\underline{\mathbf{n}}\ =\ 0 \tag{1.91} $$

we can recover ${}^{c}\underline{\mathbf{n}}$ (in metric coordinates) from (1.90) and (1.91) as follows:

$$ {}^{c}\underline{\mathbf{n}}\ =\ \frac{K^{T}\ {}^{im}\underline{\mathbf{n}}}{\|K^{T}\ {}^{im}\underline{\mathbf{n}}\|} \tag{1.92} $$

and we can similarly write ${}^{im}\underline{\mathbf{n}}$ in terms of ${}^{c}\underline{\mathbf{n}}$ as below:

$$ {}^{im}\underline{\mathbf{n}}\ =\ \frac{K^{-T}\ {}^{c}\underline{\mathbf{n}}}{\|K^{-T}\ {}^{c}\underline{\mathbf{n}}\|} \tag{1.93} $$

1.5.4.2.2 Sensing of Prism-Shaped Legs: Most of the parallel robots are equipped with prism-shaped legs. Prisms are geometric solids whose bases are identical polygons lying in parallel planes and whose sides are parallelograms (see Fig. 1.15). A cylinder may be considered as a round prism, or one that has an infinite number of sides. A prism-shaped leg can be considered as a 3D line and be modeled with Plücker coordinates:

$$ (leg) : (\ \underline{\mathbf{x}},\ \underline{\mathbf{n}}_x,\ d_x,\ shape\ ) \tag{1.94} $$

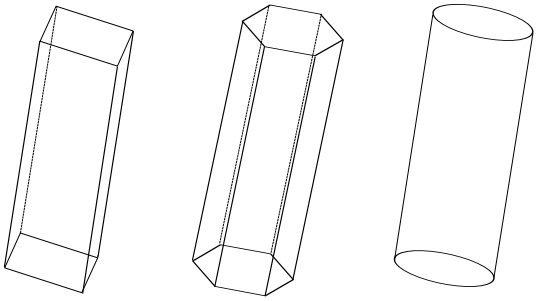


Figure 1.15 – Prisms.

where $\underline{\mathbf{x}}$ shows the direction axis, $\underline{\mathbf{n}}_x$ is the normal vector of the interpretation plane passing through the direction axis $\underline{\mathbf{x}}$ of the leg and the optical center of the camera, d_x is the shortest distance between the direction axis and the optical center, and $shape$ gives information on the geometric form of the leg (e.g., cylinder). When a leg is projected onto the image plane, we see its visual contours (left and right side edges) located at the borders between the visible and invisible parts of the leg. Figure 1.16 illustrates projection of prism-shaped legs.

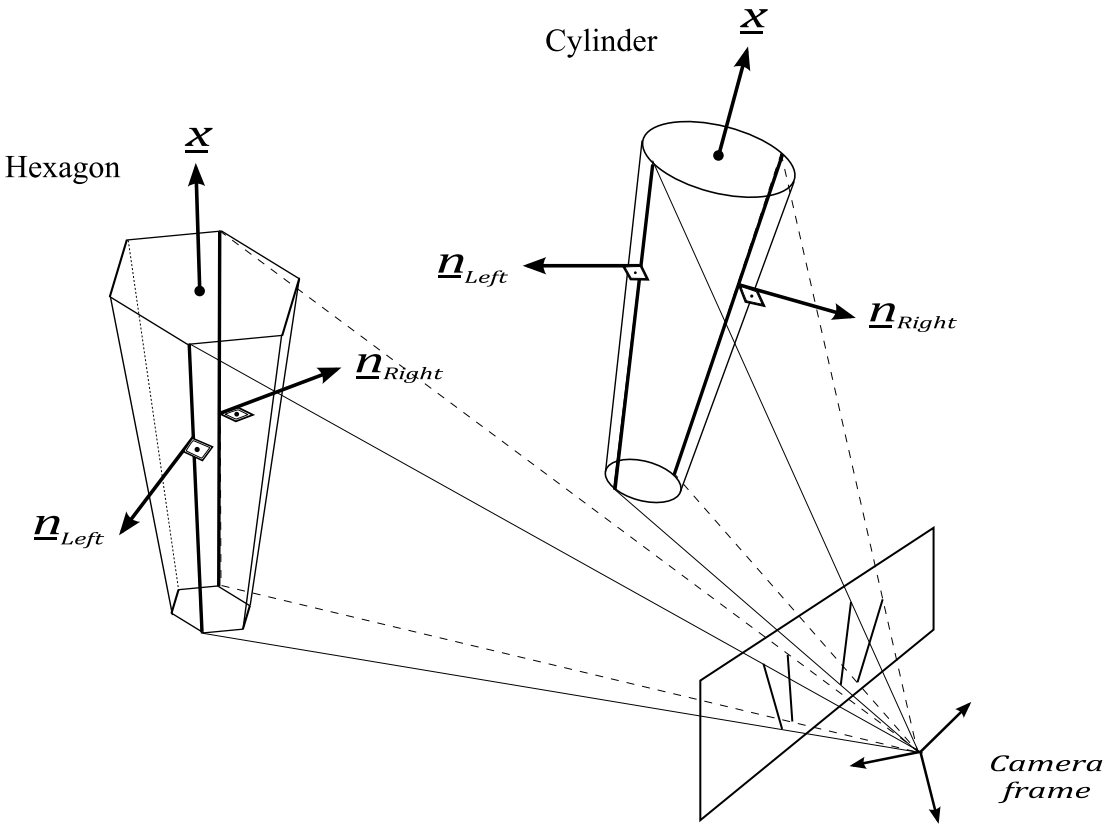


Figure 1.16 – Perspective projections of the hexagon and cylinder prisms.

These visual contours correspond to the projection lines $\{ \ell_{Left}, \ell_{Right} \}$ of the two 3D parallel lines $\{ L_{Left}, L_{Right} \}$ on the leg surface. Then, we can obtain the unit normal vectors $\{ \underline{\mathbf{n}}_{Left}, \underline{\mathbf{n}}_{Right} \}$ of the interpretation planes as explained in the previous Subsection 1.5.4.2.1. These 3D parallel lines $\{ L_{Left}, L_{Right} \}$ are also parallel to the direction axis $\underline{\mathbf{x}}$ of the leg. That is to say, the left and right side interpretation plane unit normal vectors are orthogonal to the direction axis:

$$\underline{\mathbf{n}}_{Left}^{T} \, \underline{\mathbf{x}} = 0 \,, \qquad \underline{\mathbf{n}}_{Right}^{T} \, \underline{\mathbf{x}} = 0 \tag{1.95}$$

So, it is possible to calculate the 3D direction axis $\underline{\mathbf{x}}$ of the leg from the interpretation plane unit normal vectors:

$$\underline{\mathbf{x}} = \frac{\underline{\mathbf{n}}_{Left} \times \underline{\mathbf{n}}_{Right}}{\| \, \underline{\mathbf{n}}_{Left} \times \underline{\mathbf{n}}_{Right} \, \|} \tag{1.96}$$

Consequently, we can extract the following information from the perspective projection of a prism-shaped leg with a calibrated camera (K is known) [And06]:

– *Edges*: the unit normal vectors of the interpretation planes ($\underline{\mathbf{n}}_{Left}, \underline{\mathbf{n}}_{Right}$).
– *3D direction*: unit vector showing the direction of the leg ($\underline{\mathbf{x}}$).

1.5.4.2.3 Special Case of a Cylinder: Finally, we give some more properties related to a cylindrical leg, since this type of leg is more common than other types of legs and is easier to handle. The normal vectors of the interpretation planes of a cylindrical leg can be computed as follows:

$$\underline{\mathbf{n}}_{Left} = -\cos\varphi \;\; \underline{\mathbf{n}}_{x} - \sin\varphi \;\; (\underline{\mathbf{x}} \times \underline{\mathbf{n}}_{x}) \tag{1.97}$$

$$\underline{\mathbf{n}}_{Right} = \;\; \cos\varphi \;\; \underline{\mathbf{n}}_{x} - \sin\varphi \;\; (\underline{\mathbf{x}} \times \underline{\mathbf{n}}_{x}) \tag{1.98}$$

where $\cos\varphi = d_n / d_x$, $\sin\varphi = r / d_x$, $d_n = \sqrt{d_x^2 - r^2}$, and where r is the cylindrical leg radius (see Fig 1.17). In addition, the geometry of the cylinder imposes the following constraint:

$$\mathbf{P}^{T} \, \underline{\mathbf{n}}_{Left} = -\, r \,, \quad \mathbf{P}^{T} \, \underline{\mathbf{n}}_{Right} = -\, r \tag{1.99}$$

where $\mathbf{P}$ is any point lying on the direction axis of the cylinder.

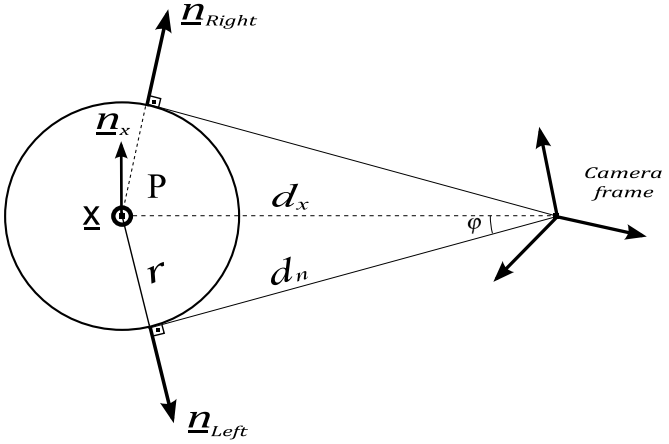


Figure 1.17 – View of the geometry of the cylindrical leg from its 3D orientation direction $\underline{\mathbf{x}}$ (perpendicular to the paper plane).

1.6 Thesis Objective

This chapter gave the state-of-the-art in control of parallel robots and the necessary background information for the rest of the thesis. We finish this chapter by stating the objective of this thesis, which is to improve the *integrated dynamic MICMAC* project further towards its remaining goals:

1. Upgrading from kinematic level to the dynamic level in control-oriented modeling of parallel robots from their leg observations.
2. Fast estimation of the dynamic state (position and velocity) of a parallel robot from its leg observations.
3. Proposing a vision-based framework for the dynamic control of parallel robots from their leg observations.
4. Identification of the dynamic parameters of a parallel robot from its leg observations.

Chapter 2

Modeling

2.1 Introduction

This chapter presents a new approach to kinematic and dynamic modeling of parallel robots which meets one of the objectives of the integrated dynamic MICMAC: « Upgrading from kinematic level to dynamic level in control-oriented modeling of parallel robots from their leg observations. »

The proposed methodology keeps the expressions simple and clear. Thus, one can easily work out all the equations from the beginning till the end with pen and paper. Here, the methodology will be carried out without paying attention to a particular parallel robot. Furthermore, for better practical understanding, the methodology will be supported with a demonstration on a simple 2 degrees of freedom planar parallel robot. Then, at the end of the chapter, the methodology will be illustrated on a very complex parallel robot: the Quattro robot. The applications of the methodology to other well-known families of parallel robots, such as the Gough-Stewart, the Delta, the 3RRR and the Orthoglide can be followed in Appendix A.

2.2 Motivation and Objective

Modeling of robots can be categorized in two groups: *application-oriented* methods and *analysis-oriented* methods.

In application-oriented modeling, the generic scenario is as follows: since the motor encoders that measure the articular positions are directly implanted in robots and are rich enough to supply information about the full geometric configuration of serial robots, they are adopted immediately as a basic medium for sensing.

This misleads one into using them also for parallel robots. However, when parallel robots are considered, this information becomes poor because of many other sensorless passive joints. If one expresses models with only active joint coordinates, then the models inflate, become slow, hard to understand and to implement. This inevitably urges one to offer simplifications [VPP03] and to omit some of the modeling errors in the mechanism, thus giving simplified and fast [NKC$^+$08] but approximate new models for control. Thus, application-oriented methods, based on joint sensing, become inefficient when the complexity of the robot increases.

On the other hand, *analysis-oriented* methods mostly concentrate on finding efficient, intuitive, simple and linear procedures for synthesis and analysis of complex robots. Some of these analysis-oriented methods use Screw theory [WS06], [Fea00], [MLS94], [BS98], [RGM08], [Tsa98], [MD85], [Tsa99] and Grassmann-Cayley [ST02] algebra which are based on lines of motions. These lines of motions are the *joint axes*, and these works model the joint motions. However, their practical applicability on real complex robots is limited. Instead, they are used for analysis.

Obviously, in these scenarios, the difficulties in modeling and applicability are sourced from the lack of an appropriate sensing. What if we had extra sensor(s)? What if we knew everything about the mechanism? Certainly, we would come up with a better concrete method.

Our objective is to provide a simple, accurate and applicable modeling methodology which would become the favorite option for researchers and engineers.

2.3 Discussions on the Inspiring Works

Most of the proposed approaches for deriving the kinematics and dynamics of a parallel robot suffer from the lack of efficiency - as defined by Kane *et al*. -, namely « *relative simplicity, ease of manipulation for purposes of designing automatic control systems and minimal consumption of time during numerical solution* » [MK96]. Hence, we shall investigate Kane, Khalil, and Tsai's methodologies which have already made important steps forward in improving efficiency and inspired us to put forward our new control-oriented methodology.

2.3.1 Kane's Method

With regards to efficiency, Kane has revealed the notion of *generalized speeds* to increase the efficiency of expressions [KL85]. The *generalized speeds* are functions of generalized coordinates (scalar joint values) and their speeds. Their choice is completely arbitrary, and is usually determined by inspection of the velocities of bodies in a mechanism. A good choice of *generalized speeds* can have important effect on the resulting equations. Once their choice is made, then the methodology proposed by Kane is simply a matter of:

- deriving the mechanism's kinematics in terms of the generalized coordinates and generalized speeds;
- computation of all the forces that exist in the mechanism;
- projecting these forces on the directions of motion (i.e., directions associated to the generalized speeds) in order to obtain the generalized forces;
- formal calculus (i.e., automatic generation of the model equations).

Kane's method might be considered as a way of overcoming the difficulties caused by the inappropriate sensing of the mechanism. However, it seems that this method is still relevant for modeling purposes and not for control ones when the parallel robots are considered. Expressed from the control point of view, this is a matter of defining the most appropriate dynamic state variables for a parallel robot. It now seems certain that using the actuator positions (independent set of generalized coordinates) as the static state variables is not necessarily the optimal choice, because there is usually not a single solution to the forward kinematic problem. Consequently,

using the actuator positions and their velocities as the dynamic state variables is certainly not the appropriate choice either.

Since the inverse kinematic problem is usually well posed for a parallel robot, a more efficient choice is to use the end-effector Cartesian pose and its velocity as dynamic state variables since this is usually the operational space. This choice leads to efficient models *as long as one is able to measure or estimate the pose and velocity of the end-effector in the Cartesian space*. And here is another loss of efficiency: one can not directly measure in the Cartesian space, so one has to estimate such variables, either through a mechanism or by optical means (e.g., laser, vision). The estimation is always a non-linear problem, except for high cost laser trackers.

2.3.2 Khalil's Method

In addition to Kane, Khalil [KI04] also proposes a methodology which is specific to parallel robots. His approach has the advantage of intuitively handling the kinematic constraints. To do so, Khalil expresses the dynamics of a parallel robot from the equilibrium of all forces applied on the moving-platform. He takes the following steps in his methodology:

- to consider each kinematic leg of a parallel robot as an independent serial robot;
- to write the inverse dynamics of each kinematic leg using all the passive and active joint coordinates (redundant set of variables);
- to transfer all the efforts of kinematic legs to the moving-platform using their inverse velocity kinematic models;
- finally, to sum all the forces collected on the moving-platform and then to project the final total effort onto the active joints.

However, the strong drawback of this method is its loss of efficiency, because it requires sensing and actuation to be collocated. Moreover, the computation of the dynamics of each kinematic leg loses its intuitiveness, and it needs computation of the inverse of the forward velocity kinematic model of a serial kinematic leg because of the balance of all the efforts on the moving-platform. Yet, as shown in [PAM09], the method becomes extremely efficient when used together with the end-effector sensing, because it turns out entirely linear. Nonetheless, the method in [PAM09] is probably not the most efficient one, since, as stated above, the sensing part of it is "sub-efficient".

2.3.3 Tsai's Method

In [Tsa99] Tsai formulated the dynamics of parallel robots based on virtual work principle. This formulation follows the steps below:

- compute the kinematic twist and wrench twosome at the mass center of every link;
- compute the link Jacobians relating the link kinematic twist to the end-effector kinematic twist.
- finally, express all the virtual works of the links, of the actuators and of the platform, in the end-effector frame through the computed Jacobians.

Expressing every effort in the end-effector frame with a similar way to Khalil, Tsai easily takes into account the motion constraints of the closed-loop kinematics. Using virtual work also simplifies writing of the final equations of motion. For application purposes, by MATLAB simula-

tions, Tsai demonstrated his formulation for the Gough-Stewart platform where the equations were written based on joint values. However, as stated earlier, for real applications of parallel robots joint sensing is not enough. Since Tsai's work was rather for analysis purposes, he did not put any discussion for the applicability of his method even it looks very intuitive.

The aim of this chapter is thus to investigate further efficient modeling.

2.4 Methodology

In order to reduce the ambiguity in the terminology and in the context, we redefine first the descriptive and then a mathematical language for a robot. Afterwards, we proceed into the details of the formulation of the proposed methodology.

2.4.1 The Descriptive Language of a Robot

Joint [articulation]: connects two or more units (e.g., a revolute joint).

Link [body, element]: a unit in a connected series of units (e.g., a bar between the joints).

Limb [kinematic chain, linkage, leg, arm]: a chain of units forming a kinematic chain.

End-effector [active tool tip]: interacts with the environment. In parallel robots, it is sometimes incorrectly called a moving-platform, (or a moving-plate). Actually the end-effector should be considered only as a sub-part of the moving-platform.

2.4.2 A Mathematical Language for a Robot

Every robot is a multi-body system and its *motion space* (i.e., static and dynamic state) can be described through a set of *variables* and *constant parameters*. These variables consist of the coordinates of joints, points, vectors fixed within the bodies (e.g., base, legs, moving-platform), matrices defining the motion constraints of the links, and forces (e.g., actuator forces, inertial forces, forces of gravity, frictions) acting on the links, while the constant parameters are the lengths, the masses and the inertias.

2.4.2.1 Variables

Scalars $\in \Re^{1\times 1}$: They are represented with small letters and symbols. For example, a distance between two points can be noted as d, and a rotation angle around an axis with θ.

Joint coordinates $\in \Re^{1\times 1}$: An active *joint coordinate*, q, may represent the angular (*radian*) or the distance (*meter*) measure depending on the type of the joint (revolute or prismatic).

Points & Position Vectors $\in \Re^{3\times 1}$: They are represented with boldface capital letters, such as: $\mathbf{O}$, $\mathbf{P}$, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{E}$ and so on. The $\mathbf{O}$ and $\mathbf{E}$ denote the respective origins of the base and the end-effector frames.

Vectors $\in \Re^{r\times 1}$: They are denoted with boldface small letters. For instance, a translational displacement $\mathbf{t}$. In addition unit vectors are underlined, such as the directions of a frame axes $\underline{\mathbf{x}}$, $\underline{\mathbf{y}}$, $\underline{\mathbf{z}}$.

Matrices $\in \Re^{r\times c}$: They are represented with capital letters, such as the M, N, V and so on.

Frames : A *frame* is a set of 3 orthogonal unit axes ($\underline{\mathbf{x}}, \underline{\mathbf{y}}, \underline{\mathbf{z}}$) fastened to a reference point which uniquely determines (with real numbers) the position and the orientation of a body. It is noted as $\mathcal{F}_{point} = \{ point, \underline{\mathbf{x}}_{point}, \underline{\mathbf{y}}_{point}, \underline{\mathbf{z}}_{point} \}$. For example, the base and end-effector frames, which exist in every robot, will be noted as $\mathcal{F}_o = \{ \mathbf{O}, \underline{\mathbf{x}}_o, \underline{\mathbf{y}}_o, \underline{\mathbf{z}}_o \}$ and $\mathcal{F}_e = \{ \mathbf{E}, \underline{\mathbf{x}}_e, \underline{\mathbf{y}}_e, \underline{\mathbf{z}}_e \}$, respectively. The $\underline{\mathbf{x}}$ axis of a body frame is always oriented in lengthwise direction of the body.

Poses $\in SE(3)$: A *pose* of a frame can be represented either as a $r \times c$ matrix or a $n \times 1$ column array. The representation of the pose of a frame will be noted with $\mathbb{X}_{point}$ regardless of whether it is a matrix or a column array. The difference between a matrix and a column array representations will be made known by defining $\mathbb{X}_{point}$ as the element of $\Re^{r \times c}$ or $\Re^{n \times 1}$, respectively. For example, in the context, the representation of the pose of a frame located at point $\mathbf{A}$ will be defined as $\mathbb{X}_a$. Only the representation of the pose of the end-effector frame, which is located at point $\mathbf{E}$, will be noted as $\mathbb{X}$ without a sub-script for the simplicity of the notation.

Mass centers $\in \Re^{3 \times 1}$: They are denoted with $\mathbf{S}_{point}$. For instance, the mass center of a link attached to an articulation point $\mathbf{P}$ will be noted as $\mathbf{S}_p$.

Velocities : They are noted with a dot ($\dot{}$) over the variables. For example, velocity of a mass center is $\dot{\mathbf{S}}$, velocity of a joint coordinate is $\dot{q}$, velocity of a unit vector is $\dot{\underline{\mathbf{x}}}$.

Accelerations : They are shown with double dot ($\ddot{}$) over the variables. For example, acceleration of a mass center is $\ddot{\mathbf{S}}$, acceleration of a joint coordinate is $\ddot{q}$, acceleration of a unit vector is $\ddot{\underline{\mathbf{x}}}$.

Active & Reactive Forces : A force will be noted with small f letter and a force vector with small boldface $\mathbf{f}$ letter. Active forces are efforts of the linear motors and forces of gravity of the bodies. In order to express a specific active force (or a force vector), a subscript will be added to f (or $\mathbf{f}$), e.g., a gravity force vector will be $\mathbf{f_g}$. On the other hand, reactive forces are inertial and frictional forces. An inertial force (force vector) will be expressed with f^* ($\mathbf{f}^*$). A frictional force (force vector) will be noted with $\bar{f}$ ($\bar{\mathbf{f}}$).

Active & Reactive Torques : A torque will be noted with τ letter and a torque vector with boldface $\boldsymbol{\tau}$ letter. Active torques are the efforts of the rotary motors. On the other hand, reactive torques are inertias and frictions. An inertial torque (torque vector) will be expressed with τ^* ($\boldsymbol{\tau}^*$). A frictional torque (torque vector) will be noted with $\bar{\tau}$ ($\bar{\boldsymbol{\tau}}$).

Note: All the variables are expressed in a single fixed reference frame (e.g., a camera frame, or the robot base frame). The relative velocities of these variables are also expressed with respect to this same fixed reference frame.

2.4.2.2 Constant Parameters

Length : ℓ shows the length of a body.

Mass : m expresses the mass of a body.

Gravity constant : $\mathbf{g}$ is the constant gravity acceleration vector of the Earth. [1]

1. If the robot is mobile, then the direction of the gravity vector with respect to robot's base frame may change. If the robot is on another planet (e.g., moon), then the magnitude of the gravity vector changes too.

Inertia [2]: I_x, I_y, I_z are the principal moments of inertias of a body.

2.4.2.3 Motion Space

State Variables [Kane's generalized coordinates]: a chosen set of variables used to define the positions, velocities, accelerations and forces of all the links in the mechanism. These variables can be either an independent set or a redundant set.

Motion Basis [Kane's generalized speeds]: a chosen set of variables (i.e., scalars, vectors, functions) whose linear combination expresses the motion of the mechanism. Normally, a basis in linear algebra is a set of linearly independent vectors in the same space, but here we let it also be a dependent set of non-homogenous variables for simplicity of the equations. Hence, in our context, a motion basis is either a set of independent (minimal) variables or a set of dependent (redundant) variables.

Motion Constraints [Change of Motion Basis]: map the motion of the mechanism expressed in a redundant motion basis into the minimal motion basis made of actuator axes.

Kinematic Coordinates [Kane's partial velocities]: express the velocity of the mechanism in a given motion basis. The kinematic coordinates of the velocity of a mechanism are the partial derivatives of its kinematic equations with respect to the motion basis components.

Dynamic Coordinates [Kane's generalized forces]: define the dynamic equilibrium of the mechanism in a given motion basis. They are computed from the active and reactive forces of all the links in the mechanism.

2.4.3 Proposed Modeling Formulation

The methodology we propose here supposes that there is no restriction on the variable set selection needed for modeling, and suggests that the selected state variables should fulfil as much as possible the following criteria:

- *Algebraicity*: ability to be solved simply with formal tools, such as linear algebra;
- *Completeness*: ability to represent fully both kinematics and dynamics;
- *Sensibility*: ability to be perceived directly by physical sensors;
- *Readability*: ability to allow for a good and easy understanding of the model;
- *Codability*: ability to be implemented easily on a computer;

so that the *simplicity*, the *geometric intuitiveness* and the *numerical efficiency* requested by Kane can be kept on the final expressions. The procedure which longs for these expressions passes through the following steps:

Decomposition : We decompose the parallel robot into its simplest possible rectilinear parts: the kinematic elements of the kinematic legs and the moving-platform.

Choosing State Variables : We represent the states of these parts with a *redundant set of variables*: orientation unit vectors of the kinematic elements, the actuator coordinates, etc., so that expressions are compact and linear.

2. Sometimes the mass center can move with respect to its body frame, then the inertia is not constant anymore.

Choosing a Motion Basis : We select a *redundant motion basis* and derive the motion constraints between the passive (i.e., unactuated) and active (i.e., actuated) variables of this basis.

Computing the Coordinates : We compute the kinematic and dynamic coordinates of the mechanism from each of the decomposed kinematic elements.

Writing Equations of Motion : We combine all the dynamic coordinates with the motion constraints and then write the final *equations of motion* of the parallel robot.

Note that we do not put any constraint on the *minimality* (which is inherited from serial robots), contrarily we allow for *redundancy*.

In the light of d'Alembert's principle of virtual work, the equations of motion (i.e., dynamic equilibrium) take the following form:

$$\begin{pmatrix} Dynamic \\ Coordinates \end{pmatrix}^T \begin{pmatrix} Redundant \\ Motion\,Basis \end{pmatrix} = 0 \tag{2.1}$$

which implies that the sum of all the exerted efforts on the defined redundant motion basis should vanish. This can be rewritten in terms of the minimal motion basis (i.e., actuation space) through the motion constraints (i.e., change of basis) as below:

$$\begin{pmatrix} Dynamic \\ Coordinates \end{pmatrix}^T \left(\begin{pmatrix} Motion\,Constraint \\ Transformations \end{pmatrix} \begin{pmatrix} Minimal \\ Motion\,Basis \end{pmatrix} \right) = 0 \tag{2.2}$$

since the above system is defined at minimal motion basis, then it deduces to the following final form:

$$\begin{pmatrix} Motion\,Constraint \\ Transformations \end{pmatrix}^T \begin{pmatrix} Dynamic \\ Coordinates \end{pmatrix} = \mathbf{0} \tag{2.3}$$

where the dynamic coordinates (i.e., generalized forces) are written as follows:

$$\begin{pmatrix} Dynamic \\ Coordinates \end{pmatrix} = \begin{pmatrix} Kinematic \\ Coordinates \end{pmatrix}^T \left(\begin{pmatrix} Active \\ Forces \end{pmatrix} + \begin{pmatrix} Reactive \\ Forces \end{pmatrix} \right) \tag{2.4}$$

and where the kinematic coordinates express velocities of kinematic elements, namely the velocity of the whole mechanism:

$$\begin{pmatrix} Mechanism \\ Velocity \end{pmatrix} = \begin{pmatrix} Kinematic \\ Coordinates \end{pmatrix}^T \begin{pmatrix} Redundant \\ Motion\,Basis \end{pmatrix} \tag{2.5}$$

Theorem 1 *[**Linear Implicit Dynamic Model**]: Let geometric parameters, dynamic parameters, positions, velocities, and accelerations of a mechanism be known, then a linear implicit dynamic model (LImplDM) for this mechanism can be written as follows:*

$$A\,\mathbf{\Gamma} + \mathbf{b} = \mathbf{0} \tag{2.6}$$

where matrix A is dependant to mechanism configuration and it relates the unknown force vector $\mathbf{\Gamma}$ of the actuators to the contributing efforts $\mathbf{b}$ of the kinematic elements of the mechanism.

In the following subsections, we go into details of the steps (decomposing a robot to its simplest parts, choosing state variables and a motion basis, deriving the motion constraints, etc.) of the proposed modeling formulation and we prove the Theorem 1.

A little note on “cross-product": We will use the cross-product frequently in our equations for two purposes:

- (i) to produce a third perpendicular vector to a plane defined by given two independent vectors (e.g., so as to form an orthogonal basis);
- (ii) to compute rotational velocity and acceleration of a body (e.g., motion of a link attached to a revolute joint);

Symbolically cross-product is implied by “$\times$" operator which takes left-hand and right-hand operands to its sides. Let $\mathbf{a} = [a_x,\, a_y,\, a_z]^T$, $\mathbf{b} = [b_x,\, b_y,\, b_z]^T$ and $\mathbf{c} = [c_x,\, c_y,\, c_z]^T$ be vectors of 3 dimensional (3D) vector space $\Re^3$, and where the cross-product of $\mathbf{a}$ and $\mathbf{b}$ is $\mathbf{c}$. Thus, one can write:

$$\mathbf{a} \times \mathbf{b} = \mathbf{c} \tag{2.7}$$

Equation (2.7) is just a mathematical notation of the cross-product which corresponds to:

$$\begin{aligned} c_x &= a_y b_z - a_z b_y \\ c_y &= a_z b_x - a_x b_z \\ c_z &= a_x b_y - a_y b_x \end{aligned} \tag{2.8}$$

The weird calculations in above equations may mislead one to suppose that the cross-product is a non-linear operator. However, in (2.7), the cross-product is a linear map disguising itself as a vector. One can rewrite (2.7) as follows:

$$[\mathbf{a}]_\times \mathbf{b} = \mathbf{c} \tag{2.9}$$

where $\{\, [\mathbf{a}]_\times : \Re^3 \longrightarrow \Re^3 \,|\, [\mathbf{a}]_\times \in SO(3) \,\}$ is a linear function of infinitesimal rotation group which maps vector $\mathbf{b}$ to vector $\mathbf{c}$ in 3D Euclidean space:

$$[\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \in \Re^{3\times 3} \tag{2.10}$$

In our expressions, we will use the first written style shown in (2.7) for the simplicity of notation, and the second written style shown in (2.9) for the algebraic manipulation consistency of equations.

2.5 New Construction « Kinematic Element » Definition

A robot can be broken down into its basic primitives as below:

$$\begin{aligned} (robot) &: (\,\{\, base\,platform \,\} \cup \{\, limb(s) \,\} \cup \{\, end-effector \,\}\,) \\ (base\ platform) &: (\,\{\, link(s) \,\}\,) \\ (limb) &: (\,\{\, joints \,\} \cup \{\, links \,\}\,) \\ (end-effector) &: (\,\{\, link \,\}\,) \end{aligned} \tag{2.11}$$

where each of $\{\, joints \,\}$ can be either active or passive. *Roughly speaking, a robot is a set of static and moving bodies*. Our objective is here to homogenize the theoretical construction

of a robotic mechanism by defining a single basic structure. We will call this basic structure as a *kinematic element*. Here, a *kinematic element* is an elementary rectilinear sub kinematic chain which contains both joint(s) and link(s), and consequently which has mobilities. In the light of the rough definition of a robot, these *kinematic elements* will replace the static and moving bodies. The definition of a *kinematic element* is built upon rigid-body assumption of its primitive components (e.g., links, joints). That is to say, the external forces exerted on a component of the *kinematic element* do not cause any deformation, or any given two points in the component preserve their distance. A rigid-body robotic mechanism can be thus simply redefined as follows:

$$(robot) : \{ (kinematic\, element)_1 , \ldots , (kinematic\, element)_n \} \tag{2.12}$$

Tej Dallej in his Ph.D. thesis [Dal07] has roughly employed the “kinematic element" terminology for the last link of a kinematic chain of a parallel robot in order to propose a framework for the modeling and control of parallel robots at kinematic level. However, in [Dal07] a “kinematic element" was never fully formalized. On the other hand, here, we will give a mathematical model of a *kinematic element* which is intuitive and empiric. Furthermore, we are interested not only with the kinematics but also with the dynamics of this *kinematic element*. In the next subsections, we briefly recall some basic primitives and go into details of the definition of such a *kinematic element*.

Note: In IFToMM dictionary (standardization of terminology of machine and mechanism theory), the term “element” has already been defined as a solid body or a fluid component of a mechanism [Ion03]. Here, we augmented this term with the adjective “kinematic” to express our *new basic structure* which has various mobilities. For the time being, we could not find a better name.

2.5.1 Construction Primitives

2.5.1.1 Joints

The mobility of the links of a robot are defined by these joint types:
- *Revolute*: Rotates around an axis (1 dof). It is noted by (**R**).
- *Universal*: Rotates around two axes (2 dof). It is noted by (**U**).
- *Spherical*: Rotates around three axes (3 dof). It is noted by (**S**).
- *Prismatic*: Slides on a direction (1 dof). It is noted by (**P**).
- *Parallelogram*: Translates in three axes (3 dof). It is noted by (**Pa**).

We note that a parallelogram actually is a mechanical structure composed of links and joints. The reason why we included it here is because another link can be firmly fastened to such a parallelogram, and here we consider the parallelogram link as a pseudo-joint.

2.5.1.2 Links

The base platform, the limbs (so-called *arms* or *legs*) and the end-effector of a robot are formed by a set of links (*rigid bodies*). Links are connected to each other with previously mentioned joint types and move interactively with respect to the mobility given by these joints.

2.5.2 Geometric Representation of a « Kinematic Element »

Here, we propose to model the geometric state of a *kinematic element* according to its mobilities which are generated by the interactions with other *kinematic elements* and by the joints exist in itself. So, we suppose that a *kinematic element* can have *extrinsic* and *intrinsic* mobilities.

The extrinsic mobility can exist due to the connection joint(s) at the input articulation point of a *kinematic element*, and it can be noted as follows:

$$(extrinsic\, mobility\ joints) :\in \{\, \varnothing,\, (\mathbf{R}),\, (\mathbf{U}),\, (\mathbf{Pa}) \,\} \tag{2.13}$$

We define the state of the extrinsic mobility as below:

$$(extrinsic\, mobility\ state) : \{\, \mathbf{A},\, \underline{\mathbf{x}} \,\} \tag{2.14}$$

where $\mathbf{A}$ is the input articulation point and $\underline{\mathbf{x}}$ is the $3D$ unit direction vector of the *kinematic element* representing the pose sourced from the angular position(s) of the revolute, universal, spherical or parallelogram type joints.

The intrinsic mobility can exist due to the implanted joint(s) inside of a *kinematic element*, and it can be noted as follows:

$$(intrinsic\, mobility\ joints) :\in \{\, \varnothing,\, (\mathbf{P}),\, (\mathbf{R}) \,\} \tag{2.15}$$

We define the state of the intrinsic mobility as below:

$$(intrinsic\, mobility\ state) : \{\, \varnothing,\, d,\, \theta \,\} \tag{2.16}$$

where $\varnothing$, d and θ denote nothing (when there is no intrinsic mobility), the implanted prismatic joint coordinate (elongation along $\underline{\mathbf{x}}$) and the implanted revolute joint coordinate (twisting about $\underline{\mathbf{x}}$), respectively. An implanted prismatic joint elongates or contracts the *kinematic element* along its direction $\underline{\mathbf{x}}$. Similarly, an implanted revolute joint rotates the *kinematic element* around itself, namely around $\underline{\mathbf{x}}$ (see Fig. 2.1). Thus, a generic geometric representation of a *kinematic element* can be described as follows:

$$(geometric\ state) : \{\, \mathbf{A},\, \underline{\mathbf{x}},\, d,\, \theta \,\} \tag{2.17}$$

Analogy 1 *Chasles' theorem [Cha30] states that any rigid body displacement can be reduced to a canonical form, where the displacement is achieved by a rotation (θ) around a geometric line L and a translation (d) along the same line L.*

This implies that a kinematic element, which uses the same geometric variables for its state (see Fig. 2.1), can be considered as a physical (visually concrete) representation of the canonic displacement between its connection points with the previous and next kinematic elements in the chain. Let $\mathbf{A}$ and $\mathbf{B}$ be the connection points at the tips of a kinematic element (not necessarily on the direction axis), then the displacement from $\mathbf{A}$ to $\mathbf{B}$ can be represented as follows:

$$\mathbf{B} \,=\, \mathbf{A} \,+\, d\,\underline{\mathbf{x}} \,+\, \mathbf{r}(\theta) \tag{2.18}$$

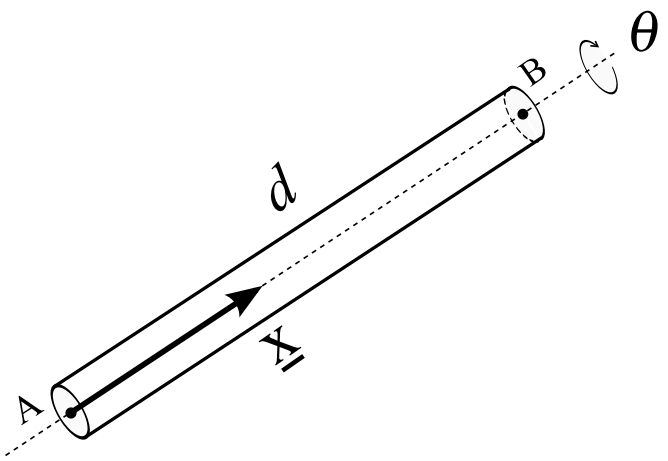


Figure 2.1 – A kinematic element and its geometric state variables: the input joint center $\mathbf{A}$, the unit 3D direction vector $\underline{\mathbf{x}}$, the length d and the rotation angle θ around its direction vector. $\mathbf{B}$ is the output joint center.

where $\mathbf{r}$ *is a vector which is a function of the rotation parameter* θ*. Usually, these connection points are designed to lie on the direction axis (or an axis parallel to the direction axis) of an element so that* $\|\mathbf{r}\| \approx 0$*, thus (2.18) appears in this way:*

$$\mathbf{B} = \mathbf{A} + d\,\underline{\mathbf{x}} \tag{2.19}$$

this helps keep the expressions simple. Consequently, θ *does not have any effect on the position of the output articulation point (*$\mathbf{B}$*) of the element. If* θ *is not an* ***internal mobility***[3] *of the mechanism, then it does influence the direction axis of the next kinematic element in the chain.*

Distinguishing between intrinsic and extrinsic mobilities allows us to define some basic kinematic element types such as:

Bar Element $[B]$ A kinematic element which has only extrinsic mobility. It moves under the influence of one or more joints, which can be revolute joint, or universal joint, or a combination of revolute and universal joints, or parallelogram pseudo-joint. Hence, we write its geometric state as follows:

$$(bar\ state) : \underbrace{\{\,\mathbf{A},\,\underline{\mathbf{x}}\,\}}_{varying} \cup \underbrace{\{\,d,\,\theta\,\}}_{constant} \tag{2.20}$$

where $\mathbf{A}$ is the varying input point and $\underline{\mathbf{x}}$ is the varying $3D$ direction unit vector of the kinematic element. They vary due to the interactions with other kinematic elements and extrinsic joints of the kinematic element. d and θ are the constant length and angular rotation coordinate of the kinematic element, respectively. In some cases, there might

3. Internal mechanism mobility: a redundant motion which does not change the geometric configuration of the mechanism.

be a rotation around the direction vector $\underline{\mathbf{x}}$, then its geometric state can be rigorously rewritten as $(\,\mathbf{A},\,\underline{\mathbf{x}},\,\theta\,)$ by replacing θ from constant part to the varying part of the state representation in (2.20). However, we note that in most of the parallel robots this rotation θ is passive (i.e., internal mechanism mobility) and does not effect the orientation of the next kinematic element in the chain. Therefore, the geometric state of a *bar type* kinematic element will be considered only as its input point $\mathbf{A}$ and its unit direction vector $\underline{\mathbf{x}}$ without a rotation. For instance, a parallelogram link, whose rotation around its lengthwise direction is restricted, perfectly fits into this bar type kinematic element representation. The *bar kinematic element* is the most common type and exists almost in every parallel robot.

Spindle Element $[Sp]$ A kinematic element which has an extrinsic mobility with an active rotation around its direction vector, or an extrinsic mobility plus an intrinsic mobility with an implanted revolute joint. Thus, we write its geometric state as follows:

$$(spindle\ state) : \underbrace{\{\,\mathbf{A},\,\underline{\mathbf{x}},\,\theta\,\}}_{varying} \;\cup\; \underbrace{\{\,d\,\}}_{constant} \tag{2.21}$$

where θ is the varying angular coordinate of the rotation due to active extrinsic joints and implanted intrinsic revolute joint. One can find some spindle type kinematic elements in Zlatanov's 3-URU DYMO parallel mechanism [ZBG02].

Telescopic Element $[T]$ A kinematic element which has extrinsic mobility and as well as intrinsic mobility due to an implanted prismatic joint. For instance, this *telescopic type* kinematic element exists in a Gough-Stewart parallel robot. Its geometric state is as follows:

$$(telescopic\ state) : \underbrace{\{\,\mathbf{A},\,\underline{\mathbf{x}},\,d\,\}}_{varying} \;\cup\; \underbrace{\{\,\theta\,\}}_{constant} \tag{2.22}$$

where d is the varying metric coordinate of the implanted prismatic joint, and it also corresponds to the length of the kinematic element.

Screw Element $[Sc]$ A kinematic element which has an extrinsic mobility and as well as an intrinsic mobility due to the implanted active prismatic and revolute joints. The fourth leg of the Delta parallel robot (i.e., the one that gives a rotation to the end-effector) can be considered as an example of this *screw type* kinematic element. Its geometric state is written as follows:

$$(screw\ state) : \underbrace{\{\,\mathbf{A},\,\underline{\mathbf{x}},\,d,\,\theta\,\}}_{varying} \tag{2.23}$$

One can imagine other types and can easily define their states with the concept given in these examples. *This is a non-minimal representation, but nonetheless it allows one to write the equations in a compact and linear fashion.*

Remark: Given the base connection point of a kinematic chain, one can express any point (e.g., an input articulation point $\mathbf{A}$ of a kinematic element) along this kinematic chain in terms of only the $\{\,\underline{\mathbf{x}},\,d,\,\theta\,\}$ variables of the kinematic elements belong to the this kinematic chain.

2.5.3 Dynamic Representation of a « Kinematic Element »

Dynamic state of a kinematic element can be defined with:

$$(dynamic\ state) : \{\, \mathbf{A}, \underline{\mathbf{x}},\, d,\, \theta \,\} \;\cup\; \{\, m,\, \mathcal{I},\, \mathbf{S},\, \dot{\mathbf{S}},\, \ddot{\mathbf{S}},\, \dot{\mathbf{A}},\, \dot{\underline{\mathbf{x}}},\, \dot{d},\, \dot{\theta},\, \ddot{\mathbf{A}},\, \ddot{\underline{\mathbf{x}}},\, \ddot{d},\, \ddot{\theta} \,\} \cup \{\, \mathbf{f_g},\, \mathbf{f}^*,\, \boldsymbol{\tau}^*,\, \boldsymbol{\tau}_{\underline{\mathbf{x}}},\, \boldsymbol{\tau}^*_{\underline{\mathbf{x}}},\, \bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}},\, \mathbf{f}_d,\, \mathbf{f}^*_d,\, \bar{\mathbf{f}}_d,\, \boldsymbol{\tau}_\theta,\, \boldsymbol{\tau}^*_\theta,\, \bar{\boldsymbol{\tau}}_\theta \,\} \tag{2.24}$$

which contains:

- (i) intrinsic dynamic parameters and higher-order kinematics:
  - its mass m and its central inertia dyadic $\mathcal{I}$;
  - its mass center position, velocity and acceleration: $\mathbf{S}$, $\dot{\mathbf{S}}$ and $\ddot{\mathbf{S}}$;
  - the velocities of its geometric state variables: $\dot{\mathbf{A}}$, $\dot{\underline{\mathbf{x}}}$, $\dot{d}$, $\dot{\theta}$;
  - the accelerations of its geometric state variables: $\ddot{\mathbf{A}}$, $\ddot{\underline{\mathbf{x}}}$, $\ddot{d}$, $\ddot{\theta}$;
- (ii) forces and torques:
  - its gravity force $\mathbf{f_g}$;
  - its body inertial force $\mathbf{f}^*$ and inertial torque $\boldsymbol{\tau}^*$;
  - an active extrinsic torque $\boldsymbol{\tau}_{\underline{\mathbf{x}}}$ of an extrinsic rotary actuator that turns the kinematic element around $\underline{\mathbf{z}}$ axis of its body frame, where
  $$\underline{\mathbf{y}} = \dot{\underline{\mathbf{x}}} \,/\, \|\dot{\underline{\mathbf{x}}}\|\,, \qquad (\dot{\underline{\mathbf{x}}} \perp \underline{\mathbf{x}})\,, \qquad \underline{\mathbf{z}} = \underline{\mathbf{x}} \times \underline{\mathbf{y}} \tag{2.25}$$
  and as well as the inertial torque $\boldsymbol{\tau}^*_{\underline{\mathbf{x}}}$ and the frictional torque $\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}}$ of this rotary actuator;
  - an active intrinsic force $\mathbf{f}_d$ which elongates or shortens the kinematic element along its direction $\underline{\mathbf{x}}$, and as well as the inertial force $\mathbf{f}^*_d$ and frictional force $\bar{\mathbf{f}}_d$ of this active prismatic joint;
  - an active intrinsic torque $\boldsymbol{\tau}_\theta$ of an intrinsic rotary actuator that turns the kinematic element around its direction $\underline{\mathbf{x}}$, and as well as the inertial torque $\boldsymbol{\tau}^*_\theta$ and frictional torque $\bar{\boldsymbol{\tau}}_\theta$ of this intrinsic rotary actuator.

2.6 Kinematics of a « Kinematic Element »

2.6.1 Positions

The output articulation point (the end point) $\mathbf{B}$ of the kinematic element can be computed by using its state parameters and its articulation input point $\mathbf{A}$ (the initial point) as follows:

$$\mathbf{B} = \mathbf{A} + d\,\underline{\mathbf{x}} \tag{2.26}$$

and the mass center $\mathbf{S}$ of a kinematic element can be written as follows:

$$\mathbf{S} = \mathbf{A} + x\,\underline{\mathbf{x}} + \mathbf{e} \tag{2.27}$$

where x is the projection coordinate of the mass center onto the direction vector $\underline{\mathbf{x}}$ of the kinematic element, and $\mathbf{e}$ is a vector representing the eccentricity of the mass center to the direction axis $\underline{\mathbf{x}}$ of the kinematic element ($\mathbf{e} \perp \underline{\mathbf{x}}$). If the kinematic element is axis-symmetric and it has a uniform mass distribution along $\underline{\mathbf{x}}$, then x and $\mathbf{e}$ become:

$$x = \frac{d}{2}\,, \qquad \mathbf{e} = \mathbf{0} \tag{2.28}$$

2.6.2 Rotational Velocity and Acceleration

Lemma 1 *The rotational velocity of a kinematic element, expressed in a fixed reference frame with respect to the same fixed reference frame, can be directly written with its unit direction vector, the velocity of it and the angular velocity of a kinematic element around its unit direction vector:*

$$\boldsymbol{\omega} = \underline{\mathbf{x}} \times \dot{\underline{\mathbf{x}}} + \dot{\theta}\,\underline{\mathbf{x}} \tag{2.29}$$

Proof of Lemma 1: The velocity of a kinematic element's unit direction vector, subject to an arbitrary rotational velocity $\boldsymbol{\omega}$, is governed by the following equation:

$$\dot{\underline{\mathbf{x}}} = \boldsymbol{\omega} \times \underline{\mathbf{x}} \tag{2.30}$$

The arbitrary rotational velocity $\boldsymbol{\omega}$ can be written as follows:

$$\boldsymbol{\omega} = \dot{q}\,\underline{\mathbf{z}}_q \tag{2.31}$$

where $\dot{q}$ is the angular velocity of a kinematic element around an arbitrary instantaneous unit vector $\underline{\mathbf{z}}_q$. Then, substituting (2.31) into (2.30) yields:

$$\dot{\underline{\mathbf{x}}} = (\,\dot{q}\,\underline{\mathbf{z}}_q\,) \times \underline{\mathbf{x}} \tag{2.32}$$

Expressing $\underline{\mathbf{z}}_q$ in terms of the kinematic element's frame vectors $\{\underline{\mathbf{x}}, \underline{\mathbf{y}}, \underline{\mathbf{z}}\}$ as below:

$$\underline{\mathbf{z}}_q = a\,\underline{\mathbf{x}} + b\,\underline{\mathbf{y}} + c\,\underline{\mathbf{z}} \tag{2.33}$$

then (2.32) is rewritten as follows:

$$\dot{\underline{\mathbf{x}}} = (\,a\,\dot{q}\,\underline{\mathbf{x}} + b\,\dot{q}\,\underline{\mathbf{y}} + c\,\dot{q}\,\underline{\mathbf{z}}\,) \times \underline{\mathbf{x}} \tag{2.34}$$

which simplifies into:

$$\dot{\underline{\mathbf{x}}} = -\,b\,\dot{q}\,\underline{\mathbf{z}} + c\,\dot{q}\,\underline{\mathbf{y}} \tag{2.35}$$

If (2.35) is cross-producted from the left hand side with $\underline{\mathbf{x}}$, it yields:

$$\underline{\mathbf{x}} \times \dot{\underline{\mathbf{x}}} = b\,\dot{q}\,\underline{\mathbf{y}} + c\,\dot{q}\,\underline{\mathbf{z}} \tag{2.36}$$

and by adding the term $(\,a\,\dot{q}\,\underline{\mathbf{x}}\,)$ to the both sides of (2.36), we end up with:

$$a\,\dot{q}\,\underline{\mathbf{x}} + \underline{\mathbf{x}} \times \dot{\underline{\mathbf{x}}} = \boldsymbol{\omega} \tag{2.37}$$

which can be deduced to:

$$\dot{\theta}\,\underline{\mathbf{x}} + \underline{\mathbf{x}} \times \dot{\underline{\mathbf{x}}} = \boldsymbol{\omega} \tag{2.38}$$

The term $(\dot{\theta}\,\underline{\mathbf{x}})$ corresponds to the rotation of the kinematic element around itself and the term $(\underline{\mathbf{x}} \times \dot{\underline{\mathbf{x}}})$ corresponds to the rotation component which changes the direction $\underline{\mathbf{x}}$. □

Remark: In cases where the kinematic element has an extrinsic mobility influenced only by a revolute joint ($\mathbf{R}$) and an intrinsic mobility without a revolute joint ($\mathbf{R}$), the rotation of this kinematic element around itself is not possible and the rotational velocity can be written as follows:

$$if\ (extrinsic\,mobility\ joint) = (\mathbf{R})\ \&\ (intrinsic\,mobility\ joint) \neq (\mathbf{R}),\ then$$

$$\boldsymbol{\omega} \triangleq \underline{\mathbf{x}} \times \dot{\underline{\mathbf{x}}} \tag{2.39}$$

as long as rigidity is preserved.

In some other cases, even if the kinematic element turns around itself (due to universal or spherical joints), this does not change the configuration of the mechanism that it exists in (i.e. just creates an internal mechanism mobility). Then, the term ($\dot{\theta}\,\underline{\mathbf{x}}$) is useless and can be dropped while expressing the rotational velocity of this kinematic element.

Thus, the rotational acceleration of a kinematic element can be then written regarding (2.29) as below:

$$\dot{\boldsymbol{\omega}} = \underline{\mathbf{x}} \times \ddot{\underline{\mathbf{x}}} + \ddot{\theta}\,\underline{\mathbf{x}} + \dot{\theta}\,\dot{\underline{\mathbf{x}}} \tag{2.40}$$

2.6.3 Translational Velocity and Acceleration

The translational velocities of a kinematic element are as follows:

$$\dot{\mathbf{B}} = \dot{\mathbf{A}} + \dot{d}\,\underline{\mathbf{x}} + d\,\dot{\underline{\mathbf{x}}} \tag{2.41}$$

$$\dot{\mathbf{S}} = \dot{\mathbf{A}} + \dot{x}\,\underline{\mathbf{x}} + x\,\dot{\underline{\mathbf{x}}} + \dot{\mathbf{e}} \tag{2.42}$$

where, thanks to rigidity,

$$\dot{\mathbf{e}} = \boldsymbol{\omega} \times \mathbf{e} \tag{2.43}$$

The translational accelerations of a kinematic element are as follows:

$$\ddot{\mathbf{B}} = \ddot{\mathbf{A}} + \ddot{d}\,\underline{\mathbf{x}} + 2\,\dot{d}\,\dot{\underline{\mathbf{x}}} + d\,\ddot{\underline{\mathbf{x}}} \tag{2.44}$$

$$\ddot{\mathbf{S}} = \ddot{\mathbf{A}} + \ddot{x}\,\underline{\mathbf{x}} + 2\,\dot{x}\,\dot{\underline{\mathbf{x}}} + x\,\ddot{\underline{\mathbf{x}}} + \ddot{\mathbf{e}} \tag{2.45}$$

where

$$\ddot{\mathbf{e}} = \dot{\boldsymbol{\omega}} \times \mathbf{e} + \boldsymbol{\omega} \times \dot{\mathbf{e}} \tag{2.46}$$

If a kinematic element is *homogenous*, *symmetric* and has a *constant length*, then its translational velocities and accelerations can be represented as follows:

$$\dot{\mathbf{B}} = \dot{\mathbf{A}} + d\,\dot{\underline{\mathbf{x}}}, \qquad \ddot{\mathbf{B}} = \ddot{\mathbf{A}} + d\,\ddot{\underline{\mathbf{x}}} \tag{2.47}$$

$$\dot{\mathbf{S}} = \dot{\mathbf{A}} + x\,\dot{\underline{\mathbf{x}}}, \qquad \ddot{\mathbf{S}} = \ddot{\mathbf{A}} + x\,\ddot{\underline{\mathbf{x}}} \tag{2.48}$$

2.7 Dynamics of a « Kinematic Element »

Disturbing a kinematic element with some active forces (e.g., contact forces, distance forces, actuator torques/forces) will cause reactive inertial forces (linear and angular momentums) and frictional forces. In the following subsections, we will explore these forces in detail. The reader is referred to Section 2.5.3, if needs to remember the dynamic state variables of a kinematic element.

2.7.1 Active Forces and Torques

2.7.1.1 Forces and Torques of Actuators

We write the actuator force/torque vectors ($\mathbf{f}_d$, $\boldsymbol{\tau}_{\underline{\mathbf{x}}}$, $\boldsymbol{\tau}_\theta$), which can exist in a kinematic element, as follows:

$$\mathbf{f}_d \,=\, f_d\,\underline{\mathbf{x}}\,, \qquad \boldsymbol{\tau}_{\underline{\mathbf{x}}} \,=\, \tau_{\underline{\mathbf{x}}}\,\underline{\mathbf{z}}\,, \qquad \boldsymbol{\tau}_\theta \,=\, \tau_\theta\,\underline{\mathbf{x}} \tag{2.49}$$

where f_d is the intrinsic translational force of a kinematic element due to the intrinsic active prismatic actuator whose orientation is along the direction ($\underline{\mathbf{x}}$) of the kinematic element; $\tau_{\underline{\mathbf{x}}}$ is the extrinsic torque of a kinematic element due to the active rotary actuator whose rotation axis is $\underline{\mathbf{z}}$ and whose rotating kinematic element is oriented along $\underline{\mathbf{x}}$; τ_θ is the intrinsic torque of a kinematic element due to the intrinsic active rotary actuator whose rotation axis is $\underline{\mathbf{x}}$ and whose self-rotating kinematic element is oriented along $\underline{\mathbf{x}}$.

2.7.1.2 Force of Gravity

Afterwards, the active force of gravity ($\mathbf{f}_\mathbf{g}$), which is assumed to act at the center of mass of a kinematic element, is given as below:

$$\mathbf{f}_\mathbf{g} \,=\, m\,\mathbf{g} \tag{2.50}$$

where m is the mass of the kinematic element and $\mathbf{g}$ is the gravity acceleration vector oriented towards the center of the Earth.

2.7.2 Reactive Forces and Torques

2.7.2.1 Inertial Forces and Torques

Inertial forces and torques will appear at a kinematic element due to its accelerated matter inertia and mass. These inertial forces and torques offer resistance to change of motion of the kinematic element.

Inertial Forces and Torques of Actuators: The rotary actuator inertial torque $\boldsymbol{\tau}^*_{\underline{\mathbf{x}}}$, which appears due to the extrinsic torque $\boldsymbol{\tau}_{\underline{\mathbf{x}}}$, can be written as follows:

$$\boldsymbol{\tau}^*_{\underline{\mathbf{x}}} \,=\, -\,\mathcal{I}_z\,(\ddot{q}\,\underline{\mathbf{z}}) \,=\, -\,\mathcal{I}_z\,(\,\underline{\mathbf{x}} \,\times\, \ddot{\underline{\mathbf{x}}}\,) \tag{2.51}$$

where $\mathcal{I}_z$ is the rotary inertia of the actuator around the $\underline{\mathbf{z}}$ axis and $\ddot{q}$ is the angular acceleration of the actuator. The rotary actuator inertial torque $\boldsymbol{\tau}_\theta^*$, which appears due to the intrinsic torque $\boldsymbol{\tau}_\theta$, can be written as below:

$$\boldsymbol{\tau}_\theta^* \;=\; -\,\mathcal{I}_x\,(\ddot{\theta}\,\underline{\mathbf{x}}) \tag{2.52}$$

where $\mathcal{I}_x$ is the rotary inertia of the actuator around the $\underline{\mathbf{x}}$ axis and $\ddot{\theta}$ is the angular acceleration of the actuator. The linear actuator inertial force $\mathbf{f}_d^*$ can be written as follows:

$$\mathbf{f}_d^* \;=\; -\,m_d\,\ddot{d}\,\underline{\mathbf{x}} \tag{2.53}$$

where m_d is the mass moved inside the kinematic element by the linear actuator and $\ddot{d}$ is the linear acceleration coordinate of this linear actuator.

Body Inertial Force and Torque: The accelerated matter mass and inertia of the kinematic element produce the inertial force and torque. The inertial force and torque ($\mathbf{f}^*\,,\boldsymbol{\tau}^*$) of a kinematic element can be calculated with the Newton-Euler equations:

$$\mathbf{f}^* \;=\; -\,m\,\ddot{\mathbf{S}}\,, \qquad\qquad \boldsymbol{\tau}^* \;=\; -\,\mathcal{I}^T\,\dot{\boldsymbol{\omega}} \;-\; \boldsymbol{\omega} \;\times\; (\mathcal{I}^T\,\boldsymbol{\omega}) \tag{2.54}$$

where m, $\ddot{\mathbf{S}}$, $\mathcal{I}$ and $\boldsymbol{\omega}$ are the mass, the translational acceleration vector of the mass center, the central inertia dyadic and the rotational velocity vector of the kinematic element, respectively.

2.7.2.2 Frictional Forces and Torques

Frictional forces/torques ($\bar{\mathbf{f}}_d$, $\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}}$, $\bar{\boldsymbol{\tau}}_\theta$) will appear at the joint locations of a kinematic element due to its relative mobility. A frictional force/torque offers resistance on the motion of a kinematic element. The extrinsic frictional torque of a rotating kinematic element can be calculated as follows:

$$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}} \;=\; -\,\left(\,\bar{\tau}_{v(\underline{\mathbf{x}})}\,\dot{q} \;+\; \bar{\tau}_{c(\underline{\mathbf{x}})}\,sign(\dot{q})\,\right)\,\underline{\mathbf{z}} \;=\; -\,\bar{\tau}_{v(\underline{\mathbf{x}})}\,\boldsymbol{\omega} \;-\; \bar{\tau}_{c(\underline{\mathbf{x}})}\,sign(\,\boldsymbol{\omega}^T\,\underline{\mathbf{z}}\,)\,\underline{\mathbf{z}} \tag{2.55}$$

where $\bar{\tau}_{v(\underline{\mathbf{x}})}$ and $\bar{\tau}_{c(\underline{\mathbf{x}})}$ are the viscous and Coulomb friction coefficients of the extrinsic joint (q) of the kinematic element; $\boldsymbol{\omega}$ is the relative rotational velocity vector between the extrinsic joint and the rest of the kinematic element; and $\underline{\mathbf{z}}$ is the axis of rotation of the kinematic element. The intrinsic frictional torque of a self-rotating kinematic element can be calculated as follows:

$$\bar{\boldsymbol{\tau}}_\theta \;=\; -\,\left(\,\bar{\tau}_{v(\theta)}\,\dot{\theta} \;+\; \bar{\tau}_{c(\theta)}\,sign(\dot{\theta})\,\right)\,\underline{\mathbf{x}} \tag{2.56}$$

where $\bar{\tau}_{v(\theta)}$ and $\bar{\tau}_{c(\theta)}$ are the viscous and Coulomb friction coefficients of the intrinsic joint (θ) of the kinematic element whose orientation is along $\underline{\mathbf{x}}$. The intrinsic frictional force of an intrinsically translating kinematic element can be calculated as below:

$$\bar{\mathbf{f}}_d \;=\; -\,\left(\,\bar{f}_{v(d)}\,\dot{d} \;+\; \bar{f}_{c(d)}\,sign(\dot{d})\,\right)\,\underline{\mathbf{x}} \tag{2.57}$$

where $\bar{f}_{v(d)}$ and $\bar{f}_{c(d)}$ are the viscous and Coulomb friction coefficients of the intrinsic translational joint (d) of the kinematic element whose direction is $\underline{\mathbf{x}}$.

2.8 Physical Formation of a Parallel Robot

Here, we homogeneously define the physical formation of a parallel robot through this new construction *kinematic element*.

2.8.1 A Base Platform

A *base platform* is composed of base element(s) and its formation can be noted as follows:

$$(base\ platform) : \{\, (base\ element)_1,\ \ldots,\ (base\ element)_k \,\} \tag{2.58}$$

with $k \geq 1$ and where a base element is a kinematic element without motion.

2.8.2 A Kinematic Leg

A *kinematic leg* is a chain of consecutive kinematic elements and it can be formed as follows:

$$(kinematic\ leg) : \{\, (kinematic\ element)_1,\ \ldots,\ (kinematic\ element)_j \,\} \tag{2.59}$$

where j is the number of kinematic elements, whose value may be different in another kinematic leg of the parallel robot.

2.8.3 Nacelle

A nacelle is made of an articulated set of nacelle elements. The form of a nacelle can be defined as follows:

$$(nacelle) : \{\, (nacelle\ element)_1,\ \ldots,\ (nacelle\ element)_n \,\} \tag{2.60}$$

where n is number of nacelle elements. One of these nacelle elements is the moving platform, or if $n = 1$, then the nacelle itself is the moving platform. Therefore, a moving platform is made of a single nacelle element. The moving platform can be always a nacelle element, but any nacelle element is not necessarily the moving platform. Hence, the moving platform can be defined as below:

$$(moving\ platform) : \{\, nacelle\ element \,\} \tag{2.61}$$

where a nacelle element is a kinematic element. A parallel robot can have either a nacelle or only the moving platform. If a nacelle exists in a parallel robot, then it is attached to all of the kinematic legs, and its moving platform carries a payload. Otherwise, when the nacelle does not exist, then the moving platform is directly attached to all of the kinematic legs, and it carries the payload.

2.8.4 A Parallel Robot

A *parallel robot* is composed of a base platform, k kinematic legs and a nacelle. Its formation can be written as follows:

$$(parallel\ robot) : \{\, (base\ platform),\ \begin{bmatrix} (kinematic\ leg)_1 \\ \vdots \\ (kinematic\ leg)_k \end{bmatrix},\ (nacelle)\,\} \tag{2.62}$$

2.9 Distribution of Nacelle Dynamics

Here, the distribution of nacelle dynamics has not yet been solved completely. We try, however, to give some solutions for this problem.

If the nacelle of a parallel robot has equal number of nacelle elements with the number of kinematic legs, then the dynamics of nacelle can be shared such that a kinematic leg contains an additional nacelle element. Otherwise, if the nacelle of a parallel robot has different number of nacelle elements than the number of kinematic legs, then we show how to treat this case by examples in the following subsections.

2.9.1 Moving Platform Distribution

In order to share equally the dynamics of the moving platform by each of the kinematic legs of the parallel robot, we will divide the moving platform into k virtual platform elements in the sense of inertia and mass. The number of virtual platform elements is equal to the number of kinematic legs. So, the new form of the moving platform can be written as below:

$$(moving\ platform) : \{\, (virtual\ platform\ element)_1,\ \ldots,\ (virtual\ platform\ element)_k\,\} \tag{2.63}$$

where a virtual platform element is a kinematic element with a virtual mass and a virtual inertia.

Mass Distribution: The mass m of the moving platform of the Gough-Stewart parallel robot can be divided into 6 point masses (i.e., $m/6$). These point masses are concentric and located at the same position with the moving platform's real mass center. Figure 2.2 illustrates the mass distribution of the Gough-Stewart parallel robot's moving platform.

Inertia Distribution: For example, let the inertia of the moving platform of the Gough-Stewart parallel robot be $\mathcal{I}$. This inertia is calculated about a frame which is fixed at the mass center of the moving platform. This moving platform can be divided into 6 virtual pieces such that each kinematic leg of the Gough-Stewart parallel robot can be augmented with one of these pieces. Then, let the inertias of these pieces, calculated again about the same frame which is fixed at the mass center of the moving platform, be $\mathcal{I}_1$, $\mathcal{I}_2$, $\mathcal{I}_3$, $\mathcal{I}_4$, $\mathcal{I}_5$ and $\mathcal{I}_6$. Then, we can write the total inertia of the moving platform in terms of the inertias of the virtual pieces as follows:

$$\mathcal{I} = \sum_{i=1}^{6} \mathcal{I}_i \tag{2.64}$$

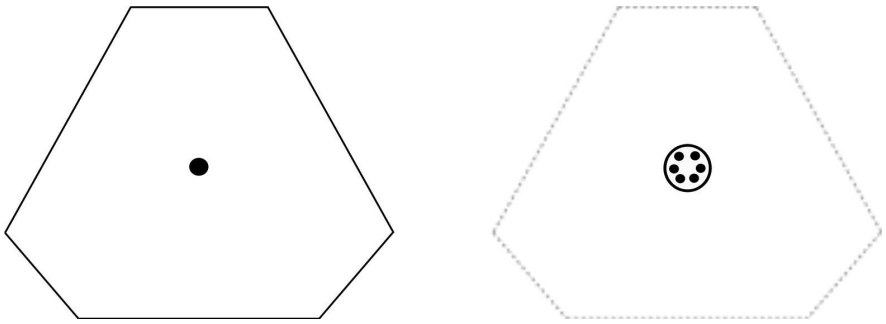


Figure 2.2 – (Left): The moving platform of the Gough-Stewart parallel robot. The black circle is the mass center of the platform. (Right): The moving platform with 6 point masses. These point masses are concentric and located at the same position with the moving platform's real mass center.

Figure 2.3 illustrates the inertia distribution of the Gough-Stewart parallel robot's moving platform.

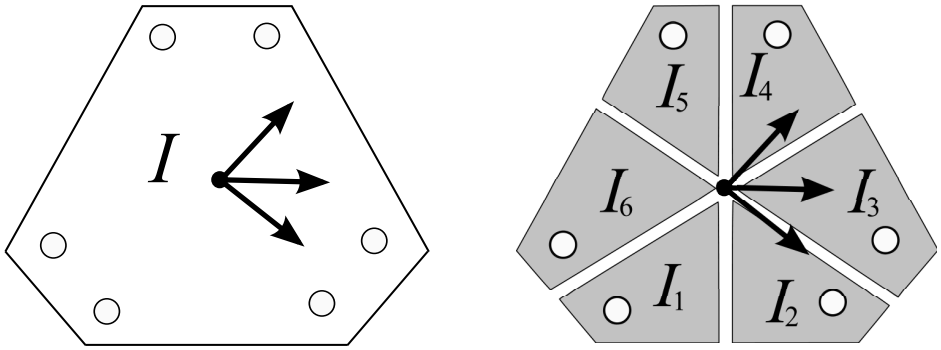


Figure 2.3 – (Left): The moving platform of the Gough-Stewart parallel robot and its inertia calculated about a given frame. The white circles are the connection points of the kinematic legs. (Right): The 6 virtual pieces of the moving platform and their inertias calculated around the same frame.

Example of a Distributed Moving Platform: Figure 2.4 shows how the Gough-Stewart parallel robot's moving platform can be split into 6 virtual platform elements. The masses of these virtual platform elements are assigned to be $m/6$ as explained in the mass distribution subsection, and their inertias are assigned to be $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3, \mathcal{I}_4, \mathcal{I}_5, \mathcal{I}_6$ as explained above in the inertia distribution subsection.

Remark: Since the moving platform is rigid, any virtual platform element's rotational velocity is always equal to the real moving platform's rotational velocity. Thus, total dynamics of all the virtual elements will be also equal to the dynamics of the moving platform.

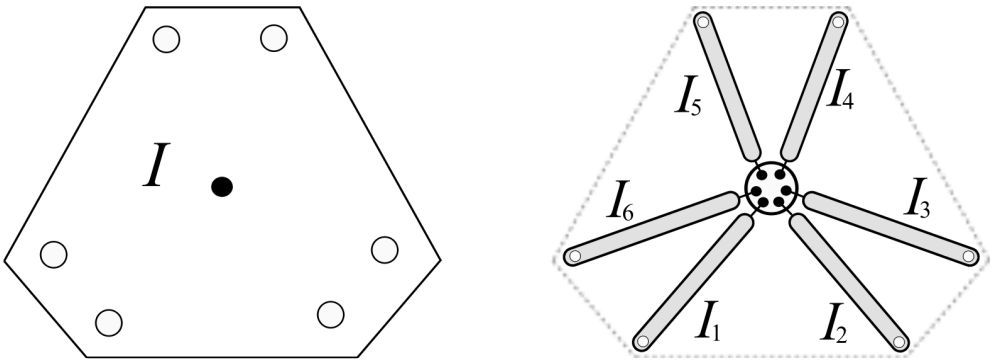


Figure 2.4 – (Left): The moving platform of the Gough-Stewart parallel robot. The white circles are the connection points of the kinematic legs, and the black circle is the mass center of the platform. (Right): The distributed moving platform with 6 virtual platform elements. The mass m of the platform is shared by six of the virtual platform elements equally (i.e., $m/6$). Their mass centers are concentric and located at the same position with the moving platform's real mass center. The inertia $\mathcal{I}$ of the platform is shared by these virtual platform elements (i.e., $\mathcal{I} = \mathcal{I}_1 + \ldots + \mathcal{I}_6$).

2.9.2 Nacelle Distribution

Figure 2.5 shows an example of the H4 parallel robot's nacelle. Its nacelle is composed of 3 articulated nacelle elements while it has 4 kinematic legs. The moving platform (i.e., one of the nacelle elements) of the nacelle of H4 parallel robot is located in the middle of the two parallel nacelle elements and connects them with revolute joints. The parallel nacelle elements have only translational motion, therefore they do not have inertias. However, their relative motion turns the moving platform (i.e., the middle nacelle element) and thus the moving platform has an inertia $\mathcal{I}$. Each of the parallel nacelle elements can be divided into 2 virtual nacelle elements so that we have a virtual nacelle element per kinematic leg. The kinematic legs are now augmented with these virtual nacelle elements up to the connection points of the moving platform. We can divide also the moving platform of the H4 parallel robot into 4 virtual platform elements in a similar way as explained in the previous section for a Gough-Stewart parallel robot's moving platform. So, the inertia $\mathcal{I}$ of the moving platform is shared by 4 virtual platform elements whose inertias are $\mathcal{I}_1$, $\mathcal{I}_2$, $\mathcal{I}_3$ and $\mathcal{I}_4$. Note that the inertia $\mathcal{I}$ of the moving platform should be recalculated at each iteration (or anytime its configuration changes) before sharing it among the virtual platform elements.

For the other types of nacelles, one can use a similar concept to share equally the masses and the inertias of the nacelle elements to the each of the parallel robot's kinematic legs. Finally, we remark that there is still some work to do to formalize completely the nacelle dynamics.

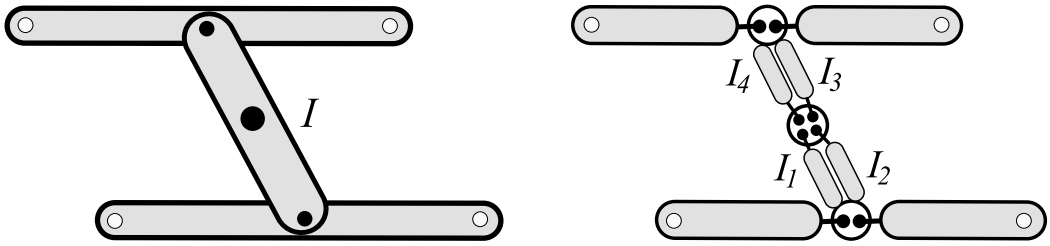


Figure 2.5 – (Left): The nacelle of the H4 parallel robot. It is composed of 3 articulated nacelle elements and it has 4 kinematic legs. The white circles are the connection points of the kinematic legs, and the black circles are the mass centers of the nacelle elements. (Right): The reconstruction of the nacelle with 8 virtual nacelle elements. The mass of each nacelle element is shared equally by virtual nacelle elements. The mass centers of these virtual kinematic elements are concentric and located at the same positions with the nacelle elements' real mass centers. The inertia $\mathcal{I}$ of the moving platform is shared by 4 virtual platform elements (i.e., $\mathcal{I} = \mathcal{I}_1 + \mathcal{I}_2 + \mathcal{I}_3 + \mathcal{I}_4$).

2.10 Sate Variables and Motion Basis of a Parallel Robot

2.10.1 State Variables

Instead of writing the geometric relations and the motion of a mechanism in terms of the independent number (n_q) of *generalized scalar coordinates* $\{q_1, \ldots, q_{n_q}\}$ (i.e., active joint values), we break out of the customary routine and we express the geometric relations and the motion with the unit direction vectors of all (n_{ke}) kinematic elements $\{\underline{\mathbf{x}}_1, \ldots, \underline{\mathbf{x}}_{n_{ke}}\}$, the varying lengths of these kinematic elements $\{d_1, \ldots, d_{n_{ke}}\}$, and the self-rotation angles around the unit directions of these kinematic elements $\{\theta_1, \ldots, \theta_{n_{ke}}\}$. That is to say, we use a redundant set of variables ($5n_{ke} \gg n_q$) and we express equations in a vector form wherever possible rather than in a scalar form.

2.10.2 Motion Basis

Choosing a motion basis, different from the first order derivatives of the independent generalized coordinates $\{\dot{q}_1, \ldots, \dot{q}_{n_q}\}$ of a robot, was proposed for the first time by Kane which is named as *generalized speeds*. Traditionally, in Kane's method the generalized speeds, u_r, are defined (still as scalars) as functions of the derivatives of a minimal set of n_q generalized scalar coordinates $\{q_1, \ldots, q_{n_q}\}$:

$$u_r \triangleq \sum_{i=1}^{n_q} \mathbf{y}_{ri}\, \dot{q}_i + \mathbf{z}_r\,, \qquad r = 1, \ldots, n_q \tag{2.65}$$

where $\mathbf{y}_{ri}$ and $\mathbf{z}_r$ are functions of $\{q_1, \ldots, q_{n_q}\}$ and the time t. The choice of these functions in (2.65) should yield a unique solution for $\{\dot{q}_1, \ldots, \dot{q}_{n_q}\}$ [KL85].

Since the redundant set of state variables that we proposed,

$$\{\underline{\mathbf{x}}_1, \ldots, \underline{\mathbf{x}}_{n_{ke}}\}, \quad \{d_1, \ldots, d_{n_{ke}}\}, \quad \{\theta_1, \ldots, \theta_{n_{ke}}\}, \tag{2.66}$$

compactly represents the configuration of the mechanism and linearizes the expressions, the choices of the generalized speeds appear spontaneously themselves. So (without needing to

inspect the expressions), we define directly *the motion basis* as the time derivatives of the redundant set of motion variables of (2.66):

$$\mathbf{u}_{xi} \triangleq \dot{\underline{\mathbf{x}}}_i \qquad u_{di} \triangleq \dot{d}_i \qquad u_{\theta i} \triangleq \dot{\theta}_i \qquad i = 1, \ldots, n_{ke} \tag{2.67}$$

Namely, $\mathbf{y}_{ri} = 1$ and $\mathbf{z}_r = 0$ in (2.65).

This definition preserves the geometric intuitiveness of the mechanism and eases the following of equations.

2.11 Kinematics of a Parallel Robot

To give the notion clearly in the rest of the context, from time to time we will refer to a simple 2 degrees of freedom (dof) five-bar mechanism which is a RRR-RR structure planar parallel robot. Figure 2.6 illustrates this 2 dof five-bar mechanism. Regarding defined kinematic element types, a kinematic leg of this robot is composed of two consecutive *bar type* kinematic elements, and this five-bar mechanism can be renamed as a 2BB parallel robot.

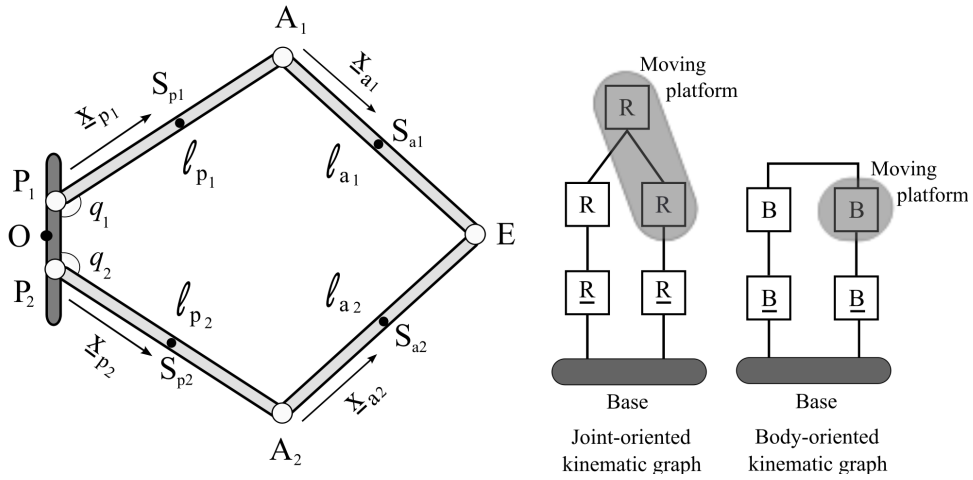


Figure 2.6 – A 2-dof planar five-bar mechanism. The revolute joints rotate around the $\underline{\mathbf{z}}_{pi}$ and $\underline{\mathbf{z}}_{ai}$ axes which are orthogonal to the paper plane. Actuators are located at $\mathbf{P}_i$ points. All the kinematic elements are homogenous and symmetric. ℓ_{pi} and ℓ_{ai} are the constant lengths of the kinematic elements. On the right side of the figure, we see the joint-oriented kinematic graph and the new body-oriented kinematic graph of the five-bar mechanism. A pseudo moving platform can be imagined as one of the identical kinematic elements (e.g., $[\mathbf{A}_2\mathbf{E}]$). The end-effector is located at point $\mathbf{E}$.

2.11.1 Mass Centers

The mass center position of the i^{th} kinematic element of a kinematic leg (with respect to a constant attachment point $\mathbf{P}$ of the kinematic leg onto the base) can be formulated by summing

the $i-1$ elements and adding finally the i^{th} mass center:

$$\mathbf{S}_i = \mathbf{P} + \sum_{j=1}^{i-1} d_j \, \underline{\mathbf{x}}_j \; + \; x_i \, \underline{\mathbf{x}}_i \; + \; \mathbf{e}_i \tag{2.68}$$

Example: So, assuming that all the kinematic elements of the five-bar mechanism are homogenous and symmetric, the mass center positions of the kinematic elements shown in Fig. 2.6 can be simply expressed as follows:

$$\mathbf{S}_{pi} = \mathbf{P}_i + \frac{\ell_{pi}}{2} \, \underline{\mathbf{x}}_{pi} \, , \qquad \mathbf{S}_{ai} = \mathbf{P}_i + \ell_{pi} \, \underline{\mathbf{x}}_{pi} + \frac{\ell_{ai}}{2} \, \underline{\mathbf{x}}_{ai} \, , \quad i = 1, 2 \tag{2.69}$$

where $\mathbf{P}_i$ is a constant point, $\{\underline{\mathbf{x}}_{pi} \, , \, \underline{\mathbf{x}}_{ai}\}$ are the unit direction vectors defining the state of the kinematic leg, and $\{\ell_{pi} \, , \, \ell_{ai}\}$ are the constant lengths of the kinematic elements.

2.11.2 Velocities

2.11.2.1 Translational Velocity

The mass center velocity of the i^{th} kinematic element of a kinematic leg (with respect to a constant attachment point $\mathbf{P}$ of the kinematic leg onto the base) can be formulated by simply time differentiating (2.68), which yields:

$$\dot{\mathbf{S}}_i = \sum_{j=1}^{i-1} \left(\dot{d}_j \, \underline{\mathbf{x}}_j \; + \; d_j \, \dot{\underline{\mathbf{x}}}_j \right) \; + \; \dot{x}_i \, \underline{\mathbf{x}}_i \; + \; x_i \, \dot{\underline{\mathbf{x}}}_i \; + \; \dot{\mathbf{e}}_i \tag{2.70}$$

Example: So, the velocities of the mass centers $\mathbf{S}_{pi}$ and $\mathbf{S}_{ai}$ of the five-bar mechanism shown in Fig. 2.6 are written as follows:

$$\dot{\mathbf{S}}_{pi} = \frac{\ell_{pi}}{2} \, \dot{\underline{\mathbf{x}}}_{pi} \, , \qquad \dot{\mathbf{S}}_{ai} = \ell_{pi} \, \dot{\underline{\mathbf{x}}}_{pi} + \frac{\ell_{ai}}{2} \, \dot{\underline{\mathbf{x}}}_{ai} \, , \qquad i = 1, 2 \tag{2.71}$$

2.11.2.2 Rotational Velocity

According to Lemma 1, the rotational velocity of any kinematic element in a kinematic leg, expressed in a fixed reference frame with respect to the same fixed reference frame (e.g., camera frame or robot base frame), will be equal to (2.29):

$$\boldsymbol{\omega}_i \; = \; \underline{\mathbf{x}}_i \; \times \; \dot{\underline{\mathbf{x}}}_i \; + \; \dot{\theta}_i \, \underline{\mathbf{x}}_i \tag{2.72}$$

Example: So, the rotational velocity vectors of the kinematic elements $\{\mathbf{PA}\}_i$ and $\{\mathbf{AE}\}_i$ of the i^{th} kinematic leg of the five-bar mechanism given in Fig. 2.6 can be calculated as below:

$$\boldsymbol{\omega}_{pi} \triangleq \underline{\mathbf{x}}_{pi} \times \dot{\underline{\mathbf{x}}}_{pi} \, , \quad \boldsymbol{\omega}_{ai} \triangleq \underline{\mathbf{x}}_{ai} \times \dot{\underline{\mathbf{x}}}_{ai} \, , \qquad i = 1, 2 \tag{2.73}$$

2.11.3 Accelerations

2.11.3.1 Translational Acceleration

The mass center acceleration of the i^{th} (with respect to base) kinematic element of a kinematic leg is derived from the time derivative of the mass center velocity:

$$\ddot{\mathbf{S}}_i = \sum_{j=1}^{i-1} \left(\ddot{d}_j \, \underline{\mathbf{x}}_j \, + \, 2 \, \dot{d}_j \, \dot{\underline{\mathbf{x}}}_j \, + \, d_j \, \ddot{\underline{\mathbf{x}}}_j \right) \, + \, \ddot{x}_i \, \underline{\mathbf{x}}_i \, + \, 2 \, \dot{x}_i \, \dot{\underline{\mathbf{x}}}_i \, + \, x_i \, \ddot{\underline{\mathbf{x}}}_i \, + \, \ddot{\mathbf{e}}_i \tag{2.74}$$

Example: Then the accelerations of the mass centers $\mathbf{S}_{pi}$ and $\mathbf{S}_{ai}$ of the five-bar mechanism shown in Fig. 2.6 are computed as below:

$$\ddot{\mathbf{S}}_{pi} = \frac{\ell_{pi}}{2} \, \ddot{\underline{\mathbf{x}}}_{pi} \, , \qquad \ddot{\mathbf{S}}_{ai} = \ell_{pi} \, \ddot{\underline{\mathbf{x}}}_{pi} + \frac{\ell_{ai}}{2} \, \ddot{\underline{\mathbf{x}}}_{ai} \, , \qquad i = 1, 2 \tag{2.75}$$

2.11.3.2 Rotational Acceleration

The rotational acceleration vector of the i^{th} (with respect to base) kinematic element of a kinematic leg will be equal to (2.40):

$$\dot{\boldsymbol{\omega}}_i \, = \, \underline{\mathbf{x}}_i \, \times \, \ddot{\underline{\mathbf{x}}}_i \, + \, \ddot{\theta}_i \, \underline{\mathbf{x}}_i \, + \, \dot{\theta}_i \, \dot{\underline{\mathbf{x}}}_i \tag{2.76}$$

Example: Then the rotational accelerations of the kinematic elements $\{\mathbf{PA}\}_i$ and $\{\mathbf{AE}\}_i$ of the i^{th} kinematic leg of the five-bar mechanism given in Fig. 2.6 can be expressed as below:

$$\dot{\boldsymbol{\omega}}_{pi} \, \triangleq \, \underline{\mathbf{x}}_{pi} \, \times \, \ddot{\underline{\mathbf{x}}}_{pi} \, , \quad \dot{\boldsymbol{\omega}}_{ai} \, \triangleq \, \underline{\mathbf{x}}_{ai} \, \times \, \ddot{\underline{\mathbf{x}}}_{ai} \, , \qquad i = 1, 2 \tag{2.77}$$

2.12 Kinematic Constraints of a Parallel Robot

2.12.1 Configuration Constraints

If the positions, orientations, and lengths of the kinematic elements of a robot are restricted by the presence of each other's contacts, then the robot is said to be subject to *configuration constraints*. Such restrictions are expressed through the implicit kinematic model (ImplKM) of the robot or so-called the *holonomic constraint equation* [KL85]:

$$f \left(\mathbf{O}, \, \mathbf{P}_i, \, \mathbf{E}, \, \left\{ \underline{\mathbf{x}}_{ji}, \, d_{ji}, \, (\theta_{ji}) \right\}_{j=1,\ldots,n_{ke(i)}} , \, \boldsymbol{\xi}_{geo} \right) \, = \, \mathbf{0} \, , \quad \forall i \, = \, 1, \ldots, \, n_{leg} \tag{2.78}$$

where $\boldsymbol{\xi}_{geo}$ is the vector of constant geometric parameters, n_{leg} is the number of kinematic legs, and $n_{ke(i)}$ is the number of kinematic elements in the i^{th} kinematic leg of a parallel robot. Assuming that: the connection points of the kinematic elements are lying on the axes of the direction vectors; the self-rotations θ_{ji} of the kinematic elements do not change the positions

of these connection points; and the end-effector frame is located at the mass center of the moving platform; then, equation (2.78) can be precisely rewritten as follows:

$$\overrightarrow{\mathbf{OE}} - \sum_{j=1}^{n_{ke(i)}} d_{ji}\, \underline{\mathbf{x}}_{ji} - \overrightarrow{\mathbf{OP}_i} = \mathbf{0}\,, \qquad i = 1, \ldots, n_{leg} \tag{2.79}$$

where the sum $n_{ke(1)} + \ldots + n_{ke(n_{leg})} = n_{ke}$ is equal to the total number of kinematic elements in a parallel robot.

Example: For the five-bar mechanism shown in Fig. 2.6, the *closed-loop holonomic constraint equations* can be written as follows:

$$\overrightarrow{\mathbf{OE}} - \ell_{ai}\, \underline{\mathbf{x}}_{ai} - \ell_{pi}\, \underline{\mathbf{x}}_{pi} - \overrightarrow{\mathbf{OP}_i} = \mathbf{0}\,, \qquad i = 1, 2 \tag{2.80}$$

2.12.2 Motion Constraints

If the components of motion basis $\{\, \dot{\underline{\mathbf{x}}}_i,\, \dot{d}_i,\, \dot{\theta}_i \,\}$ of the mechanism are not mutually independent, then the mechanism is said to be subject to *motion constraints*, and the mechanism is named as a *nonholonomic system*. The motion constraints equation can be written by differentiating the configuration constraints equation (2.79), which gives:

$$M_{Ci}\, \dot{\mathbb{X}} - \sum_{j=1}^{n_{ke(i)}} \left(\dot{d}_{ji}\, \underline{\mathbf{x}}_{ji} + d_{ji}\, \dot{\underline{\mathbf{x}}}_{ji} \right) = \mathbf{0}\,, \qquad i = 1, \ldots, n_{leg} \tag{2.81}$$

where $\mathbb{X}$ is a Cartesian pose representation of the end-effector in a column-array form, and where $M_{Ci} \in \Re^{3\times\, dim(\mathbb{X})}$ is the Cartesian position kinematic matrix which relates the velocity of the Cartesian pose to the velocity of the tip position of the i^{th} kinematic leg. M_{Ci} depends on the representation of $\mathbb{X}$, the structure of the rigid moving platform (or the articulated nacelle), and the tip position of the i^{th} kinematic leg on the moving platform (or nacelle).

Example: For the five-bar mechanism shown in Fig. 2.6, the motion constraint equations are written by differentiating (2.80) as follows:

$$\dot{\mathbb{X}} - \ell_{pi}\, \dot{\underline{\mathbf{x}}}_{pi} - \ell_{ai}\, \dot{\underline{\mathbf{x}}}_{ai} = \mathbf{0}\,, \qquad i = 1, 2 \tag{2.82}$$

where $\dot{\mathbb{X}} = \dot{\mathbf{E}}$. From (2.82), we can derive the inverse differential kinematic models of the kinematic elements' variables. To do so, we exploit two properties of the vectors:

- The projection of a vector onto its velocity vector is equal to zero: $\underline{\mathbf{x}}^T\, \dot{\underline{\mathbf{x}}} = 0$,
- If vectors $\mathbf{a}$ and $\mathbf{c}$ are parallel ($\mathbf{a} \,//\, \mathbf{c}$), then $\mathbf{a}\,(\,\mathbf{b}^T\,\mathbf{c}\,) = (\,\mathbf{b}^T\,\mathbf{a}\,)\,\mathbf{c}$.

By projecting (2.82) with $\underline{\mathbf{x}}_{ai}$, we eliminate its motion variable $\dot{\underline{\mathbf{x}}}_{ai}$ from the equation:

$$\underline{\mathbf{x}}_{ai}^T\, \dot{\mathbb{X}} - \ell_{pi}\, \underline{\mathbf{x}}_{ai}^T\, \dot{\underline{\mathbf{x}}}_{pi} = 0 \tag{2.83}$$

Afterwards, multiplying the last equation with $\underline{\mathbf{y}}_{pi}$ which is parallel to $\dot{\underline{\mathbf{x}}}_{pi}$, we obtain:

$$\underline{\mathbf{y}}_{pi}\, \underline{\mathbf{x}}_{ai}^T\, \dot{\mathbb{X}} - \ell_{pi}\, \underline{\mathbf{y}}_{pi}\, \underline{\mathbf{x}}_{ai}^T\, \dot{\underline{\mathbf{x}}}_{pi} = \mathbf{0} \tag{2.84}$$

This allows us to use the second property of the vectors mentioned above. Then, we rewrite (2.84) as follows:

$$\underline{\mathbf{y}}_{pi}\,\underline{\mathbf{x}}_{ai}^{T}\,\dot{\mathbb{X}} \;-\; \ell_{pi}\;(\underline{\mathbf{x}}_{ai}^{T}\,\underline{\mathbf{y}}_{pi})\,\dot{\underline{\mathbf{x}}}_{pi} = \mathbf{0} \tag{2.85}$$

This avoids a matrix inversion while computing the inverse differential kinematic model of $\dot{\underline{\mathbf{x}}}_{pi}$:

$$\dot{\underline{\mathbf{x}}}_{pi} = M_{pi}\,\dot{\mathbb{X}}, \qquad M_{pi} \;=\; \left[\; \frac{\underline{\mathbf{y}}_{pi}\,\underline{\mathbf{x}}_{ai}^{T}}{\ell_{pi}\;(\,\underline{\mathbf{x}}_{ai}^{T}\,\underline{\mathbf{y}}_{pi}\,)} \;\right] \in \Re^{3\times 3} \tag{2.86}$$

Then, to derive the other inverse differential kinematic model related to $\dot{\underline{\mathbf{x}}}_{ai}$, we proceed as follows:

$$\dot{\mathbb{X}} \;-\; \ell_{pi}\;M_{pi}\,\dot{\mathbb{X}} \;-\; \ell_{ai}\;\dot{\underline{\mathbf{x}}}_{ai} = \mathbf{0} \tag{2.87}$$

and from (2.87) we write easily:

$$\dot{\underline{\mathbf{x}}}_{ai} \;=\; M_{ai}\,\dot{\mathbb{X}}, \quad M_{ai} \;=\; \left[\; \frac{1}{\ell_{ai}}\;(\,I_3 \;-\; \ell_{pi}\;M_{pi}\,) \;\right] \in \Re^{3\times 3} \tag{2.88}$$

where I_3 is the 3 by 3 identity matrix. Finally, we derive the inverse differential kinematic model of the active joint coordinates $\dot{q}_i$. Knowing that:

$$\boldsymbol{\omega}_{pi} \;\triangleq\; \underline{\mathbf{x}}_{pi} \;\times\; \dot{\underline{\mathbf{x}}}_{pi} \;=\; \dot{q}_i\,\underline{\mathbf{z}}_{pi} \tag{2.89}$$

we can take out $\dot{q}_i$ as below:

$$\dot{q}_i \;=\; (\,\underline{\mathbf{x}}_{pi} \;\times\; \dot{\underline{\mathbf{x}}}_{pi}\,)^{T}\,\underline{\mathbf{z}}_{pi} \tag{2.90}$$

which can be reformulated in terms of $\dot{\mathbb{X}}$ as follows:

$$\dot{q}_i \;=\; M_{qi}\,\dot{\mathbb{X}}, \quad M_{qi} \;=\; \left[\; \underline{\mathbf{z}}_{pi}^{T}\;[\,\underline{\mathbf{x}}_{pi}\,]_{\times}\;M_{pi} \;\right] \in \Re^{1\times 3} \tag{2.91}$$

where $[\cdot]_{\times}$ represents the skew-symmetric matrix of an associated cross-product vector.

2.13 Kinematic Coordinates of a Parallel Robot

Kane [KL85] expresses the linear and rotational velocities of the kinematic elements uniquely through the minimal set of generalized scalar coordinates, the generalized speeds in (2.65) and the *partial velocities*. Then, Kane writes the linear velocity for the mass center and the rotational velocity of a kinematic element as follows:

$$\dot{\mathbf{S}} = \sum_{r=1}^{n_q} \mathbf{v}_r\,u_r + \mathbf{v}_t \tag{2.92}$$

$$\boldsymbol{\omega} = \sum_{r=1}^{n_q} \mathbf{w}_r\,u_r + \mathbf{w}_t \tag{2.93}$$

where $\mathbf{v}_r$, $\mathbf{w}_r$, $\mathbf{v}_t$ and $\mathbf{w}_t$ are functions of $\{\,q_1,\ldots,q_{n_q}\,\}$ and the time t.

The vectors $\mathbf{v}_r \in \Re^{3\times 1}$ and $\mathbf{w}_r \in \Re^{3\times 1}$ are the r^{th} *partial linear and rotational velocities* of the kinematic element. So, for a kinematic element, Kane defines n_q partial linear velocities and n_q partial rotational velocities with the use of n_q scalar generalized speeds:

$$\mathbf{v}_r = \frac{\partial \dot{\mathbf{S}}}{\partial u_r}, \qquad \mathbf{w}_r = \frac{\partial \boldsymbol{\omega}}{\partial u_r}, \qquad r = 1, \ldots, n_q \tag{2.94}$$

Before proceeding on the rest of the text, we would like to first clarify the naming of some technical terms in Kane's method and in our method:

- What Kane calls "generalized coordinates", here we replace it with "state variables";
- What Kane calls "generalized speeds", here we replace it with "motion basis";
- What Kane calls "partial velocities", here we replace it with "kinematic coordinates";

Regarding the definition of our motion basis (direction vectors, lengths and rotation angles) in (2.67), the kinematic coordinates take the form of either matrices or vectors:

$$V_{xi} = \frac{\partial \dot{\mathbf{S}}}{\partial \mathbf{u}_{xi}}, \qquad W_{xi} = \frac{\partial \boldsymbol{\omega}}{\partial \mathbf{u}_{xi}}, \qquad i = 1, \ldots, n_{ke} \tag{2.95}$$

$$\mathbf{v}_{di} = \frac{\partial \dot{\mathbf{S}}}{\partial u_{di}}, \qquad \mathbf{w}_{di} = \frac{\partial \boldsymbol{\omega}}{\partial u_{di}}, \qquad i = 1, \ldots, n_{ke} \tag{2.96}$$

$$\mathbf{v}_{\theta i} = \frac{\partial \dot{\mathbf{S}}}{\partial u_{\theta i}}, \qquad \mathbf{w}_{\theta i} = \frac{\partial \boldsymbol{\omega}}{\partial u_{\theta i}}, \qquad i = 1, \ldots, n_{ke} \tag{2.97}$$

where $V_{xi} \in \Re^{3\times 3}$ and $W_{xi} \in \Re^{3\times 3}$ are the linear and rotational kinematic coordinates (matrices) of the kinematic element with respect to the i^{th} kinematic element's direction vector variable $\underline{\mathbf{x}}_i$, and where $\mathbf{v}_{di} \in \Re^{3\times 1}$ and $\mathbf{w}_{di} \in \Re^{3\times 1}$ are the linear and rotational kinematic coordinates (vectors) of the kinematic element with respect to the i^{th} kinematic element's length variable d_i, and where $\mathbf{v}_{\theta i} \in \Re^{3\times 1}$ and $\mathbf{w}_{\theta i} \in \Re^{3\times 1}$ are the linear and rotational kinematic coordinates (vectors) of the kinematic element with respect to the i^{th} kinematic element's self-rotation variable θ_i.

Example: Then, for the five-bar mechanism shown in Fig. 2.6, the motion basis will be as follows:

$$\mathbf{u}_{x1} \triangleq \underline{\dot{\mathbf{x}}}_{p1}, \qquad \mathbf{u}_{x2} \triangleq \underline{\dot{\mathbf{x}}}_{p2} \qquad \mathbf{u}_{x3} \triangleq \underline{\dot{\mathbf{x}}}_{a1} \qquad \mathbf{u}_{x4} \triangleq \underline{\dot{\mathbf{x}}}_{a2} \tag{2.98}$$

Table 2.1 tabulates the linear and rotational kinematic coordinates of this five-bar mechanism. These kinematic coordinates are algebraic expressions written from the geometric states of the kinematic elements. Note that while calculating the kinematic coordinates of a kinematic leg, the kinematic leg will have contributions only from itself since it can solely be represented by its own motion variables. The contributions from the rest of kinematic legs will be zero.

Table 2.1 – The (transposed) kinematic coordinates of the five-bar mechanism (2BB), i=1,2.

	$\partial\,\dot{\mathbf{S}}_{pi}$	$\partial\,\boldsymbol{\omega}_{pi}$	$\partial\,\dot{\mathbf{S}}_{ai}$	$\partial\,\boldsymbol{\omega}_{ai}$
$\partial\,\dot{\underline{\mathbf{x}}}_{pi}$	$\frac{\ell_{pi}}{2}\,I_3$	$[\underline{\mathbf{x}}_{pi}]^T_\times$	$\ell_{pi}\,I_3$	$\mathbf{0}$
$\partial\,\dot{\underline{\mathbf{x}}}_{ai}$	$\mathbf{0}$	$\mathbf{0}$	$\frac{\ell_{ai}}{2}\,I_3$	$[\underline{\mathbf{x}}_{ai}]^T_\times$

2.14 Dynamic Coordinates of a Parallel Robot

The forces consist of *contributing* and *non-contributing* parts for the dynamics of a robot. The computation of the dynamic coordinates (i.e., generalized forces) is concerned only with the extraction of the *contributing* parts.

The forces acting on a robot can be listed in two groups: active and reactive forces. Firstly, we will list these active and reactive forces. Then, we will explain how to compute the dynamic coordinates of a robot from its revealed active and reactive forces.

2.14.1 Listing the Active and Reactive Forces

As it is explained in Section 2.7 for a kinematic element, the active forces of a robot are similarly the *actuator forces and torques* (i.e., generated by the linear and rotary motor motions) and the *distance forces* (e.g., gravitational, magnetic). And, subsequently, the reactive forces are the *inertial forces and torques* (i.e., generated by the accelerated masses and inertias) and the *contact forces* (e.g., friction).

Example: In this example, we list all the active and reactive forces of five-bar mechanism shown in Figure 2.6.

– *Forces of Actuators and Gravity:* Parallel robots have usually a single actuator per kinematic leg. Thus, probably only one of the efforts of (2.49) will appear as an actuator force/torque in a kinematic leg. For the kinematic elements of the five-bar mechanism shown in Fig. 2.6, the active forces and torques can be written as follows:

$$\boldsymbol{\tau}_{\underline{\mathbf{x}}_{pi}} \;=\; \tau_{\underline{\mathbf{x}}_{pi}}\,\underline{\mathbf{z}}_{pi}\,, \qquad \mathbf{f}_{\mathbf{g}(pi)} \;=\; m_{pi}\,\mathbf{g}\,, \qquad \mathbf{f}_{\mathbf{g}(ai)} \;=\; m_{ai}\,\mathbf{g}\,, \quad i=1,2 \tag{2.99}$$

where $\boldsymbol{\tau}_{\underline{\mathbf{x}}_{pi}}$ is the actuator torque, $\mathbf{f}_{\mathbf{g}(pi)}$ and $\mathbf{f}_{\mathbf{g}(ai)}$ are the forces of gravity.

– *Inertial Forces of the Actuators:* For the five-bar mechanism shown in Fig. 2.6, the inertial torques of actuators can be written as follows:

$$\boldsymbol{\tau}^*_{\underline{\mathbf{x}}_{pi}} \;=\; -\,\mathcal{I}_{pi}\,(\underline{\mathbf{x}}_{pi} \,\times\, \ddot{\underline{\mathbf{x}}}_{pi})\,, \quad i=1,2 \tag{2.100}$$

– *Body Inertial Forces of the Kinematic Elements:* For the five-bar mechanism shown in Fig. 2.6, the inertial forces and torques of the kinematic elements can be written as follows:

$$\mathbf{f}^*_{pi} \;=\; -\,m_{pi}\,\ddot{\mathbf{S}}_{pi}\,, \qquad \boldsymbol{\tau}^*_{pi} \;=\; -\,\mathcal{I}^T_{pi}\,\dot{\boldsymbol{\omega}}_{pi} \,-\, \boldsymbol{\omega}_{pi} \,\times\, (\mathcal{I}^T_{pi}\,\boldsymbol{\omega}_{pi})\,, \quad i=1,2 \tag{2.101}$$

$$\mathbf{f}^*_{ai} \;=\; -\,m_{ai}\,\ddot{\mathbf{S}}_{ai}\,, \qquad \boldsymbol{\tau}^*_{ai} \;=\; -\,\mathcal{I}^T_{ai}\,\dot{\boldsymbol{\omega}}_{ai} \,-\, \boldsymbol{\omega}_{ai} \,\times\, (\mathcal{I}^T_{ai}\,\boldsymbol{\omega}_{ai})\,, \quad i=1,2 \tag{2.102}$$

– *Frictional Forces:* For the five-bar mechanism shown in Fig. 2.6, the frictional torques can be written as follows:

$$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{pi}} \;=\; -\,\bar{\tau}_{v(\underline{\mathbf{x}}_{pi})}\,\boldsymbol{\omega}_{pi} \;-\; \bar{\tau}_{c(\underline{\mathbf{x}}_{pi})}\,sign(\,\boldsymbol{\omega}_{pi}^{T}\,\underline{\mathbf{z}}_{pi}\,)\,\underline{\mathbf{z}}_{pi}\,, \qquad i=1,2 \tag{2.103}$$

where $\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{pi}}$ is the extrinsic actuator frictional torque. Since the actuators are placed in the fixed base platform, the relative rotational velocity vector is directly equal to the velocity of the actuator. Then, the frictional torques on the passive joints are as follows:

$$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{ai}} \;=\; -\,\bar{\tau}_{v(\underline{\mathbf{x}}_{ai})}\;(\,\boldsymbol{\omega}_{ai} \;-\; \boldsymbol{\omega}_{pi}\,) - \bar{\tau}_{c(\underline{\mathbf{x}}_{ai})}\,sign\left(\,(\,\boldsymbol{\omega}_{ai} \;-\; \boldsymbol{\omega}_{pi}\,)^{T}\;\underline{\mathbf{z}}_{ai}\right)\underline{\mathbf{z}}_{ai}\,, \qquad i=1,2 \tag{2.104}$$

We can now list all the local forces and torques for the five-bar mechanism as in Table 2.2.

Table 2.2 – The local forces and torques of the five-bar mechanism, i=1,2.

	$Active$		$Friction$		$Inertia^{*}$	
	$Actuator$	$Gravity$	$Actuator$	$PassiveJoint$	$Actuator$	$Element$
$Forces\,(pi)$	$\mathbf{0}$	$\mathbf{f}_{\mathbf{g}(pi)}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{f}^{*}_{pi}$
$Torques\,(pi)$	$\tau_{\underline{\mathbf{x}}_{pi}}\,\underline{\mathbf{z}}_{pi}$	$\mathbf{0}$	$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{pi}}$	$\mathbf{0}$	$\boldsymbol{\tau}^{*}_{\underline{\mathbf{x}}_{pi}}$	$\boldsymbol{\tau}^{*}_{pi}$
$Forces\,(ai)$	$\mathbf{0}$	$\mathbf{f}_{\mathbf{g}(ai)}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{f}^{*}_{ai}$
$Torques\,(ai)$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{ai}}$	$\mathbf{0}$	$\boldsymbol{\tau}^{*}_{ai}$

2.14.2 Computing Dynamic Coordinates

Here, for the computation of dynamic Coordinates, Kane's method is used. This method simply eliminates the *non-contributing forces* by projecting the resultant forces and torques, which act on the mass centers of the kinematic elements, onto the motion directions (i.e., kinematic coordinates) of the kinematic elements:

$$\left[\begin{array}{c} A \\ Dynamic \\ Coordinate \\ of\ a \\ Kinematic \\ Element \end{array}\right]_{r} = \sum_{i=1}^{n_{ke}} \left(\left[\left(\begin{array}{c} Linear \\ Kinematic \\ Coordinate \end{array} \right)^{T}_{ir} \left(\begin{array}{c} Rotational \\ Kinematic \\ Coordinate \end{array} \right)^{T}_{ir} \right] \left[\begin{array}{c} \sum Force_{\,i} \\ \\ \sum Torque_{\,i} \end{array} \right] \right) \tag{2.105}$$

where n_{ke} is the total number of kinematic elements in a parallel robot, and $r \in \{\,\dot{\underline{\mathbf{x}}}_{j},\, \dot{d}_{j},\, \dot{\theta}_{j}\,\}$ with $j = 1, \ldots, n_{ke}$ shows that a kinematic coordinate corresponds to which component of the motion basis. One dynamic coordinate per motion basis component is computed. Each kinematic element has 3 dynamic coordinates corresponding to its own motion basis components $\{\,\dot{\underline{\mathbf{x}}},\, \dot{d},\, \dot{\theta}\,\}$. Totally, $3\,n_{ke}$ dynamic coordinates are computed for n_{ke} kinematic elements.

Example: The dynamic coordinates of the five-bar mechanism (2BB) shown in Fig. 2.6 can be simply computed through the matrix-wise multiplication of the Tables 2.1 (transposed kine-

matic coordinates) and 2.2 (sum of the local forces and torques).

$$\begin{bmatrix} \mathbb{F}_{\underline{x}_{pi}} \\ \mathbb{F}_{\underline{x}_{ai}} \end{bmatrix} = \begin{bmatrix} Kinematic \\ Coordinates \\ Table\ 2.1 \end{bmatrix}_{(2\times 4)} \begin{bmatrix} Sum\ of \\ Forces \\ Torques \\ Table\ 2.2 \end{bmatrix}_{(4\times 1)}$$

which can be explicitly written as follows:

$$\begin{bmatrix} \mathbb{F}_{\underline{x}_{pi}} \\ \mathbb{F}_{\underline{x}_{ai}} \end{bmatrix} = \begin{bmatrix} \frac{\ell_{pi}}{2} I_3 & [\underline{\mathbf{x}}_{pi}]^T_{\times} & \ell_{pi} I_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{\ell_{ai}}{2} I_3 & [\underline{\mathbf{x}}_{ai}]^T_{\times} \end{bmatrix} \begin{bmatrix} \mathbf{f}_{\mathbf{g}(pi)} + \mathbf{f}^*_{pi} \\ \tau_{\underline{\mathbf{x}}_{pi}} \underline{\mathbf{z}}_{pi} + \bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{pi}} + \boldsymbol{\tau}^*_{\underline{\mathbf{x}}_{pi}} + \boldsymbol{\tau}^*_{pi} \\ \mathbf{f}_{\mathbf{g}(ai)} + \mathbf{f}^*_{ai} \\ \bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{ai}} + \boldsymbol{\tau}^*_{ai} \end{bmatrix} \quad (2.106)$$

where the dynamic coordinates, $\mathbb{F}_{\underline{x}_{pi}}$ and $\mathbb{F}_{\underline{x}_{ai}}$, are exiting forces effecting the rotations of the bars of the mechanism. In other words, these rotations of the bars are the result of the total work done by these exiting forces along the displacement directions $\dot{\underline{\mathbf{x}}}_{pi}$ and $\dot{\underline{\mathbf{x}}}_{ai}$.

2.15 Dynamic Constraints of a Parallel Robot

The dynamic constraints of a parallel robot can be written from d'Alembert's principle of virtual work as follows:

$$\sum_{i=1}^{n_{ke}} \left(\mathbb{F}^T_{\underline{x}_i} \dot{\underline{\mathbf{x}}}_i + \mathbb{F}_{d_i} \dot{d}_i + \mathbb{F}_{\theta_i} \dot{\theta}_i \right) = 0 \quad (2.107)$$

where $\mathbb{F}_{\underline{x}_i} \in \Re^{3\times 1}$, $\mathbb{F}_{d_j} \in \Re^{1\times 1}$ and $\mathbb{F}_{\theta_k} \in \Re^{1\times 1}$ are the corresponding dynamic coordinates.

The dynamic constraints (2.107) can be reformulated through the known motion constraint models which relate the kinematic elements' motions to the velocity of the end-effector pose (i.e., to a motion basis of the constraint space):

$$\dot{\underline{\mathbf{x}}}_i = M_{\underline{x}_i} \dot{\mathbb{X}}, \quad \dot{d}_i = M_{d_i} \dot{\mathbb{X}}, \quad \dot{\theta}_i = M_{\theta_i} \dot{\mathbb{X}} \quad (2.108)$$

The substitution of (2.108) into (2.107) yields:

$$\sum_{i=1}^{n_{ke}} \left(\mathbb{F}^T_{\underline{x}_i} (M_{\underline{x}_i} \dot{\mathbb{X}}) + \mathbb{F}_{d_i} (M_{d_i} \dot{\mathbb{X}}) + \mathbb{F}_{\theta_i} (M_{\theta_i} \dot{\mathbb{X}}) \right) = 0 \quad (2.109)$$

Eliminating $\dot{\mathbb{X}}$ from (2.109), dynamic constraints take the final form as below:

$$\sum_{i=1}^{n_{ke}} \left(M^T_{\underline{x}_i} \mathbb{F}_{\underline{x}_i} + M^T_{d_i} \mathbb{F}_{d_i} + M^T_{\theta_i} \mathbb{F}_{\theta_i} \right) = \mathbf{0}_{r\times 1} \quad (2.110)$$

Remark: Equation (2.110) can be proved probably by a better solution which passes through the differential implicit kinematic model (DImplKM) of a robot rather than passing through the expressions in (2.108).

Example: Exploiting (2.110), the dynamic constraints of the five-bar mechanism (2$\underline{\text{B}}$B) are written as follows:

$$\sum_{i=1}^{2} \left(M_{pi}^T \mathbb{F}_{\underline{x}_{pi}} \; + \; M_{ai}^T \mathbb{F}_{\underline{x}_{ai}} \right) \; = \; \mathbf{0}_{3\times 1} \tag{2.111}$$

$$M_p^T \; \mathbb{F}_{\underline{x}_p} \; + \; M_a^T \; \mathbb{F}_{\underline{x}_a} \; = \; \mathbf{0}_{3\times 1} \tag{2.112}$$

where $\mathbb{F}_{\underline{x}_p} \in \Re^{6\times 1}$ and $\mathbb{F}_{\underline{x}_a} \in \Re^{6\times 1}$ are the stacked vectors of the dynamic coordinates of $\mathbb{F}_{\underline{x}_{pi}} \in \Re^{3\times 1}$ and $\mathbb{F}_{\underline{x}_{ai}} \in \Re^{3\times 1}$, respectively. $M_p \in \Re^{6\times 3}$ and $M_a \in \Re^{6\times 3}$ are also stacked matrices of the motion constraint models $M_{pi} \in \Re^{3\times 3}$ and $M_{ai} \in \Re^{3\times 3}$, respectively.

2.16 Linear Solution for the Inverse Dynamics

Every equation from the beginning up to the last equation (2.110) is expressed in a linear form. Therefore, progressing from (2.110) to the linear implicit dynamic model (LImplDM) expressed in Theorem 1 of a parallel robot is just a matter of some simple linear algebraic manipulations, once the motorized joints are specified. In order to write this LImplDM, the following parameters and variables are required:

- $\boldsymbol{\xi}_{geo}$: constant geometric parameters of the robot (e.g., lengths, points).
- $\boldsymbol{\xi}_{dyn}$: constant dynamic parameters of the robot (e.g., masses, inertias, frictions).
- $\{\underline{\mathbf{x}}, d, \theta\}$: 0^{th} order variables of the kinematic elements. They allow us to write the static configuration of the robot, the motion constraint models and the kinematic coordinates.
- $\{\dot{\underline{\mathbf{x}}}, \dot{d}, \dot{\theta}\}$, $\{\ddot{\underline{\mathbf{x}}}, \ddot{d}, \ddot{\theta}\}$: 1^{st} and 2^{nd} order variables of the kinematic elements. They allow us to write the local forces and torques.
- $\boldsymbol{\Gamma}$: force vector of the robot's actuators (e.g., forces of active prismatic joints and torques of active revolute joints).

Corollary 1 *The inverse dynamics (IDM) of a parallel robot can then be obtained by solving a unique linear system of the LImplDM:*

$$A\,\boldsymbol{\Gamma} \; + \; \mathbf{b} \; = \; \mathbf{0}_{\,r\times 1} \tag{2.113}$$

where $A \in \Re^{r\times k}$ is the configuration-dependant matrix relating the unknown force vector $\boldsymbol{\Gamma} \in \Re^{k\times 1}$ of the actuators to the contributing efforts $\mathbf{b} \in \Re^{r\times 1}$ of the kinematic elements. The r is the dimension of a surjective motion basis ($r \geq k$). As long as the matrix A is full rank, one can solve for $\boldsymbol{\Gamma}$:

$$\boldsymbol{\Gamma} \; = \; -\,A^{\dagger}\,\mathbf{b} \tag{2.114}$$

where $A^{\dagger}$ is the pseudo-inverse of the matrix A to be computed with a QR or SVD decomposition for a fast and robust solution rather than literally with the Moore-Penrose formula.

Example: We solve for the inverse dynamic of the five-bar mechanism shown in Fig. 2.6. To do so, we first write explicitly the equation of the dynamic coordinate (the first one in (2.106)) which includes the motor torques:

$$\mathbb{F}_{\underline{x}_{pi}} \; = \; [\,\underline{\mathbf{x}}_{pi}\,]_{\times}^{T}\,\underline{\mathbf{z}}_{pi}\,\tau_{\underline{\mathbf{x}}_{pi}} \; + \; \widetilde{\mathbb{F}}_{\underline{x}_{pi}} \tag{2.115}$$

which can be rewritten as follows:

$$\mathbb{F}_{\underline{x}_{pi}} \;=\; \tau_{\underline{\mathbf{x}}_{pi}}\,\underline{\mathbf{y}}_{pi} \;+\; \widetilde{\mathbb{F}}_{\underline{x}_{pi}} \tag{2.116}$$

where

$$\widetilde{\mathbb{F}}_{\underline{x}_{pi}} \;=\; \frac{\ell_{pi}}{2}\,(\,\mathbf{f}_{\mathbf{g}(pi)} \;+\; \mathbf{f}^{*}_{pi}\,) \;+\; [\,\underline{\mathbf{x}}_{pi}\,]^{T}_{\times}\left(\,\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{pi}} \;+\; \boldsymbol{\tau}^{*}_{\underline{\mathbf{x}}_{pi}} \;+\; \boldsymbol{\tau}^{*}_{pi}\right) \;+\; \ell_{pi}\,(\,\mathbf{f}_{\mathbf{g}(ai)} \;+\; \mathbf{f}^{*}_{ai}\,) \tag{2.117}$$

Afterwards, we can rewrite the equations of motion (2.112) of the five-bar mechanism as below:

$$M_{p}^{T}\left(\left[\begin{array}{cc}\underline{\mathbf{y}}_{p1} & \mathbf{0}_{3\times1}\\ \mathbf{0}_{3\times1} & \underline{\mathbf{y}}_{p2}\end{array}\right]\left[\begin{array}{c}\tau_{\underline{\mathbf{x}}_{p1}}\\ \tau_{\underline{\mathbf{x}}_{p2}}\end{array}\right] + \left[\begin{array}{c}\widetilde{\mathbb{F}}_{\underline{x}_{p1}}\\ \widetilde{\mathbb{F}}_{\underline{x}_{p2}}\end{array}\right]\right) \;+\; M_{a}^{T}\,\mathbb{F}_{\underline{x}_{a}} \;=\; \mathbf{0}_{3\times1} \tag{2.118}$$

which can be reformulated in the form of (2.113):

$$A\left[\begin{array}{c}\tau_{\underline{\mathbf{x}}_{p1}}\\ \tau_{\underline{\mathbf{x}}_{p2}}\end{array}\right] \;+\; \mathbf{b} \;=\; \mathbf{0}_{\,3\times1} \tag{2.119}$$

where $A \in \Re^{3\times2}$ is as follows:

$$A \;=\; M_{p}^{T}\left[\begin{array}{cc}\underline{\mathbf{y}}_{p1} & \mathbf{0}_{3\times1}\\ \mathbf{0}_{3\times1} & \underline{\mathbf{y}}_{p2}\end{array}\right] \tag{2.120}$$

and where $\mathbf{b} \in \Re^{3\times1}$ is as below:

$$\mathbf{b} \;=\; M_{p}^{T}\left[\begin{array}{c}\widetilde{\mathbb{F}}_{\underline{x}_{p1}}\\ \widetilde{\mathbb{F}}_{\underline{x}_{p2}}\end{array}\right] \;+\; M_{a}^{T}\,\mathbb{F}_{\underline{x}_{a}} \tag{2.121}$$

Finally, the solution $\boldsymbol{\Gamma} \;=\; [\,\tau_{\underline{\mathbf{x}}_{p1}},\,\tau_{\underline{\mathbf{x}}_{p2}}\,]^{T}$ (torque vector of the motors) of inverse dynamics of the five-bar mechanism is computed as follows:

$$\left[\begin{array}{c}\tau_{\underline{\mathbf{x}}_{p1}}\\ \tau_{\underline{\mathbf{x}}_{p2}}\end{array}\right] \;=\; -\,A^{\dagger}\,\mathbf{b} \tag{2.122}$$

2.17 A Global View to the Proposed Methodology

In short, using the proposed methodology, one can write *efficiently* the inverse dynamic model of a parallel robot by simply following these 6 steps:

1. Decompose the parallel robot to its kinematic elements (by inspection);
2. Define the type of each of the « kinematic elements » (by inspection);
3. Compute the kinematic coordinates and the kinematic constraints (automatic);
4. List the local forces and torques on the kinematic elements (automatic);
5. Compute the dynamic coordinates and the dynamic constraints (automatic);

6. Solve linearly for the inverse dynamic model (automatic).

The state variables $\{\underline{\mathbf{x}},\, d,\, \theta\}$ can be directly measured with proprioceptive/exteroceptive sensors (e.g., motor encoders, camera, etc.), and/or be obtained through some mechanical kinematic models, if this does not lower the efficiency. Therefore, we generalize the representation of the inverse dynamic model, without concern for sensors and the models used, as follows:

$$\boldsymbol{\Gamma} \;=\; IDM(\,\ddot{\mathbf{s}},\, \dot{\mathbf{s}},\, \mathbf{s},\, \xi_{geo},\, \xi_{dyn}\,) \;=\; -\,A^{\dagger}(\mathbf{s})\,\mathbf{b}(\ddot{\mathbf{s}},\, \dot{\mathbf{s}},\, \mathbf{s}) \tag{2.123}$$

where $\mathbf{s}$ is the set of state variables of the kinematic elements:

$$\mathbf{s} \,:\, \{\underline{\mathbf{x}}_i,\, d_i,\, \theta_i\}\,, \qquad i = 1, \ldots, n_{ke} \tag{2.124}$$

2.17.1 Compared to/with Khalil's, Kane's and Tsai's Methods

- We were inspired by the idea of using passive joint coordinates with the active ones (redundancy) in modeling from Khalil, and we recommended a new redundant set of state variables which keeps the equations compact and linear. Furthermore, in this way we do not need to compute the global balancing force at the end-effector.
- We were inspired by the idea of to be free in our choice of a motion basis (minimal or redundant) from Kane, and we proposed a unique redundant motion basis which makes Kane's method easily applicable to broad range of robots (serial and parallel).
- We were inspired by the idea of writing final equations of motion easily using the local efforts done on each of the kinematic elements from Tsai, and we improved Tsai's formulation by Khalil and Kane's inspirational ideas such that it became geometrically more intuitive, simpler, completely linear and more practical.

2.17.2 Originality

Body-based modeling and control methodology rather than joint-based:

- We use lines to model the moving bodies (concrete) rather than the joint axes (abstract). This enhances the visual perception of a robot, such that a human brain can almost vividly imagine a robot's motion by just reading the equations (augmented reality).
- Equations of motion use simple linear vector algebra and are in compact form. This eases the codability and the fast solvability of equations on the computer. Even for the most complex robots, the inverse dynamic model can be worked out with pen and paper.

Practical applicability:

- Vision allows us to sense the moving bodies (directed 3D lines) in Cartesian space, which uniquely define the state of the robot. Thus, vision allows for a direct use of our method on the real robots.
- Vision and motor encoders together increase the quality (accuracy and richness) of the information for simple modeling and precise control.

2.18 Applied to the Quattro Parallel Robot

The proposed modeling methodology was applied to the Quattro, the Gough-Stewart, the Delta, the 3-RRR and the Orthoglide parallel robots. At the end of this chapter, the proposed

modeling methodology is shown in detail only for the Quattro parallel robot. For the rest of the parallel robots, the reader is referred to the Appendix A.

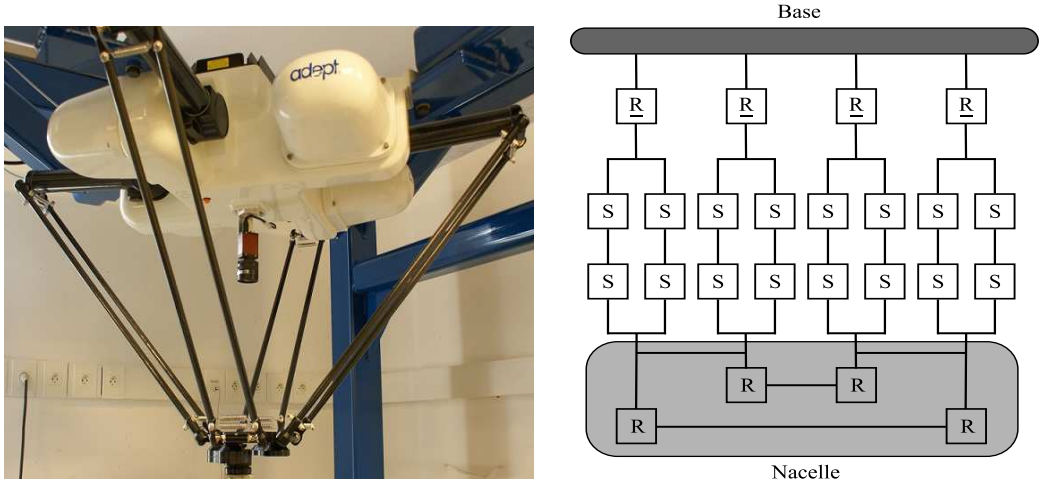


Figure 2.7 – The Quattro parallel robot with a base-mounted camera (*left*) and its joint-oriented graphical layout (*right*).

2.18.1 State Variables

The Quattro is composed of four identical kinematic legs which carry the articulated nacelle (see Fig. 2.7). Each of the 4 kinematic legs is actuated from the base by a revolute motor located at $\mathbf{P}_i$. A kinematic leg has two consecutive kinematic elements: an upper-leg $[\mathbf{P}_i\mathbf{A}_i]$ and a lower-leg $[\mathbf{A}_i\mathbf{B}_i]$. Lower-leg and upper-leg are attached to each other at $\mathbf{A}_i$. At the top, the upper-legs are connected to the motors, while at the bottom, the lower-legs are connected to the articulated nacelle. Also, the form of the lower-legs with attached nacelle is called the *umbrella* of the Quattro robot, which can be thought of as being an articulated object attached to the independent upper-legs. The articulated nacelle is designed with four kinematic elements [NRC$^+$05]: the two lateral kinematic elements (either $[\mathbf{B}_1\mathbf{B}_2]$, $[\mathbf{B}_3\mathbf{B}_4]$ or $[\mathbf{C}_1\mathbf{C}_2]$, $[\mathbf{C}_3\mathbf{C}_4]$) and the two central kinematic elements ($[\mathbf{C}_3\mathbf{C}_2]$, $[\mathbf{C}_4\mathbf{C}_1]$) linking lateral ones with revolute joints (see Fig. 2.8).

Kinematic Element Types: Hence, we have the following kinematic elements in the Quattro robot to inspect and to decide their types: upper-legs, lower-legs and the four parts of the nacelle. The configurations of those kinematic elements will be defined by the unit direction vectors of the bodies rather than the non-linear joint coordinates:

Upper-legs $[\mathbf{P}_i\mathbf{A}_i]$ An upper-leg rotates around a fixed axis. It has 1 dof rotational extrinsic mobility due to a motorized (R)evolute joint. It is a *bar type* kinematic element: $\underline{\mathbf{x}}_{pi}$.

Lower-legs $[\mathbf{A}_i\mathbf{B}_i]$ Each lower-leg consists of two slim and cylindrical shaped rods fitted with ball-joints (($\mathbf{A}_{i1}, \mathbf{A}_{i2}$) and ($\mathbf{B}_{i1}, \mathbf{B}_{i2}$)), forming a parallelogram. The joint-oriented notation of a kinematic leg, which contains this type of a lower-leg, is symbolically

denoted as $\underline{R}-(S-S)_2$. This $\underline{R}-(S-S)_2$ also equals $\underline{R}-U-U$ where $\underline{R}$ and S stand for an actuated revolute joint and a passive spherical joint, respectively. The $(S-S)_2$ architecture of a lower-leg fastened to an upper-leg ensures that the vectors $\overrightarrow{\mathbf{A}_{i1}\mathbf{A}_{i2}}$ and $\overrightarrow{\mathbf{B}_{i1}\mathbf{B}_{i2}}$ are always kept parallel to a fixed direction ($\underline{\mathbf{z}}_{pi}$). This motion constraint completely restricts the rotation of a lower-leg around its lengthwise direction ($\underline{\mathbf{x}}_{ai}$). Thus, this parallelogram lower-leg rotates only around a moving axis which means that it has 2 dof rotational extrinsic mobility due to the existing (S)pherical joints. Therefore, it is a *bar type* kinematic element: $\underline{\mathbf{x}}_{ai}$.

Nacelle $[\mathbf{C}_i\mathbf{C}_{i+1}]$ The articulated nacelle has two lateral and two central kinematic elements which are structurally restricted to be coplanar with respect to each other. The articulated nacelle itself has only a translational motion, but each of its lateral kinematic elements can be independently translated in one fixed direction ($\underline{\mathbf{x}}_b$) which lets the central ones have an additional relative rotational motion around an axis that is parallel to a fixed direction. Thus, the end-effector $\mathbf{E}$, which is located on the $[\mathbf{C}_2\mathbf{C}_3]$ central kinematic element (i.e., the moving platform), reaches 4 degrees of freedom, namely 3 for the translational movements and 1 for the rotational movement. The moving-platform also has an amplification system to transform the relative rotation θ into a proportional rotation ($\beta=\kappa\theta$) in the end-effector $\mathbf{E}$ (see Fig. 2.8). As a result of this structure of the nacelle, the lateral kinematic elements have only translational extrinsic mobility due to the parallelogram lower-legs, and the central kinematic elements have the same translational extrinsic mobility with a relative rotation allowed by the (R)evolute joints. Thus, each kinematic element is a *bar type*: $\underline{\mathbf{x}}_{bi}=\delta_i\,\underline{\mathbf{x}}_e$ where $\delta_1=\delta_3=0,\ \delta_2=-1,\ \delta_4=1$. If $i=4$ then $i+1\triangleq 1$.

The static state of the mechanism is therefore totally and redundantly defined by the unit vectors $\{\underline{\mathbf{x}}_{pi},\,\underline{\mathbf{x}}_{ai},\,\underline{\mathbf{x}}_{bi}\}$. Figure 2.9 shows the new body-oriented graphical layout of the Quattro parallel robot. In modeling, the following notation is used:

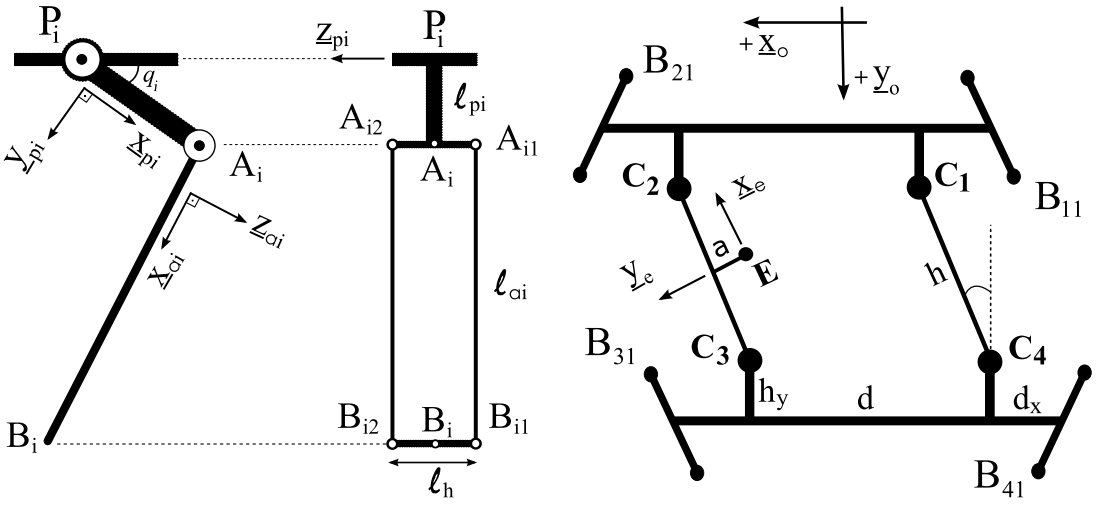


Figure 2.8 – Side and front views of a kinematic leg with its variables and parameters (*left*). The plan of the nacelle with its variables and parameters (*right*).

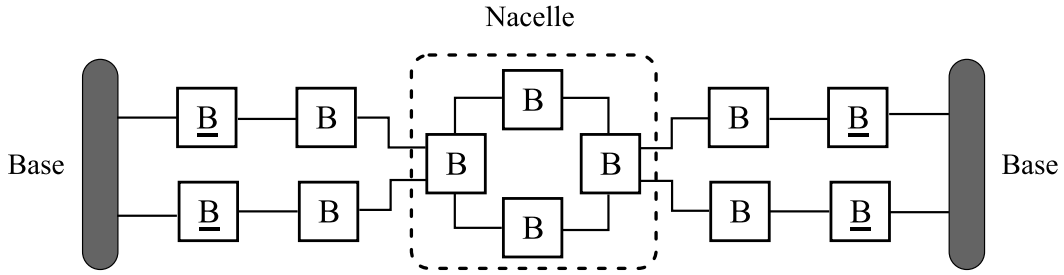


Figure 2.9 – Body-oriented graphical layout of the Quattro parallel robot.

- $i = 1, 2, 3, 4$ denotes the kinematic legs.
- $j = \{p, a, b\}$ is the literal representation of the kinematic elements in the mechanism.
- $\xi_{geo} = \{\mathbf{P}_i, \ell_{ji}, \ell_h, d, d_x, h, h_y, \mathsf{a}\}$ are the geometric parameters (e.g., constant lengths and points).
- $\xi_{dyn} = \{m_{ji}, \mathcal{I}_{ji}, f_{v_i}, f_{c_i}\}$ are the dynamic parameters (e.g., weights, inertias, frictions).
- $\mathcal{F}_o = (\mathbf{O}, \underline{\mathbf{x}}_o, \underline{\mathbf{y}}_o, \underline{\mathbf{z}}_o)$, $\mathcal{F}_e = (\mathbf{E}, \underline{\mathbf{x}}_e, \underline{\mathbf{y}}_e, \underline{\mathbf{z}}_e)$, $\mathcal{F}_{pi} = (\mathbf{P_i}, \underline{\mathbf{x}}_{pi}, \underline{\mathbf{y}}_{pi}, \underline{\mathbf{z}}_{pi})$ and $\mathcal{F}_{ai} = (\mathbf{A_i}, \underline{\mathbf{x}}_{ai}, \underline{\mathbf{y}}_{ai}, \underline{\mathbf{z}}_{ai})$ denote respectively the base, the end-effector, the i^{th} upper-leg and the i^{th} lower-leg frames.
- q_i is the articulated position of the i^{th} upper-leg motor.
- The end-effector pose ($\mathbb{X}$) is composed of the origin ($\mathbf{E}$) of the end-effector frame and the orientation of the $[\mathbf{C}_2\mathbf{C}_3]$ moving platform ($\underline{\mathbf{x}}_e$). The end-effector pose velocity is then $\dot{\mathbf{E}}$ and $\underline{\dot{\mathbf{x}}}_e$:

$$\mathbb{X} \triangleq \begin{bmatrix} \mathbf{E} \\ \underline{\mathbf{x}}_e \end{bmatrix}, \quad \dot{\mathbb{X}} \triangleq \begin{bmatrix} \dot{\mathbf{E}} \\ \underline{\dot{\mathbf{x}}}_e \end{bmatrix} \in \Re^{6\times 1}$$

2.18.2 Kinematics

Mass Centers: Figure 2.10 shows the mass centers $\mathbf{S}_{ji}$ of the kinematic elements in the mechanism. The expressions are fully in vector form as none of the kinematic elements have intrinsic mobility. Then, the mass centers of the basic parts of the Quattro robot are written as follows: (i) for the upper-legs,

$$\mathbf{S}_{pi} = \mathbf{P}_i + \frac{\ell_{pi}}{2}\underline{\mathbf{x}}_{pi} \tag{2.125}$$

(ii) for the lower-legs,

$$\mathbf{S}_{ai} = \mathbf{P}_i + \ell_{pi}\underline{\mathbf{x}}_{pi} + \frac{\ell_{ai}}{2}\underline{\mathbf{x}}_{ai} \tag{2.126}$$

(iii) for the nacelle's kinematic elements,

$$\mathbf{S}_{bi} = \mathbf{P}_i + \ell_{pi}\underline{\mathbf{x}}_{pi} + \ell_{ai}\underline{\mathbf{x}}_{ai} + \overrightarrow{\mathbf{B}_i\mathbf{S}_{bi}} \tag{2.127}$$

where $\overrightarrow{\mathbf{B}_1\mathbf{S}_{b1}}$ and $\overrightarrow{\mathbf{B}_3\mathbf{S}_{b3}}$ are constant, since they are located on the translational lateral kinematic elements of the nacelle. The $\overrightarrow{\mathbf{B}_2\mathbf{S}_{b2}}$ and $\overrightarrow{\mathbf{B}_4\mathbf{S}_{b4}}$ are the sum of a constant vector and a component along the rotating central kinematic elements of the nacelle:

$$\overrightarrow{\mathbf{B}_i\mathbf{S}_{bi}} = const_i + \frac{h}{2}\underline{\mathbf{x}}_{bi} \tag{2.128}$$

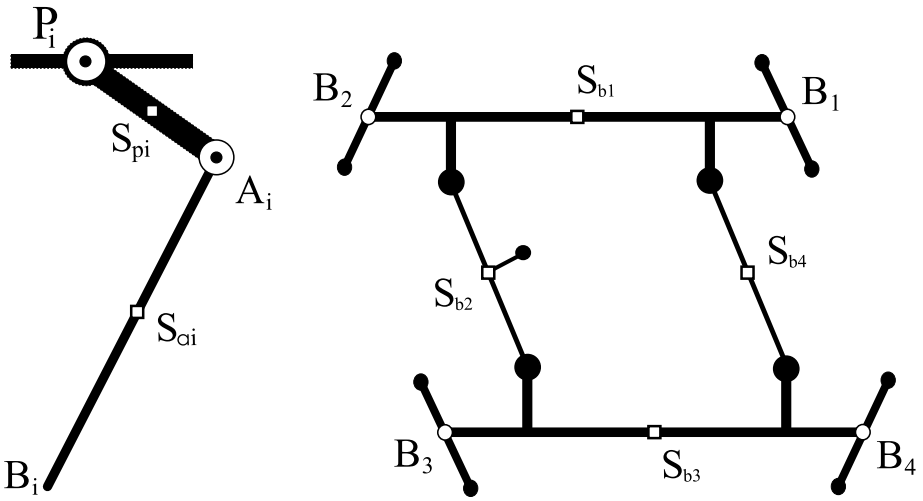


Figure 2.10 – The mass centers $\mathbf{S}_{ji}$ (white squares) of the kinematic elements. All of the kinematic elements are assumed to be homogenous and symmetric.

Velocities: The translational velocities are derived by simply differentiating (2.125), (2.126) and (2.127). The rotational velocities are compactly represented through the unit orientation vectors. Then, the velocities are written as follows: (i) for the upper-legs,

$$\dot{\mathbf{S}}_{pi} \;=\; \frac{\ell_{pi}}{2}\,\dot{\underline{\mathbf{x}}}_{pi}\,, \qquad\qquad \boldsymbol{\omega}_{pi} \;\triangleq\; \underline{\mathbf{x}}_{pi} \times \dot{\underline{\mathbf{x}}}_{pi} \tag{2.129}$$

(ii) for the lower-legs,

$$\dot{\mathbf{S}}_{ai} \;=\; \ell_{pi}\,\dot{\underline{\mathbf{x}}}_{pi} + \frac{\ell_{ai}}{2}\,\dot{\underline{\mathbf{x}}}_{ai}\,, \qquad\qquad \boldsymbol{\omega}_{ai} \;\triangleq\; \underline{\mathbf{x}}_{ai} \times \dot{\underline{\mathbf{x}}}_{ai} \tag{2.130}$$

(iii) for the nacelle's kinematic elements,

$$\dot{\mathbf{S}}_{bi} \;=\; \ell_{pi}\,\dot{\underline{\mathbf{x}}}_{pi} + \ell_{ai}\,\dot{\underline{\mathbf{x}}}_{ai} + \frac{h}{2}\,\dot{\underline{\mathbf{x}}}_{bi}\,, \qquad \boldsymbol{\omega}_{bi} \;\triangleq\; \underline{\mathbf{x}}_{bi} \times \dot{\underline{\mathbf{x}}}_{bi} \tag{2.131}$$

Accelerations: The accelerations are obtained by differentiating (2.129), (2.130) and (2.131) with respect to time. Then, the accelerations are written as follows: (i) for the upper-legs,

$$\ddot{\mathbf{S}}_{pi} \;=\; \frac{\ell_{pi}}{2}\,\ddot{\underline{\mathbf{x}}}_{pi}\,, \qquad\qquad \dot{\boldsymbol{\omega}}_{pi} \;\triangleq\; \underline{\mathbf{x}}_{pi} \times \ddot{\underline{\mathbf{x}}}_{pi} \tag{2.132}$$

(ii) for the lower-legs,

$$\ddot{\mathbf{S}}_{ai} \;=\; \ell_{pi}\,\ddot{\underline{\mathbf{x}}}_{pi} + \frac{\ell_{ai}}{2}\,\ddot{\underline{\mathbf{x}}}_{ai}\,, \qquad\qquad \dot{\boldsymbol{\omega}}_{ai} \;\triangleq\; \underline{\mathbf{x}}_{ai} \times \ddot{\underline{\mathbf{x}}}_{ai} \tag{2.133}$$

(iii) for the nacelle's kinematic elements,

$$\ddot{\mathbf{S}}_{bi} \;=\; \ell_{pi}\,\ddot{\underline{\mathbf{x}}}_{pi} + \ell_{ai}\,\ddot{\underline{\mathbf{x}}}_{ai} + \frac{h}{2}\,\ddot{\underline{\mathbf{x}}}_{bi}\,, \qquad \dot{\boldsymbol{\omega}}_{bi} \;\triangleq\; \underline{\mathbf{x}}_{bi} \times \ddot{\underline{\mathbf{x}}}_{bi} \tag{2.134}$$

2.18.3 Kinematic Constraints

The closed-loop constraint equation for each of the kinematic legs can be written as follows:

$$\overrightarrow{\mathbf{OE}} + (\mathsf{a}\,\underline{\mathbf{y}}_e + \varepsilon_i \frac{h}{2}\underline{\mathbf{x}}_e + \beta_i \overrightarrow{\mathbf{C}_2\mathbf{C}_1} + \gamma_i \overrightarrow{\mathbf{C}_3\mathbf{C}_4} + \overrightarrow{\mathbf{C}_i\mathbf{B}_i}) - \ell_{ai}\,\underline{\mathbf{x}}_{ai} - \ell_{pi}\,\underline{\mathbf{x}}_{pi} - \overrightarrow{\mathbf{OP}_i} = \mathbf{0} \tag{2.135}$$

where $\mathbf{O}$, $\mathbf{P}_i$, $\overrightarrow{\mathbf{C}_2\mathbf{C}_1}$, $\overrightarrow{\mathbf{C}_3\mathbf{C}_4}$ and $\overrightarrow{\mathbf{C}_i\mathbf{B}_i}$ are constants, and where ε_i, β_i and γ_i are as follows:

$$\varepsilon_1 = \varepsilon_2 = 1\,,\ \ \varepsilon_3 = \varepsilon_4 = -1\,,\quad \beta_1 = 1\,,\ \ \beta_2 = \beta_3 = \beta_4 = 0\,,\quad \gamma_1 = \gamma_2 = \gamma_3 = 0\,,\ \ \gamma_4 = 1 \tag{2.136}$$

Afterwards, one can differentiate (2.135) with respect to time in order to obtain the motion constraint equation, which yields:

$$\dot{\mathbf{E}} + (\mathsf{a}\,\dot{\underline{\mathbf{y}}}_e + \varepsilon_i \frac{h}{2}\dot{\underline{\mathbf{x}}}_e) - \ell_{ai}\,\dot{\underline{\mathbf{x}}}_{ai} - \ell_{pi}\,\dot{\underline{\mathbf{x}}}_{pi} = \mathbf{0} \tag{2.137}$$

The motion constraints for the attachment points $\mathbf{B}_i$ of the nacelle can be written from (2.137) as below:

$$\dot{\mathbf{E}} + (\mathsf{a}\,\dot{\underline{\mathbf{y}}}_e + \varepsilon_i \frac{h}{2}\dot{\underline{\mathbf{x}}}_e) = \dot{\mathbf{B}}_i \tag{2.138}$$

where

$$\dot{\underline{\mathbf{y}}}_e = \boldsymbol{\omega}_e \times \underline{\mathbf{y}}_e = (\underline{\mathbf{x}}_e \times \dot{\underline{\mathbf{x}}}_e) \times \underline{\mathbf{y}}_e = [\,\underline{\mathbf{z}}_e\,]_\times \dot{\underline{\mathbf{x}}}_e \tag{2.139}$$

Then, (2.138) becomes:

$$\dot{\mathbf{E}} + (\mathsf{a}\,[\,\underline{\mathbf{z}}_e\,]_\times + \varepsilon_i \frac{h}{2} I_3)\,\dot{\underline{\mathbf{x}}}_e = \dot{\mathbf{B}}_i \tag{2.140}$$

which can be rewritten as follows:

$$\dot{\mathbf{B}}_i = L_{\mathbf{B}_i}\,\dot{\mathbb{X}}\,, \qquad L_{\mathbf{B}_i} = \left[\ I_3 \quad (\mathsf{a}\,[\,\underline{\mathbf{z}}_e\,]_\times + \varepsilon_i \tfrac{h}{2} I_3)\ \right] \tag{2.141}$$

where $L_{\mathbf{B}_i} \in \Re^{3\times 6}$ is the relation between the Cartesian velocity of the terminal point of the i^{th} kinematic leg and the end-effector pose velocity $\dot{\mathbb{X}} \in \Re^{6\times 1}$. Finally, we reformulate (2.137) to ease the derivation of kinematic relations as below:

$$L_{\mathbf{B}_i}\,\dot{\mathbb{X}}_e - \ell_{ai}\,\dot{\underline{\mathbf{x}}}_{ai} - \ell_{pi}\,\dot{\underline{\mathbf{x}}}_{pi} = \mathbf{0} \tag{2.142}$$

Constraints on the Active Upper-Legs: The kinematic constraint on each unit orientation vector of the active upper-legs is defined from (2.142) by exploiting the following relations:

$$\underline{\mathbf{x}}_{ai}^T\,\dot{\underline{\mathbf{x}}}_{ai} = 0\,, \qquad \underline{\mathbf{y}}_{pi}\,//\,\dot{\underline{\mathbf{x}}}_{pi} \tag{2.143}$$

By multiplying (2.142) with $\underline{\mathbf{x}}_{ai}^T$ and then with $\underline{\mathbf{y}}_{pi}$, one finds that:

$$\ell_{pi}\,\underline{\mathbf{y}}_{pi}\,(\underline{\mathbf{x}}_{ai}^T\,\dot{\underline{\mathbf{x}}}_{pi}) = \underline{\mathbf{y}}_{pi}\,\underline{\mathbf{x}}_{ai}^T\,L_{\mathbf{B}_i}\,\dot{\mathbb{X}} \tag{2.144}$$

where the left side of (2.144) can be rewritten from the parallelism of the vectors as follows:

$$\ell_{pi}\,(\,\underline{\mathbf{x}}_{ai}^{T}\,\underline{\mathbf{y}}_{pi})\,\dot{\underline{\mathbf{x}}}_{pi}\;=\;\underline{\mathbf{y}}_{pi}\,\underline{\mathbf{x}}_{ai}^{T}\,L_{\mathbf{B}_i}\;\dot{\mathbb{X}} \tag{2.145}$$

then, one can obtain the inverse differential kinematic model for an upper-leg orientation unit vector as below:

$$\dot{\underline{\mathbf{x}}}_{pi}\;=\;M_{pi}\;\dot{\mathbb{X}}\;=\;\frac{\underline{\mathbf{y}}_{pi}\,\underline{\mathbf{x}}_{ai}^{T}}{\ell_{pi}\,\underline{\mathbf{x}}_{ai}^{T}\,\underline{\mathbf{y}}_{pi}}\;L_{\mathbf{B}_i}\;\dot{\mathbb{X}} \tag{2.146}$$

where $M_{pi} \in \Re^{3\times 6}$. The complete $M_p \in \Re^{12\times 6}$ will be noted as below:

$$M_p\;=\;\left[\begin{array}{c} M_{p1} \\ \vdots \\ M_{p4} \end{array}\right] \tag{2.147}$$

Constraints on the Passive Lower-Legs: The kinematic constraint on the unit orientation vector of each passive lower-legs can be furnished by substituting (2.146) into (2.142) as below:

$$\dot{\underline{\mathbf{x}}}_{ai}\;=\;M_{ai}\;\dot{\mathbb{X}}\;=\;\frac{1}{\ell_{ai}}\,\Big(\,L_{\mathbf{B}_i}\;-\;\ell_{pi}\;M_{pi}\Big)\;\dot{\mathbb{X}} \tag{2.148}$$

where $M_{ai} \in \Re^{3\times 6}$ is the inverse differential kinematic model associated between a lower-leg unit orientation vector and the pose. The full $M_a \in \Re^{12\times 6}$ can be written in the following stacked form:

$$M_a\;=\;\left[\begin{array}{c} M_{a1} \\ \vdots \\ M_{a4} \end{array}\right] \tag{2.149}$$

Constraints on the Passive Nacelle: The kinematic constraint on a orientation unit vector of the passive nacelle is defined with the following relation:

$$\dot{\underline{\mathbf{x}}}_{bi}\;=\;M_{bi}\;\dot{\mathbb{X}}=\delta_i\left[\begin{array}{cc}\mathbf{0}_3 & I_3\end{array}\right]\;\dot{\mathbb{X}} \tag{2.150}$$

where $M_{bi} \in \Re^{3\times 6}$. Due to parallelograms, the nacelle stays parallel to its initial plane and thus, the end-effector has a rotation axis $\underline{\mathbf{z}}_e$ parallel to a fixed direction. Then, the complete $M_b \in \Re^{12\times 6}$ will be defined as follows:

$$M_b\;=\;\left[\begin{array}{c} M_{b1} \\ \vdots \\ M_{b4} \end{array}\right] \tag{2.151}$$

Since $\underline{\mathbf{x}}_e = \underline{\mathbf{x}}_{b4}$, then $M_e = M_{b4}$ and one can write:

$$\dot{\underline{\mathbf{x}}}_e\;=\;M_e\,\dot{\mathbb{X}}\;=\;M_{b4}\;\dot{\mathbb{X}}\;=\;\left[\begin{array}{cc}\mathbf{0}_3 & I_3\end{array}\right]\;\dot{\mathbb{X}} \tag{2.152}$$

2.18.4 Kinematic Coordinates

The instantaneous configuration of the Quattro is expressed with the following redundant set of 12 generalized vectors (orientation unit vectors of the kinematic elements):

$$\{\, \underline{\mathbf{x}}_{pi},\, \underline{\mathbf{x}}_{ai},\, \underline{\mathbf{x}}_{bi} \,\}\,, \quad i = 1, \ldots, 4 \tag{2.153}$$

and the motion of the Quattro is expressed through the motion basis which is directly defined as the time derivatives of the 12 generalized vectors:

$$\mathbf{u}_{x_{ji}} \in \{\, \dot{\underline{\mathbf{x}}}_{ji} \,|\, j \in \{p, a, b\}\,,\; i \in \{1, 2, 3, 4\} \,\} \tag{2.154}$$

Note that the nacelle is shared by each of the kinematic legs by extending them with a corresponding kinematic element of the nacelle. Then, from now on, a kinematic leg is composed of an upper-leg, a lower-leg and a corresponding kinematic element of the nacelle ($[\mathbf{C}_i\, \mathbf{C}_{i+1}]$, if $i = 4$, then $(i+1) \triangleq 1$).

For each kinematic element in the mechanism, the translational and rotational kinematic coordinates are tabulated in the Table 2.3 by computing the partial derivatives of (2.129)-(2.131) with respect to the motion basis.

Table 2.3 – The (transposed) kinematic coordinates of the Quattro parallel robot, i=1,2,3,4.

	$\partial\, \dot{\mathbf{S}}_{pi}$	$\partial\, \boldsymbol{\omega}_{pi}$	$\partial\, \dot{\mathbf{S}}_{ai}$	$\partial\, \boldsymbol{\omega}_{ai}$	$\partial\, \dot{\mathbf{S}}_{bi}$	$\partial\, \boldsymbol{\omega}_{bi}$
$\partial\, \dot{\underline{\mathbf{x}}}_{pi}$	$\frac{1}{2}\, \ell_{pi}\, I_3$	$[\underline{\mathbf{x}}_{pi}]^T_{\times}$	$\ell_{pi}\, I_3$	$\mathbf{0}$	$\ell_{pi}\, I_3$	$\mathbf{0}$
$\partial\, \dot{\underline{\mathbf{x}}}_{ai}$	$\mathbf{0}$	$\mathbf{0}$	$\frac{1}{2}\, \ell_{ai}\, I_3$	$[\underline{\mathbf{x}}_{ai}]^T_{\times}$	$\ell_{ai}\, I_3$	$\mathbf{0}$
$\partial\, \dot{\underline{\mathbf{x}}}_{bi}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\delta_i^2\, \frac{1}{2}\, h\, I_3$	$[\underline{\mathbf{x}}_{bi}]^T_{\times}$

2.18.5 Dynamic Coordinates

2.18.5.1 Listing Active and Reactive Forces

There is a set of active forces/torques acting on the kinematic elements of the Quattro parallel robot due to actuator torques and forces of gravity.

– *Actuator Torques:* We write the actuator torque vectors ($\boldsymbol{\tau}_{\underline{\mathbf{x}}}$) of the active kinematic elements as follows:

$$\boldsymbol{\tau}_{\underline{\mathbf{x}}_{pi}} \;=\; \tau_{\underline{\mathbf{x}}_{pi}}\, \underline{\mathbf{z}}_{pi} \tag{2.155}$$

where $\tau_{\underline{\mathbf{x}}_{pi}}$ is the active torque of a rotary actuator whose rotation axis is $\underline{\mathbf{z}}_{pi}$ and whose rotating kinematic element is oriented along $\underline{\mathbf{x}}_{pi}$.

– *Forces of Gravity:* Afterwards, the active forces of gravity ($\mathbf{f_g}$), which is assumed to act at the center of masses of the kinematic elements, are given as below:

$$\mathbf{f}_{\mathbf{g}(pi)} \;=\; m_{pi}\, \mathbf{g}\,, \qquad \mathbf{f}_{\mathbf{g}(ai)} \;=\; m_{ai}\, \mathbf{g}\,, \qquad \mathbf{f}_{\mathbf{g}(bi)} \;=\; m_{bi}\, \mathbf{g} \tag{2.156}$$

where m_{ji} is the mass of the ji^{th} kinematic element and $\mathbf{g}$ is the gravity acceleration vector oriented towards the center of the Earth.

As a consequence of the accelerated matters in the Quattro parallel robot, reactive inertial and frictional forces/torques will appear at the actuators and at the kinematic elements.

– *Actuator Inertial Torques:* The rotary actuator inertial torques $\boldsymbol{\tau}^*_{\underline{\mathbf{x}}}$ can be written as follows:

$$\boldsymbol{\tau}^*_{\underline{\mathbf{x}}_{pi}} \;=\; -\,\mathcal{I}_{pi}\,\dot{\boldsymbol{\omega}}_{\,pi} \;=\; -\,\mathcal{I}_{pi}\,(\underline{\mathbf{x}}_{pi} \;\times\; \ddot{\underline{\mathbf{x}}}_{pi}) \tag{2.157}$$

where $\mathcal{I}_{pi}$ is the rotary motion inertia of the actuator around the $\underline{\mathbf{z}}_{pi}$ axis and $\dot{\boldsymbol{\omega}}_{\,pi}$ is the rotational acceleration vector of the kinematic element ($\underline{\mathbf{x}}_{pi}$) which is firmly fastened to the actuator.

– *Kinematic Element Body Inertial Forces and Torques:* The accelerated masses and inertias of the kinematic elements produce the set of inertial forces and torques $\{\mathbf{f}^*_{ji}\,,\boldsymbol{\tau}^*_{ji}\}$. These inertial forces and torques can be calculated using the Newton-Euler equations:

$$\mathbf{f}^*_{pi} \;=\; -\,m_{pi}\,\ddot{\mathbf{S}}_{pi}\,, \qquad \boldsymbol{\tau}^*_{pi} \;=\; -\,\mathcal{I}^T_{pi}\,\dot{\boldsymbol{\omega}}_{\,pi} \;-\; \boldsymbol{\omega}_{\,pi} \;\times\; (\mathcal{I}^T_{pi}\,\boldsymbol{\omega}_{\,pi}) \tag{2.158}$$

$$\mathbf{f}^*_{ai} \;=\; -\,m_{ai}\,\ddot{\mathbf{S}}_{ai}\,, \qquad \boldsymbol{\tau}^*_{ai} \;=\; -\,\mathcal{I}^T_{ai}\,\dot{\boldsymbol{\omega}}_{\,ai} \;-\; \boldsymbol{\omega}_{\,ai} \;\times\; (\mathcal{I}^T_{ai}\,\boldsymbol{\omega}_{\,ai}) \tag{2.159}$$

$$\mathbf{f}^*_{bi} \;=\; -\,m_{bi}\,\ddot{\mathbf{S}}_{bi}\,, \qquad \boldsymbol{\tau}^*_{bi} \;=\; -\,\mathcal{I}^T_{bi}\,\dot{\boldsymbol{\omega}}_{\,bi} \;-\; \boldsymbol{\omega}_{\,bi} \;\times\; (\mathcal{I}^T_{bi}\,\boldsymbol{\omega}_{\,bi}) \tag{2.160}$$

where $m_{ji}, \ddot{\mathbf{S}}_{ji}, \mathcal{I}_{ji}$ and $\boldsymbol{\omega}_{\,ji}$ are the mass, the translational acceleration vector of the mass center, the central inertia dyadic and the rotational velocity vector of the ji^{th} kinematic element, respectively.

– *Frictional Torques:* Frictional torques ($\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{ji}}$) will appear at the joint locations of the kinematic elements due to their relative motions among themselves. The frictional torques of the actuators can be computed as follows:

$$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{pi}} \;=\; -\,\bar{\tau}_{v(\underline{\mathbf{x}}_{pi})}\,\boldsymbol{\omega}_{pi} \;-\; \bar{\tau}_{c(\underline{\mathbf{x}}_{pi})}\,sign(\,\boldsymbol{\omega}^T_{pi}\,\underline{\mathbf{z}}_{pi}\,)\,\underline{\mathbf{z}}_{pi} \tag{2.161}$$

The frictional torques of the passive joints can be computed as below:

$$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{ai}} \;=\; -\,\bar{\tau}_{v(\underline{\mathbf{x}}_{ai})}\;(\,\boldsymbol{\omega}_{ai} \;-\; \boldsymbol{\omega}_{pi}\,) \;-\; \bar{\tau}_{c(\underline{\mathbf{x}}_{ai})}\,sign(\,(\,\boldsymbol{\omega}_{ai} \;-\; \boldsymbol{\omega}_{pi}\,)^T\;\underline{\mathbf{z}}_{ai}\,)\,\underline{\mathbf{z}}_{ai} \tag{2.162}$$

$$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{bi}} \;=\; -\,\bar{\tau}_{v(\underline{\mathbf{x}}_{bi})}\;(\,\boldsymbol{\omega}_{bi} \;-\; \boldsymbol{\omega}_{ai}\,) \;-\; \bar{\tau}_{c(\underline{\mathbf{x}}_{bi})}\,sign(\,(\,\boldsymbol{\omega}_{bi} \;-\; \boldsymbol{\omega}_{ai}\,)^T\;\underline{\mathbf{z}}_{bi}\,)\,\underline{\mathbf{z}}_{bi} \tag{2.163}$$

where $\bar{\tau}_{v(\underline{\mathbf{x}}_{ji})}$ and $\bar{\tau}_{c(\underline{\mathbf{x}}_{ji})}$ are the viscous and Coulomb friction coefficients of the joint of ji^{th} kinematic element, and where $\boldsymbol{\omega}_{ji}$ and $\underline{\mathbf{z}}_{ji}$ are the rotational velocity vector and the axis of rotation of the kinematic element ($\underline{\mathbf{x}}_{ji}$), respectively.

Hence, one can list all these local forces and torques as in Table 2.4. Figure 2.11 shows the local forces and torques which act on one of the identical kinematic leg of the Quattro parallel robot.

Computing Dynamic Coordinates: In order to eliminate the non-contributing forces, the dynamic coordinates are computed through the matrix-wise multiplication of the Tables 2.3 (transposed kinematic coordinates) and 2.4 (sum of local forces and torques).

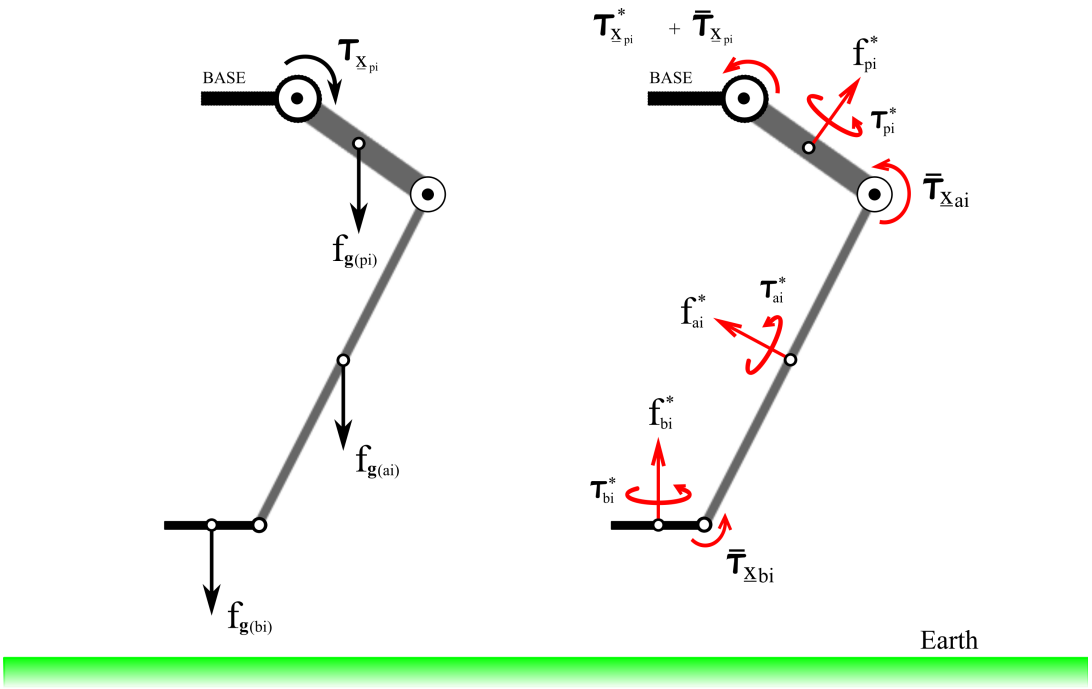


Figure 2.11 – Local forces and torques which act on one of the identical kinematic legs. Left figure shows the active forces and torques of a kinematic leg. Right figure shows the reactive inertial and frictional forces/torques which balance the active forces/torques.

Table 2.4 – The local forces and torques of the Quattro parallel robot, i=1,2,3,4.

	Active		*Friction*		*Inertia**	
	Actuator	*Gravity*	*Actuator*	*Passive Joint*	*Actuator*	*Element*
Forces (pi)	$\mathbf{0}$	$\mathbf{f}_{\mathbf{g}(pi)}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{f}^*_{pi}$
Torques (pi)	$\tau_{\underline{\mathbf{x}}_{pi}}\,\underline{\mathbf{z}}_{pi}$	$\mathbf{0}$	$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{pi}}$	$\mathbf{0}$	$\boldsymbol{\tau}^*_{\underline{\mathbf{x}}_{pi}}$	$\boldsymbol{\tau}^*_{pi}$
Forces (ai)	$\mathbf{0}$	$\mathbf{f}_{\mathbf{g}(ai)}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{f}^*_{ai}$
Torques (ai)	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{ai}}$	$\mathbf{0}$	$\boldsymbol{\tau}^*_{ai}$
Forces (bi)	$\mathbf{0}$	$\mathbf{f}_{\mathbf{g}(bi)}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{f}^*_{bi}$
Torques (bi)	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{bi}}$	$\mathbf{0}$	$\boldsymbol{\tau}^*_{bi}$

$$\begin{bmatrix} \mathbb{F}_{\underline{x}_{pi}} \\ \mathbb{F}_{\underline{x}_{ai}} \\ \mathbb{F}_{\underline{x}_{bi}} \end{bmatrix} = \begin{bmatrix} Kinematic \\ Coordinates \\ Table\ 2.3 \end{bmatrix}_{(3\times 6)} \begin{bmatrix} Sum\ of \\ Forces \\ Torques \\ Table\ 2.4 \end{bmatrix}_{(6\times 1)} \tag{2.164}$$

which can be explicitly written as follows:

$$\begin{bmatrix} \mathbb{F}_{\underline{x}_{pi}} \\ \mathbb{F}_{\underline{x}_{ai}} \\ \mathbb{F}_{\underline{x}_{bi}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\,\ell_{pi}\,I_3 & [\underline{\mathbf{x}}_{pi}]_{\times}^{T} & \ell_{pi}\,I_3 & \mathbf{0} & \ell_{pi}\,I_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{1}{2}\,\ell_{ai}\,I_3 & [\underline{\mathbf{x}}_{ai}]_{\times}^{T} & \ell_{ai}\,I_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \delta_i^2\,\frac{1}{2}\,h\,I_3 & [\underline{\mathbf{x}}_{bi}]_{\times}^{T} \end{bmatrix} \begin{bmatrix} \mathbf{f}_{\mathbf{g}(pi)} + \mathbf{f}_{pi}^{*} \\ \tau_{\underline{\mathbf{x}}_{pi}}\,\underline{\mathbf{z}}_{pi} + \widetilde{\boldsymbol{\tau}}_{pi} \\ \mathbf{f}_{\mathbf{g}(ai)} + \mathbf{f}_{ai}^{*} \\ \bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{ai}} + \boldsymbol{\tau}_{ai}^{*} \\ \mathbf{f}_{\mathbf{g}(bi)} + \mathbf{f}_{bi}^{*} \\ \bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{bi}} + \boldsymbol{\tau}_{bi}^{*} \end{bmatrix} \tag{2.165}$$

where

$$\widetilde{\boldsymbol{\tau}}_{pi} = \bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{pi}} + \boldsymbol{\tau}_{\underline{\mathbf{x}}_{pi}}^{*} + \boldsymbol{\tau}_{pi}^{*} \tag{2.166}$$

2.18.6 Dynamic Constraints

Exploiting (2.110), the dynamic constraints of the Quattro robot are written as follows:

$$M_p^T\,\mathbb{F}_p + M_a^T\,\mathbb{F}_a + M_b^T\,\mathbb{F}_b = \mathbf{0}_{6\times 1} \tag{2.167}$$

where $\mathbb{F}_p \in \Re^{12\times 1}$, $\mathbb{F}_a \in \Re^{12\times 1}$ and $\mathbb{F}_b \in \Re^{12\times 1}$ are the stacked vectors of the dynamic coordinates:

$$\mathbb{F}_p = \begin{bmatrix} \mathbb{F}_{\underline{x}_{p1}} \\ \vdots \\ \mathbb{F}_{\underline{x}_{p4}} \end{bmatrix}, \quad \mathbb{F}_a = \begin{bmatrix} \mathbb{F}_{\underline{x}_{a1}} \\ \vdots \\ \mathbb{F}_{\underline{x}_{a4}} \end{bmatrix}, \quad \mathbb{F}_b = \begin{bmatrix} \mathbb{F}_{\underline{x}_{b1}} \\ \vdots \\ \mathbb{F}_{\underline{x}_{b4}} \end{bmatrix} \tag{2.168}$$

and where $M_j \in \Re^{12\times 6}$ is as in (2.147), (2.149) and (2.151) for $j \in \{\,p,\,a,\,b\,\}$.

2.18.7 Inverse Dynamics

In order to extract the motor torques, we explicitly write the dynamic coordinate $\mathbb{F}_{\underline{x}_{pi}}$ from (2.165) which contains these torques $\tau_{\underline{\mathbf{x}}_{pi}}$:

$$\mathbb{F}_{\underline{x}_{pi}} = \tau_{\underline{\mathbf{x}}_{pi}}\,\underline{\mathbf{y}}_{pi} + \widetilde{\mathbb{F}}_{\underline{x}_{pi}} \tag{2.169}$$

where

$$\widetilde{\mathbb{F}}_{\underline{x}_{pi}} = \frac{\ell_{pi}}{2}\,(\mathbf{f}_{\mathbf{g}(pi)} + \mathbf{f}_{pi}^{*}) + [\underline{\mathbf{x}}_{pi}]_{\times}^{T}\,\widetilde{\boldsymbol{\tau}}_{pi} + \ell_{pi}\,(\mathbf{f}_{\mathbf{g}(ai)} + \mathbf{f}_{ai}^{*}) + \ell_{pi}\,(\mathbf{f}_{\mathbf{g}(bi)} + \mathbf{f}_{bi}^{*}) \tag{2.170}$$

Consequently the stacked $\mathbb{F}_p$ can be noted as follows:

$$\mathbb{F}_p = Y_p\,\boldsymbol{\Gamma} + \widetilde{\mathbb{F}}_p \tag{2.171}$$

where $Y_p \in \Re^{12\times 4}$, $\mathbf{\Gamma} \in \Re^{4\times 1}$ and $\widetilde{\mathbb{F}}_p \in \Re^{12\times 1}$ are as below:

$$Y_p = \begin{bmatrix} \underline{\mathbf{y}}_{p1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \underline{\mathbf{y}}_{p2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \underline{\mathbf{y}}_{p3} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \underline{\mathbf{y}}_{p4} \end{bmatrix}, \quad \mathbf{\Gamma} = \begin{bmatrix} \tau_{\underline{\mathbf{x}}_{p1}} \\ \tau_{\underline{\mathbf{x}}_{p2}} \\ \tau_{\underline{\mathbf{x}}_{p3}} \\ \tau_{\underline{\mathbf{x}}_{p4}} \end{bmatrix}, \quad \widetilde{\mathbb{F}}_p = \begin{bmatrix} \widetilde{\mathbb{F}}_{\underline{x}_{p1}} \\ \widetilde{\mathbb{F}}_{\underline{x}_{p2}} \\ \widetilde{\mathbb{F}}_{\underline{x}_{p3}} \\ \widetilde{\mathbb{F}}_{\underline{x}_{p4}} \end{bmatrix} \tag{2.172}$$

Afterwards, replacing (2.171) in (2.167), we can write (2.167) as follows:

$$A\,\mathbf{\Gamma} + \mathbf{b} = \mathbf{0} \tag{2.173}$$

where $A \in \Re^{6\times 4}$ and $\mathbf{b} \in \Re^{6\times 1}$ are defined as below:

$$A = M_p^T\, Y_p\,, \quad \mathbf{b} = M_p^T\, \widetilde{\mathbb{F}}_p + M_a^T\, \mathbb{F}_a + M_b^T\, \mathbb{F}_b \tag{2.174}$$

One can then write the solution of (2.173) as follows:

$$\mathbf{\Gamma} = -\,A^{\dagger}(\,\underline{\mathbf{x}}\,)\;\mathbf{b}\,(\,\underline{\ddot{\mathbf{x}}},\,\underline{\dot{\mathbf{x}}},\,\underline{\mathbf{x}}\,) \tag{2.175}$$

Consequently, the inverse dynamic model of such a complex Quattro robot has been compactly expressed by means of the orientation unit vectors.

$$\mathbf{\Gamma} = IDM(\,\underline{\ddot{\mathbf{x}}},\,\underline{\dot{\mathbf{x}}},\,\underline{\mathbf{x}}\,) \tag{2.176}$$

Note that the longest part of the calculus is the kinematics (made very easy by using the vector form), then the listing of the forces and torques. If all these are given, the dynamics are obtained through one single formula (2.167) and its linear solution.

2.19 Conclusions

This chapter outlined a framework for control-oriented dynamic modeling of parallel robots which uses the leg orientations. This modeling approach is easily applied to wide range of parallel robots. Written equations use only addition (+) and multiplication (*) operators. Neither trigonometric nor exponential functions are used. There is only the piecewise-linear *signum* function in the Coulomb friction which needs a simple sign check for its implementation. The equations are simple and compact even for complex robots. This is achieved with the 3D unit direction vectors of the leg orientations. Thus, we accomplished the first objective of the integrated dynamic MICMAC which is stated at the end of the Chapter 1.

This modeling approach is linear on the condition that these unit direction vectors, $\underline{\mathbf{x}}$, of the leg orientations and their velocities, $\dot{\underline{\mathbf{x}}}$, are given. As stated in the integrated MICMAC part of the Chapter 1, one can measure these 3D unit direction vectors of the legs from their 2D image projections, and subsequently one can numerically differentiate them to obtain the velocities of the leg orientations. But, what about the speed of measurement? Certainly, this will not be greater than a video rate $\sim 40Hz$ when a conventional camera with a conventional line tracking algorithm is used. This speed of measurement can be enough for a vision-based kinematic control, but however a dynamic control requires high-speed feedback.

Consequently, the next chapter is devoted to answer the following questions: (i) How can one measure $\underline{\mathbf{x}}$ and $\dot{\underline{\mathbf{x}}}$ at high speed in order to exploit linearity of this modeling approach? (ii) Which control law should be built upon the presented inverse dynamic model? and (iii) In which space should the control error be regulated for better dynamic control of parallel robots?

Chapter 3

Control

3.1 Introduction

Dynamic control of a robot requires high-frequency sensor feedback. In the case of a serial robot, this requirement is satisfied with dynamic control strategies based on classical high-frequency joint sensing at about 1 kHz. However, the performance of robots can be further improved by importing the vision-based control schemes [CH07], even though satisfying a reasonable accuracy and frequency for visual feedback hurts. In serial robots the solution for vision-based dynamic control is found in a cascade of two control loops, where:

- the first loop, the fast one, compensates for the dynamics. This internal loop uses an inverse dynamic model based on joint values which are easily provided by the motor encoders at high-frequency.
- the second loop, slower one, uses the feedback of a vision sensor. This outer loop is actually a kinematic control made possible by the internal loop which compensates for the dynamics of the serial robot.

This approach does not work for parallel robots, because the joint values do not determine uniquely the state of a parallel robot. The dynamic state of a parallel robot for a dynamic control, in most cases, can be simply represented by its end-effector pose and velocity, but this is not necessarily the only way. Therefore, one should now investigate how to adapt vision sensor for high-speed pose and velocity computation.

Some of the attempts to adapt vision for control schemes are as follows: Since the state of a parallel robot is expressed by its end-effector pose and velocity, these variables are tried to be found by the classical pose estimation algorithms. Unfortunately, these algorithms can not directly give the velocity information. The pose velocity is usually computed by numerical differentiation of the estimated pose, thus introducing additional noise. Besides, the predictive control techniques are exploited to adapt the visual sampling rate to the control sampling rate, but this increases the complexity as well [GM03]. Instead, from a control point of view, increasing the visual feedback frequency up to the control frequency is more appropriate [Vin00], [Cor95]. Then to do so, one compresses the image data [Ric03], builds fast communication interfaces, or embeds the image processing unit closer to the camera [WHB96], [NITM00], [CB07], etc.

We believe that these attempts encumber the system. So, the aim of this chapter is to look for a simpler solution for vision-based dynamic control of a parallel robot.

The rest of this chapter proceeds as follows: Section 3.2 exploits kinematic control to compute the posture and the velocity of a parallel robot at an instant of time through sequentially partial observation of the kinematic legs. This is a kind of fast vision-based dynamic state observer of a parallel robot where a virtual robot (a copy of the real robot) imitates its motion. This virtual robot will deliver the leg orientations $\underline{\mathbf{x}}$ and their velocities $\dot{\underline{\mathbf{x}}}$ to the inverse dynamic model presented in Chapter 2, and as well as it will provide requested feedback signal for dynamic control. In the light of MICMAC concept, from now on, we will call this observer "High-speed integrated dynamic MICMAC observer". Section 3.3 derives a versatile computed-torque control law based on different feedbacks coming from the vision-based dynamic state observer. Hence, this versatile computed-torque control allows one to control the parallel robot in different variable spaces (where a variable space should represent the state of the robot) to discover the best performance for the proposed modeling methodology in the previous chapter.

3.2 High-Speed Integrated Dynamic MICMAC Observer

Here, we propose a simpler solution inspired by the following two ideas:

– (i) Is there a visual information which can give the statics and velocity of a parallel robot for dynamic control purposes? In integrated kinematic MICMAC, it was shown that the image projection of a parallel robot's legs is a relevant alternative to express the state, since they similarly encode the static configuration of the whole mechanism [Dal07]. It was good, because control and sensing were in harmony with the kinematics since image lines correspond to the legs of the robot. On the other hand, it is not fast ($\sim 40\,Hz$) enough when a dynamic control is concerned and yet it cannot give directly the velocity of the mechanism. This is because a slow conventional camera was used and the detection of the leg edges in the full image costed too much time.

– (ii) In vision-based applications, a full image is grabbed and processed. The processing step is simply the extraction of the meaningful features from a region around an approximately predicted location. It appears that the rest of the image is untouched, and the time spent for grabbing and transmitting this part is wasted. Then, to increase the visual feedback frequency up to the control frequency, why not just grab a region of an image [URLP04], [DAMM09] where only the meaningful features exist?

Hence, the reader may now wonder: (i) How shall the previous two ideas be integrated for fast estimation? and (ii) Which compatible method shall be chosen to figure out the dynamic state of the robot?

Since a small sub-image acquisition allows only for a local observation in the field of view of the camera and the legs exist plentifully in a parallel robot, this implies *a sequential grabbing strategy* (one by one) for collecting the required amount of information from the whole mechanism. In these sub-images, we observe the contours of the legs, since in an image contours are one of the simplest visual features to detect. Then, the required information is the set of contours detected from the sub-images of the legs, which are sequentially grabbed

at discrete time instants of the motion of the robot. On the other hand, although simultaneous multiple sub-images (located anywhere in the image) grabbing strategy (non-sequential) can provide all the required information at once, it is not preferable because of the addressing problem and because it is slower for estimation.

In order to figure out the full state at each control sample time, the following methods are proposed: In [WHB96], an extended Kalman filter predicts the relative state of the robot by fusing the set of image feature points belonging to an object with some redundant measurements. In [AAALM06], a CMOS Rolling Shutter camera captures a single image row by row, which causes image artifacts for the moving objects. Then, exploiting these visual artifacts, the pose and velocity of a moving object are simultaneously estimated with a non-linear least squares method. In [DAMM09] and [Dah10], a virtual visual servoing scheme [MC02] is used and validated for estimating the state variables by sequentially grabbing the blobs of an artificial rigid pattern at high speed.

Our work also exploits the virtual visual servoing scheme associated with sequential grabbing as in [Dah10] and improves it in the sense of tracking an articulated set of legs of a parallel robot at high speed rather than tracking a rigid pattern fastened to the end-effector at high speed. Yet, since we do not need a pattern, in a roundabout way, this gets rid of the tedious calibration among the pattern, the camera and the end-effector. The objectives of this part of the chapter are as follows:

- to allow the posture and velocity of a parallel robot to be estimated through a virtual robot imitating its motion.
- to progress towards putting into effect the dynamic control of a parallel robot based on leg kinematics [OBAM11].

3.2.1 Differential Edge Kinematics of a Cylindrical Kinematic Element

Image edges of slim legs of a parallel robot are the keystones of the integrated kinematic MICMAC, and this time their differential kinematics play once more an important role for the dynamic state estimation of a parallel robot. Slim cylindrical shaped legs are the most common kinematic elements in robots. They are also easy to handle in theory and practice. Therefore, first we explain how to compute the differential kinematics of the edges of a cylindrical kinematic element. Figure 3.1 depicts a simple projective geometry of a cylindrical kinematic element in a camera frame.

3.2.1.1 Notation

- $\mathbf{B} \in \Re^{3\times 1}$ is one of the tip points lying on the revolution axis of the cylindrical kinematic element.
- $\underline{\mathbf{x}} \in \Re^{3\times 1}$ is the orientation unit vector and also corresponds to the revolution axis direction of the cylindrical kinematic element.
- $s \in \{ L\,,\ R \}$ is the literal representation for the (L) eft or the (R) ight side of the cylindrical kinematic element seen from the camera.
- $\mathbf{p}_s \in \Re^{3\times 1}$ is a projection contour point lying on the image plane and located inside a visual edge of the cylindrical kinematic element. $\mathbf{p}_s = [x, y, 1]^T$.

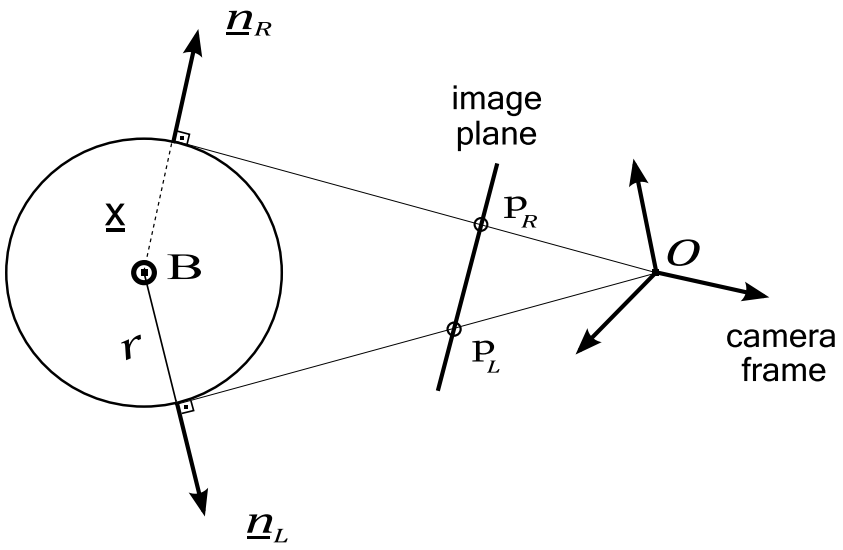


Figure 3.1 – View of the geometry of a cylindrical kinematic element from its 3D orientation direction (perpendicular to the paper plane).

- $\underline{\mathbf{n}}_s \in \Re^{3\times 1}$ is a unit vector orthogonal to the plane defined by the projection center $\mathbf{O}$ of the camera and by a side visual contour of the cylindrical kinematic element (see Fig. 3.1). Hence, it stands for a mathematical representation of such a visual contour.
- $\mathbb{X} \in \Re^{n\times 1}$ is a representation of the end-effector pose of a parallel robot.
- Unless specified in a left supper-script, all the variables are expressed in the camera frame.

3.2.1.2 Differential Edge Kinematics

Let $M_x \in \Re^{3\times n}$ and $L_B \in \Re^{3\times n}$ be respectively the differential kinematic models (e.g., for the Quattro robot see Chapter 2 equations (2.148) and (2.141)) that relate the velocity of the end-effector pose to the velocity of the orientation unit vector $\underline{\mathbf{x}}$ and to the velocity of the tip point $\mathbf{B}$ of the cylindrical kinematic element in a kinematic leg:

$$\dot{\underline{\mathbf{x}}} = M_x \ \dot{\mathbb{X}}, \qquad \dot{\mathbf{B}} = L_B \ \dot{\mathbb{X}} \tag{3.1}$$

The geometry of a cylindrical kinematic element (see Fig. 3.1) imposes the following profitable constraints:

$$\mathbf{B}^T \ \underline{\mathbf{n}}_s = -\, r\,, \qquad \underline{\mathbf{x}}^T \ \underline{\mathbf{n}}_s = 0\,, \qquad \mathbf{p}_s^T \ \underline{\mathbf{n}}_s = 0 \tag{3.2}$$

where r is the radius of the cylindrical kinematic element. *Since we would like to exploit the contours of a cylindrical kinematic element, the third constraint will be used in the virtual visual servoing scheme*. This requires the completion of a differential model defined between an edge $\underline{\mathbf{n}}_s$ and the end-effector pose $\mathbb{X}$. Exploiting the second constraint and knowing that $\dot{\underline{\mathbf{n}}}_s^T \ \underline{\mathbf{n}}_s = 0$, the image velocity of the contour $\dot{\underline{\mathbf{n}}}_s$ is expressed as follows:

$$\dot{\underline{\mathbf{n}}}_s = \alpha \ \underline{\mathbf{x}} + \beta \ (\underline{\mathbf{x}} \times \underline{\mathbf{n}}_s) \tag{3.3}$$

where α and β are two scalars to be figured out. The α comes out by differentiating the second constraint in (3.2) and replacing (3.3) into the differentiated constraint, which yields:

$$\dot{\underline{\mathbf{x}}}^T \, \underline{\mathbf{n}}_s + \underline{\mathbf{x}}^T \left(\alpha \, \underline{\mathbf{x}} + \beta \, (\underline{\mathbf{x}} \times \underline{\mathbf{n}}_s) \right) = 0 \tag{3.4}$$

that gives α as below:

$$\alpha = M_\alpha \begin{bmatrix} \dot{\mathbf{B}} \\ \dot{\underline{\mathbf{x}}} \end{bmatrix} \quad with \quad M_\alpha = \begin{bmatrix} \mathbf{0}_{1\times 3} & -\underline{\mathbf{n}}_s^T \end{bmatrix} \tag{3.5}$$

Afterwards, the β is computed by differentiating the first constraint in (3.2) and replacing (3.3) and (3.5) into the differentiated constraint, respectively. This yields:

$$\dot{\mathbf{B}}^T \, \underline{\mathbf{n}}_s + \mathbf{B}^T \left(\alpha \, \underline{\mathbf{x}} + \beta \, (\underline{\mathbf{x}} \times \underline{\mathbf{n}}_s) \right) = 0 \tag{3.6}$$

which allows to calculate β as follows:

$$\beta = M_\beta \begin{bmatrix} \dot{\mathbf{B}} \\ \dot{\underline{\mathbf{x}}} \end{bmatrix} \quad with \quad M_\beta = \begin{bmatrix} \frac{-\underline{\mathbf{n}}_s^T}{\mathbf{B}^T (\underline{\mathbf{x}} \times \underline{\mathbf{n}}_s)} & \frac{\mathbf{B}^T \underline{\mathbf{x}} \, \underline{\mathbf{n}}_s^T}{\mathbf{B}^T (\underline{\mathbf{x}} \times \underline{\mathbf{n}}_s)} \end{bmatrix} \tag{3.7}$$

Then, the differential model between an edge velocity and the end-effector pose velocity shows up by plugging (3.5), (3.7) and (3.1) into (3.3), which gives:

$$\dot{\underline{\mathbf{n}}}_s = M_s \, \dot{\mathbb{X}} \tag{3.8}$$

where $M_s \in \Re^{3\times n}$ is as below:

$$M_s = \left(\underline{\mathbf{x}} \, M_\alpha + (\underline{\mathbf{x}} \times \underline{\mathbf{n}}_s) \, M_\beta \right) \begin{bmatrix} L_B \\ M_x \end{bmatrix} \tag{3.9}$$

Finally, the complete differential model for both of the left and right edges of a cylindrical kinematic element is defined as follows:

$$\dot{\mathbf{n}} = M_n \, \dot{\mathbb{X}} \tag{3.10}$$

where $\mathbf{n} \in \Re^{6\times 1}$ and $M_n \in \Re^{6\times n}$ are as below:

$$\mathbf{n} = \begin{bmatrix} \underline{\mathbf{n}}_L \\ \underline{\mathbf{n}}_R \end{bmatrix}, \qquad M_n = \begin{bmatrix} M_L \\ M_R \end{bmatrix} \tag{3.11}$$

3.2.2 High-Speed Dynamic State Observer via Sequential Visual Sensing

3.2.2.1 Motivation for Sequential Visual Sensing

The non-sequential (simultaneous) acquisition approach instantly gives a robot's complete static posture information. On the other hand, the sequential acquisition approach requires one to take several (k) successive sub-images to collect the same amount of information. Since the sequential acquisition approach requires k successive sub-images to give the same information, one needs the previous $k-1$ sub-images to be stored. Then, the static posture can be computed

at each sub-image grabbing instant with previously stored $k-1$ sub-images. The beauty of this scenario is that it allows us to simultaneously estimate both the static posture and its velocity at the time of the latest sub-image by using the last k grabbed sub-images, when the sampling time T_{camera} between the consecutive static posture instants $\{t_k, t_{k-1}, \ldots, t_1\}$ is known. Furthermore, the estimation is a lot faster than the non-sequential approach. Table 3.1 compares the requirements of the tasks (e.g., acquisition, feature extraction, etc.) of the sequential and non-sequential sensing approaches for the computation of the static posture and the velocity of a parallel robot. To keep the comparison simple, a task time is assessed in terms of either a sub-image processing time T_{sub} or a full image processing time T_{full}, where $T_{sub} \ll T_{full}$. In Table 3.1 the algorithm for the posture computation is assumed to cost same amount of time

Table 3.1 – Comparison of sequential and non-sequential approaches.

Task	**Sequential**	**Non – Sequential**
Image acquisition	$1\,T_{sub}$	$1\,T_{full}$
Feature extraction	$1\,T_{sub}$	$1\,T_{full}$
Posture computation	$k\,T_{sub}$	$k\,T_{sub}$
Posture and velocity computation	$k\,T_{sub}$	$2k\,T_{sub}$

in both of the sequential and non-sequential approaches once all the information is available. While computing posture and velocity together, the non-sequential approach needs to calculate at least 2 sequential postures in order to obtain velocity by derivation, whereas the sequential approach gives it directly.

For example: If a 40×40 pixels sub-image is grabbed rather than a full 1024×1024 pixels image, then in sequential approach:

- image acquisition time is reduced to about 655 times.
- feature extraction time is possibly reduced to about 655 times.
- posture computation time stays the same.
- posture and velocity computation time is reduced to about 2 times.

The estimation of the dynamic state of the robot is faster when the information grabbed is smaller. On the other hand, when smaller pieces of information are grabbed, less information is kept for the present time and less accuracy is expected. Consequently, an optimum should be determined regarding this tradeoff within theoretical and physical limits.

Remark: In the robotic literature, the terms *pose* and *posture* are sometimes confused. We would like to give a clear notion of these two terms. A pose is a representation of a state of a robot. A posture is how the physical components of a robot occupy the 3D Euclidean space. A posture is always a pose, but a pose is not necessarily always a posture. A pose in a 3D task can be the position and the orientation of the end-effector of the robot, while in a 2D visual servoing task it can be the image primitives, for instance: corners, lines, circles, lightening, etc, such that they satisfy the task accomplishment. Here, the message is a pose may not provide the full geometrical configuration of the robot, nonetheless it can be defined such a way that

one can conclude the full geometry of the robot. On the other hand, a posture directly gives the full geometry of the robot and this is what we shall estimate.

3.2.2.2 Algorithm: « Single-Iteration Virtual Visual Servoing »

The classical virtual visual servoing (VVS) repeats the minimization steps of a tracking error until this error becomes smaller than a certain threshold value. Therefore, when this classical VVS is used in a vision-based control (VBC) task for tracking purposes of the robot state, the VBC loop becomes sluggish. However, unlike the classical VVS, we do not repeat the minimization steps of a tracking error more than once between 2 acquisition instants of the camera in our *single-iteration* VVS. This is because our objective is to use this fast single-iteration VVS in dynamic control of parallel robots. Table 3.2 gives the two pseudo-codes of the classical virtual visual servoing and the single-iteration virtual visual servoing which are used in a vision-based control task.

```
// Slow VBC Loop
• while ( control error > ε1 )
    • image grabbing
    • detection
    // Classical VVS Loop
    • while ( tracking error > ε2 )
        • error computation
        • correction
        • state update
    • control computation
    • move robot
```

```
// Fast VBC Loop
• while ( control error > ε )
    • sub-image grabbing
    • detection
    // Single-iteration VVS
        • error computation
        • correction
        • state update
    • control computation
    • move robot
```

Table 3.2 – Pseudo-codes for a vision-based control loop with the slow classical VVS state tracking (left) and the fast single-iteration VVS state tracking (right).

In the next subsections, we explain this fast single-iteration virtual visual servoing scheme.

3.2.2.3 Notation

- $t \in \{ t_c, \bar{t}_c \}$ denotes the time, where t_c is an acquisition instant of the camera and $\bar{t}_c$ is an estimation instant for the state variables of the virtual robot.
- $T \in \{ T_c , \overline{T}_c \}$ are the periods of sub-image acquisition of the camera and of the update of the virtual robot state variables, respectively. The virtual time period $\overline{T}_c$ should be equal to or greater than the acquisition time period T_c of the camera ($\overline{T}_c \geqslant T_c$) so that the virtual robot can catch the motion of the real robot.
- $j\,(t) \in \{ 1, 2, \dots \}$ is a function of time instants that denotes which cylindrical kinematic element is observed at time t.

3.2.2.4 Spatiotemporal Reference Signal

The reference signal, which defines the posture and the velocity of a parallel robot, is constructed from the portions of the legs' contours. These contours are extracted from the sub-images of the legs which are grabbed at successive discrete time instants of the motion of the robot. Figure 3.2 gives an example set of the grabbed sub-images during the motion of the Quattro parallel robot, and Figure 3.3 simulates this motion.

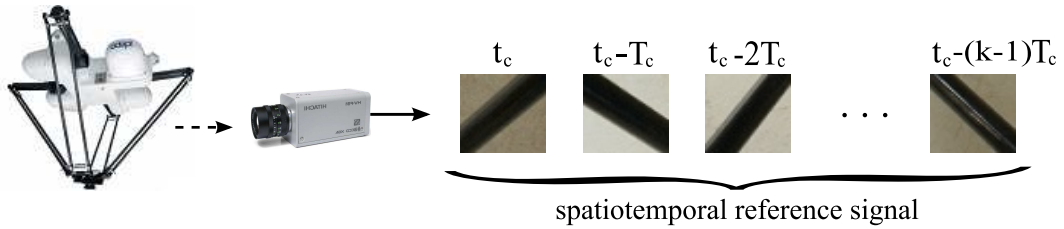


Figure 3.2 – Spatiotemporal set of the reference sub-images of the legs.

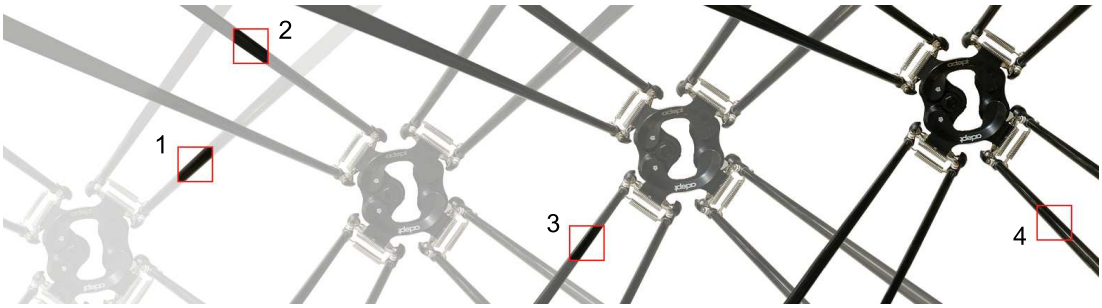


Figure 3.3 – Sequentially grabbed sub-images of the legs during the motion of the Quattro robot. The lighter the color of the robot is, the more the motion is in the past.

3.2.2.5 Sequential Postures Error

A posture error is formed with a pair of reference projection contour points (in metric units), $\{ \mathbf{p}^*_{jL\ i}, \mathbf{p}^*_{jR\ i} \}$ extracted from the sub-image of a cylindrical kinematic element of the real robot, and their associated edges (feedback signal) computed from the virtual robot's cylindrical kinematic element:

$$\mathbf{e}_{ji} = \begin{bmatrix} \mathbf{p}^{*T}_{jL\ i}\ \underline{\mathbf{n}}_{jL} \\ \\ \mathbf{p}^{*T}_{jR\ i}\ \underline{\mathbf{n}}_{jR} \end{bmatrix} \tag{3.12}$$

where $i = 1, \ldots, m$ is the index of a detected contour point in a sub-image. Figure 3.4 explains the formation of a posture error.

Then, the error vector $\mathbf{e}_j \in \Re^{2m \times 1}$ of the j^{th} cylindrical kinematic element of the virtual

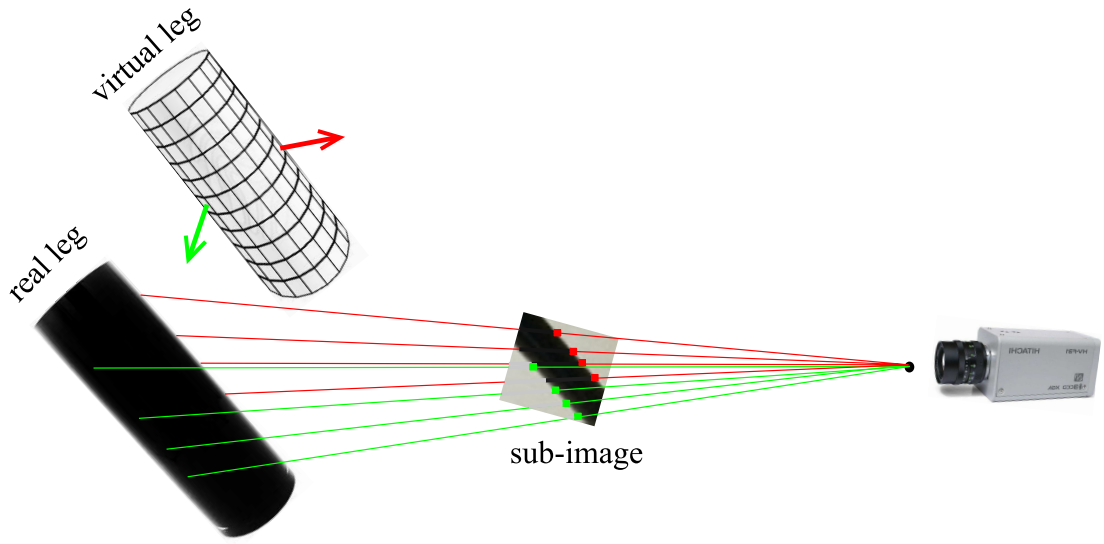


Figure 3.4 – A posture error is defined using a single sub-image which is captured from a cylindrical leg of the real robot. The detected edge pixels of this sub-image allow us to form the reference green and red side rays which pass tangent to the surface of the cylindrical leg of the real robot. And the current feedback signals are the green and red edge vectors computed from the copy of the same cylindrical leg which belongs to the virtual robot. On the condition that the reference rays (i.e., contour points) are measured and the feedback edge vectors are constructed in the same camera frame, these reference rays and these edge vectors become perpendicular to each other only when the virtual leg is superimposed onto the real leg. That is to say, minimization of error (3.12) draws the posture of the virtual robot to the posture of the real robot.

robot is noted for all the contour points as follows:

$$\mathbf{e}_j = C_j^* \ \mathbf{n}_j \tag{3.13}$$

where $\mathbf{n}$ is as in (3.11) and $C_j^* \in \Re^{2m \times 6}$ is a constant reference contour matrix:

$$C_j^* = \begin{bmatrix} P_{jL}^{*T} & \mathbf{0} \\ \mathbf{0} & P_{jR}^{*T} \end{bmatrix} \tag{3.14}$$

with $\{ P_{jL}^*, \ P_{jR}^* \}$ the detected left and right side contours of the j^{th} cylindrical kinematic element of the real robot at an instant of time:

$$P_{jL}^* = \begin{bmatrix} \mathbf{p}_{jL_1}^* & \cdots & \mathbf{p}_{jL_m}^* \end{bmatrix} \in \Re^{3 \times m} \tag{3.15}$$

$$P_{jR}^* = \begin{bmatrix} \mathbf{p}_{jR_1}^* & \cdots & \mathbf{p}_{jR_m}^* \end{bmatrix} \in \Re^{3 \times m} \tag{3.16}$$

Finally, having the sets of contour matrices from the real robot and their corresponding feedback edge pairs from the virtual robot which are saved up at k sequential discrete instants of

time:

$$\mathbf{C}^* = \left\{ \begin{array}{ccc} C^*_{j\,(t_c)} & \cdots & C^*_{j\,(t_c-(k-1)\,T_c)} \end{array} \right\}$$

$$\mathbf{N} = \left\{ \begin{array}{ccc} \mathbf{n}_{j\,(\bar{t}_c)} & \cdots & \mathbf{n}_{j\,(\bar{t}_c-(k-1)\,\overline{T}_c)} \end{array} \right\}$$

the complete error vector $\mathbf{e} \in \Re^{2\,k\,m\times 1}$ is formed by stacking the last k posture errors of the legs:

$$\mathbf{e} = \left[\begin{array}{c} C^*_{j\,(t_c)}\,\mathbf{n}_{j\,(\bar{t}_c)} \\ \vdots \\ C^*_{j\,(t_c-(k-1)\,T_c)}\,\mathbf{n}_{j\,(\bar{t}_c-(k-1)\,\overline{T}_c)} \end{array} \right] \tag{3.17}$$

where $j\,(\cdot)$ enumerates circular-wise the cylindrical kinematic elements at consecutive instants of time. Figure 3.6 shows an example for the enumeration of the lower-legs of the Quattro robot in a static pose.

In order to assemble the feedback edges $\mathbf{n}$, the virtual robot's posture is evolved back in time with a *constant velocity motion model*, since the latest estimated dynamic state $\{\mathbb{X},\,\dot{\mathbb{X}}\}$ of the robot is up to first order (i.e., velocity). Assuming that the latest estimated dynamic state and the differential kinematic models of the time instant $\bar{t}_c - \overline{T}_c$ are the approximate predictions of the time instant $\bar{t}_c$, the feedback edge set is calculated as follows:

$$\mathbf{n}_{\bar{t}_c-\Delta t} \;\approx\; \mathbf{n}_{\bar{t}_c} \;-\; \Delta t\, M_{n_{\bar{t}_c}}\,\dot{\mathbb{X}}_{\bar{t}_c} \tag{3.18}$$

where $\Delta t \;=\; i\,\overline{T}_c$ is the virtual time displacement with $i \;=\; 0, 1, \ldots,\, (k-1)$. Figure 3.5 illustrates the formation of the complete error of the postures.

3.2.2.6 Approximated Edge Evolution Model

In order to regulate this time-space error to zero, a time-space differential model between an edge $\mathbf{n}$ at time instant $t + \Delta t$ and the effector pose $\mathbb{X}$ at reference time instant t needs to be defined. To find this model, we first write small displacement of an edge $\mathbf{n}$:

$$\mathbf{n}_{t+\Delta\,t} \;=\; \mathbf{n}_t \;+\; \delta\mathbf{n}_t \tag{3.19}$$

where $\Delta\,t$ tells how far in time the displaced edge is, and where $\delta\mathbf{n}_t$ is the displacement in the edge values with respect to reference time instant t. One can approximate the displacement $\delta\mathbf{n}_t$ through (3.10):

$$\delta\mathbf{n}_t \;\approx\; M_{n_t}\,\delta\mathbb{X}_t \tag{3.20}$$

The displacement in the end-effector pose $\delta\mathbb{X}_t$ can be approximated with a constant acceleration model as follows:

$$\delta\mathbb{X}_t \;\approx\; \Delta t\,\dot{\mathbb{X}}_t \;+\; \frac{1}{2}\,\Delta t^2\,\ddot{\mathbb{X}}_t \tag{3.21}$$

If the representation of the end-effector pose $\mathbb{X}_t$ is not an element of a linear vector space, (3.21) is valid only if the rotational axis of the motion during $\Delta\,t$ time remains constant. Otherwise, it will be still acceptable for a small Δt.

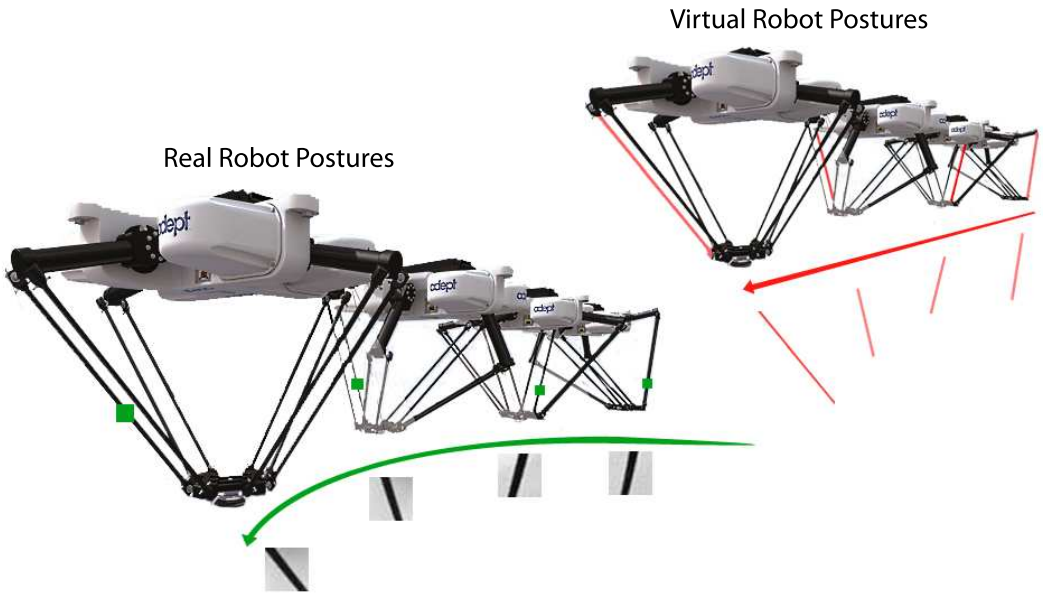


Figure 3.5 – Sequential k postures of the real robot (bottom left) and the virtual robot (top right) which imitates the real robot. We grab a sub-image per posture from a leg of the real robot and extract the edge contours. The green squares show in which posture, which leg, and which part of this leg is observed. The green arrow shows the motion path of the real robot, and below this green arrow we see the sub-images grabbed during this motion. When the last sub-image is grabbed, the virtual robot's motion is evolved with the latest estimated dynamic state $\{\mathbb{X},\, \dot{\mathbb{X}}\}$ in order to approximate the k postures of the real robot. The motion of the virtual robot evolves on a straight line whereas the motion of the real robot can be along a curve as in the green arrow. The red straight arrow shows the motion path of the virtual robot. Then, for each virtual robot posture, we find the leg which corresponds to the leg observed on the real robot. The red lines on the virtual robot postures show the corresponding observed legs of the real robot. Below the red arrow, we see the 3D cylinders of these legs recovered from the postures of the virtual robot. Finally, the stacked complete error is formed with the extracted edge contours of these k sub-images and the visual edge vectors of these k 3D cylinders.

Finally, the approximated differential model can be expressed using (3.19), (3.20) and (3.21) as below:

$$\dot{\mathbf{n}}_{t+\Delta t} \approx \frac{\mathbf{n}_{t+\Delta t} - \mathbf{n}_t}{\Delta t} = \frac{\delta \mathbf{n}_t}{\Delta t} \tag{3.22}$$

$$\dot{\mathbf{n}}_{t+\Delta t} \approx H_{(t,\,\Delta t)} \begin{bmatrix} \dot{\mathbb{X}}_t \\ \ddot{\mathbb{X}}_t \end{bmatrix} \tag{3.23}$$

where $H_{(t,\,\Delta t)} \in \Re^{6\times 2n}$ is the constant acceleration evolution model of the edge pair of a leg:

$$H_{(t,\,\Delta t)} = \begin{bmatrix} M_{n_t} & \frac{1}{2}\,\Delta t\, M_{n_t} \end{bmatrix} \tag{3.24}$$

This approximation allows us to solve the inverse kinematic problem of a parallel robot only once instead of k times, where k is equal to or greater than the number of observed legs.

3.2.2.7 Visual Servoing Control Law

The control law that will bring the error to zero is computed by first differentiating the error in (3.13) with respect to time, which gives:

$$\dot{\mathbf{e}}_j \;=\; C_j^* \;\dot{\mathbf{n}}_j \;+\; \dot{C}_j^* \;\mathbf{n}_j \tag{3.25}$$

Thus replacing $\dot{\mathbf{n}}_j$ by (3.23) and imposing $\dot{\mathbf{e}}_j = -\lambda\,\mathbf{e}_j$ for an exponential convergence, (3.25) becomes:

$$-\lambda\,\mathbf{e}_j = L_{e_j} \;\begin{bmatrix} \dot{\mathbb{X}}_t \\ \ddot{\mathbb{X}}_t \end{bmatrix} \;+\; \dot{C}_j^* \;\mathbf{n}_j \tag{3.26}$$

where $L_{e_j} \in \Re^{2m\times 2n}$ is the so-called interaction matrix which relates the end-effector pose velocity and its derivative to the implicit error function of a leg:

$$L_{e_j} = C_j^* \;H_{j(\,t,\,\Delta t)} \tag{3.27}$$

Then, so as to converge to the state of the real robot, we propose the control law $\mathbf{u} \;=\; \left[\dot{\mathbb{X}}_u^T,\, \ddot{\mathbb{X}}_u^T\right]^T$ for update of the pose and the velocity of the virtual robot which satisfies the following system:

$$L_e \;\begin{bmatrix} \dot{\mathbb{X}}_u \\ \ddot{\mathbb{X}}_u \end{bmatrix} \;=\; -\lambda\;(\,\mathbf{e} \;-\; \tilde{\mathbf{e}}\,)\;, \quad \lambda > 0 \tag{3.28}$$

where $L_e \in \Re^{\,2km\,\times 2n}$ and $\tilde{\mathbf{e}} \in \Re^{\,2km\,\times 1}$ are obtained by stacking associated interaction matrices and the individual errors at each sub-image, respectively:

$$L_e \;=\; \begin{bmatrix} L_{e_{j\,(\,\bar{t}_c)}} \\ \vdots \\ L_{e_{j\,(\,\bar{t}_c\,-\,(k-1)\,\overline{T}_c)}} \end{bmatrix}, \qquad \tilde{\mathbf{e}} \;=\; \begin{bmatrix} \dot{C}^*_{j\,(\,t_c)}\,\mathbf{n}_{j\,(\,\bar{t}_c)} \\ \vdots \\ \dot{C}^*_{j\,(\,t_c\,-\,(k-1)\,T_c)}\,\mathbf{n}_{j\,(\,\bar{t}_c\,-\,(k-1)\,\overline{T}_c)} \end{bmatrix} \tag{3.29}$$

The term $\tilde{\mathbf{e}}$ in (3.28) is difficult to approximate, because we do not use any correspondence between the successively detected reference contour points. Therefore, it will be considered as a disturbance and it will be neglected. The classical solution to (3.28) is then as follows:

$$\begin{bmatrix} \dot{\mathbb{X}}_u \\ \ddot{\mathbb{X}}_u \end{bmatrix} \;=\; -\lambda\,L_e^{\dagger}\,\mathbf{e}\,, \quad \lambda > 0 \tag{3.30}$$

However, (3.30) should not be directly calculated with such an ordinary pseudo-inverse least squares regression, since it can be unstable and slow. Here, (3.30) is just a representation of the numerical solution of (3.28). Actually, a better way to solve (3.28) uses damped total least squares and QR decomposition yielding robust and fast solutions. Thus, one can write (3.28) as follows:

$$A\,\mathbf{u} \;=\; \mathbf{b} \tag{3.31}$$

where $A \in \Re^{(2km+2n)\times 2n}$ and $\mathbf{b}^{(2km+2n)\times 1}$ are the augmented coefficient matrix and the augmented error vector, respectively:

$$A = \begin{bmatrix} L_e \\ \mu\, I \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} -\lambda\, \mathbf{e} \\ \mathbf{0} \end{bmatrix} \tag{3.32}$$

with μ damping parameter, I $(2n \times 2n)$ identity matrix, and $\mathbf{0}$ $(2n \times 1)$ zero vector. Finally, (3.31) can be solved for $\mathbf{u}$ with QR decomposition.

Indeed, high-speed control of parallel kinematic mechanisms is hardly conceived in a kinematic way, but rather in a dynamic way (computed-torque control). However, this kinematic control is relevant for tracking where $\{\, \dot{\mathbb{X}}_u,\, \ddot{\mathbb{X}}_u \,\}$ can be numerically integrated.

3.2.2.8 Virtual Parallel Robot Dynamic State Update

Pose Representations: In our case, the end-effector pose $\mathbb{X}$ of the virtual parallel robot has to be in a vector form. In the literature, some well known pseudo-vector form representations of the end-effector pose are as follows:

$$\mathbb{X}_1 = \begin{bmatrix} \mathbf{t} \\ \alpha \\ \beta \\ \gamma \end{bmatrix}, \quad \mathbb{X}_2 = \begin{bmatrix} \mathbf{t} \\ \mathbf{u}\,\theta \end{bmatrix}, \quad \ldots \tag{3.33}$$

where $\mathbf{t} = [x, y, z]^T$ is the Cartesian translation vector, $(\alpha,\, \beta,\, \gamma)$ are the Euler orientation angles, $\mathbf{u}\,\theta$ is the axis-angle orientation vector. There is also a matrix-form representation of end-effector pose which is simple to integrate through exponential formulas:

$$\mathbb{X}_3 = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \in SE(3) \tag{3.34}$$

where $R \in SO(3)$ is the orientation matrix. However, this matrix form of end-effector pose does not coherent with the estimation equations, because, for instance, the edge velocity relation in (3.8) does not fit.

Redundant Pose Representation: Nonetheless, we can still directly profit from the exponential formulas, thanks to our redundant representation of the end-effector pose which fits better to our objectives: linearity, simplicity, algebraicity, readability, codability. For example, let the redundant end-effector pose $\mathbb{X}$, its velocity $\dot{\mathbb{X}}$ and its acceleration $\ddot{\mathbb{X}}$ for a 6 degrees of freedom parallel robot be given as follows:

$$\mathbb{X} = \begin{bmatrix} \mathbf{E} \\ \underline{\mathbf{x}} \\ \underline{\mathbf{y}} \end{bmatrix}, \quad \dot{\mathbb{X}} = \begin{bmatrix} \dot{\mathbf{E}} \\ \dot{\underline{\mathbf{x}}} \\ \dot{\underline{\mathbf{y}}} \end{bmatrix}, \quad \ddot{\mathbb{X}} = \begin{bmatrix} \ddot{\mathbf{E}} \\ \ddot{\underline{\mathbf{x}}} \\ \ddot{\underline{\mathbf{y}}} \end{bmatrix} \in \Re^{9\times 1} \tag{3.35}$$

where $\mathbf{E}$, $\underline{\mathbf{x}}$ and $\underline{\mathbf{y}}$ are the origin, the unit x-axis vector, and the unit y-axis vector of the end-effector frame, respectively. The doublet of dynamic state, $\{\mathbb{X},\, \dot{\mathbb{X}}\}$, allows us easily to pass to

the linear differential twist space $se(3)$:

$$\boldsymbol{\zeta} = \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\omega} \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{E}} \\ \underline{\mathbf{x}} \times \dot{\underline{\mathbf{x}}} + \dot{\theta}\,\underline{\mathbf{x}} \end{bmatrix}, \quad \widehat{\boldsymbol{\zeta}} = \begin{bmatrix} [\boldsymbol{\omega}]_\times & \boldsymbol{v} \\ \mathbf{0} & 0 \end{bmatrix} \in se(3) \tag{3.36}$$

where $\boldsymbol{\zeta} \in \Re^{6\times 1}$ is the velocity twist of the end-effector frame and $\widehat{\boldsymbol{\zeta}} \in \Re^{4\times 4}$ is the homogenous coordinates of the velocity twist $\boldsymbol{\zeta}$. We wrote the rotational velocity $\boldsymbol{\omega}$ using Chapter 2 Lemma 1. Then, we replace the angular velocity $\dot{\theta}$ using the known motion of the y-axis as below:

$$\dot{\theta} = (\underline{\mathbf{y}} \times \dot{\underline{\mathbf{y}}})^T \underline{\mathbf{x}} \tag{3.37}$$

At this point, we can reformulate (3.36) in a matrix-vector product from $se(3)$ space to our redundant pose space:

$$\dot{\mathbb{X}} = L_{\mathbb{X}}\,\boldsymbol{\zeta}\,, \qquad L_{\mathbb{X}} = \begin{bmatrix} I_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & [\underline{\mathbf{x}}]_\times^T \\ \mathbf{0}_{3\times 3} & [\underline{\mathbf{y}}]_\times^T \end{bmatrix} \in \Re^{9\times 6} \tag{3.38}$$

and as well as from our redundant pose space to $se(3)$ space:

$$\boldsymbol{\zeta} = L_{\mathbb{X}}^{\dagger}\,\dot{\mathbb{X}}, \qquad L_{\mathbb{X}}^{\dagger} = \begin{bmatrix} I_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & [\underline{\mathbf{x}}]_\times & \left(\underline{\mathbf{x}}(\underline{\mathbf{x}} \times \underline{\mathbf{y}})^T - (\underline{\mathbf{x}} \times \underline{\mathbf{y}})^T \underline{\mathbf{x}}\, I_{3\times 3}\right) \end{bmatrix} \in \Re^{6\times 9} \tag{3.39}$$

where $I_{3\times 3}$ is 3×3 identity matrix.

Consequently, the triplet $\{\mathbb{X}, \dot{\mathbb{X}}, \ddot{\mathbb{X}}\}$ yields the acceleration twist:

$$\dot{\boldsymbol{\zeta}} = \begin{bmatrix} \dot{\boldsymbol{v}} \\ \dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \ddot{\mathbf{E}} \\ \underline{\mathbf{x}} \times \ddot{\underline{\mathbf{x}}} + \ddot{\theta}\underline{\mathbf{x}} + \dot{\theta}\,\dot{\underline{\mathbf{x}}} \end{bmatrix} \in \Re^{6\times 1} \tag{3.40}$$

where angular acceleration $\ddot{\theta}$ can be replaced by the differentiation of (3.37) with respect to time:

$$\ddot{\theta} = (\underline{\mathbf{y}} \times \ddot{\underline{\mathbf{y}}})^T \underline{\mathbf{x}} + (\underline{\mathbf{y}} \times \dot{\underline{\mathbf{y}}})^T \dot{\underline{\mathbf{x}}} \tag{3.41}$$

Static State Update: The latest estimated pose $\mathbb{X}$ in (3.35) at time instant $\bar{t}_c - \overline{T}_c$ can be updated with the control law $\{\dot{\mathbb{X}}_u, \ddot{\mathbb{X}}_u\}$ as follows:

$$\begin{bmatrix} \mathbb{X} \\ 1 \end{bmatrix}_{\bar{t}_c} = \delta T \begin{bmatrix} \mathbb{X} \\ 1 \end{bmatrix}_{(\bar{t}_c - \overline{T}_c)} \tag{3.42}$$

where $\delta T \in \Re^{10\times 10}$ is a homogenous transformation for a small displacement in a redundant Euclidean space:

$$\delta T = \begin{bmatrix} \delta R & \mathbf{0} & \mathbf{0} & \delta \mathbf{t} \\ \mathbf{0} & \delta R & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \delta R & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 \end{bmatrix} \tag{3.43}$$

The δR and $\delta \mathbf{t}$ are the orientation and position displacements generated by the exponential map of the control twists $\{\boldsymbol{\zeta}_u, \dot{\boldsymbol{\zeta}}_u\}$. These control twists are generated by the $\{\mathbb{X}, \dot{\mathbb{X}}_u, \ddot{\mathbb{X}}_u\}$ triplet as it is explained through (3.35)-(3.41). For a constant velocity evolution model, these displacements are computed as follows:

$$\begin{bmatrix} \delta R & \delta \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} = e^{\overline{T}_c \widehat{\boldsymbol{\zeta}}_u} \tag{3.44}$$

For a constant acceleration evolution model, these displacements are computed as follows:

$$\begin{bmatrix} \delta R & \delta \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} = e^{\overline{T}_c (\widehat{\boldsymbol{\zeta}}_u + \frac{1}{2} \overline{T}_c \widehat{\dot{\boldsymbol{\zeta}}}_u)} \tag{3.45}$$

where $\widehat{\boldsymbol{\zeta}}_u \in \Re^{4\times4}$ and $\widehat{\dot{\boldsymbol{\zeta}}}_u \in \Re^{4\times4}$ are the homogenous coordinates of the control twists.

Velocity State Update: The latest estimated pose velocity $\dot{\mathbb{X}}$ at time instant $\bar{t}_c - \overline{T}_c$ can be updated with the acceleration control twist $\dot{\boldsymbol{\zeta}}_u$. This acceleration control twist is computed using the triplet $\{\mathbb{X}, \dot{\mathbb{X}}, \ddot{\mathbb{X}}_u\}$ as it is explained through (3.35)-(3.41). Hence, assuming that the acceleration is constant during $\overline{T}_c$ seconds, the end-effector pose velocity is updated as follows:

$$\dot{\mathbb{X}}_{\bar{t}_c} = \dot{\mathbb{X}}_{(\bar{t}_c - \overline{T}_c)} + \overline{T}_c \left[\dot{\boldsymbol{\zeta}}_u \right] \begin{bmatrix} \mathbb{X} \\ 1 \end{bmatrix}_{(\bar{t}_c - \overline{T}_c)} \tag{3.46}$$

where $\left[\dot{\boldsymbol{\zeta}}_u \right] \in \Re^{9\times10}$ is a homogenous transformation which relates the homogenous form of the current end-effector pose $\mathbb{X}$ to the end-effector pose velocity $\dot{\mathbb{X}}$:

$$\left[\dot{\boldsymbol{\zeta}}_u \right] = \begin{bmatrix} [\dot{\boldsymbol{\omega}}_u]_\times & \mathbf{0} & \mathbf{0} & \dot{\boldsymbol{\upsilon}}_u \\ \mathbf{0} & [\dot{\boldsymbol{\omega}}_u]_\times & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & [\dot{\boldsymbol{\omega}}_u]_\times & \mathbf{0} \end{bmatrix} \tag{3.47}$$

3.2.2.9 Predicting Future Sub-Image Location

The next dynamic state estimation needs a future sub-image to be grabbed at the next future sampling time $t_c + T_c$. The position of this sub-image on the image plane must be correctly predicted from the current dynamic state $\{\mathbb{X}, \dot{\mathbb{X}}\}$ computed for the time instant t_c. Otherwise there will not be any useful signal in the grabbed sub-image and tracking will fail. *Achieving a correct prediction is, itself, a proof of the correct performance of the proposed method*. In order to predict any of the corresponding sub-image positions of the legs, we first find the likely future pose of the real robot:

$$\begin{bmatrix} \widehat{\mathbb{X}} \\ 1 \end{bmatrix}_{(t_c + T_c)} = \delta T \begin{bmatrix} \mathbb{X} \\ 1 \end{bmatrix}_{\bar{t}_c} \tag{3.48}$$

where δT is a homogenous transformation, as it is shown in (3.43), written from the exponential map of the current end-effector velocity twist $\boldsymbol{\zeta}$ using a constant velocity motion model.

Once the likely future end-effector pose $\widehat{\mathbb{X}}_{(t_c+T_c)}$ is found, it is dissolved for the tip point $\widehat{\mathbf{B}}_j$ and for the orientation unit vector $\widehat{\underline{\mathbf{x}}}_j$ of a leg through the inverse kinematic model (IKM). Subsequently, the next location of the corresponding sub-image center is calculated as follows:

$$z_j \begin{bmatrix} {}^{im}\mathbf{w}_j \\ 1 \end{bmatrix} = K\,(\,\widehat{\mathbf{B}}_j \;-\; d_j\,\widehat{\underline{\mathbf{x}}}_j\,) \tag{3.49}$$

where ${}^{im}\mathbf{w}_j \in \Re^{2\times 1}$ is the next predicted location of a sub-image center in pixel units, d_j is a distance that indicates how far along the cylindrical leg to the leg's tip point the observed region is, z_j is the projective scale factor, and K is the camera intrinsic matrix.

3.2.3 Applied to the Quattro Parallel Robot

In the case of the Quattro robot, the static posture is encoded in the contours of the 4 lower-legs. Thus, the dynamic state of the Quattro robot can be estimated by using at least 4 sub-images, which are grabbed from each of the lower-legs at consecutive discrete time instants of the motion. Figure 3.6 shows these 4 sub-images on the lower-legs of the Quattro.

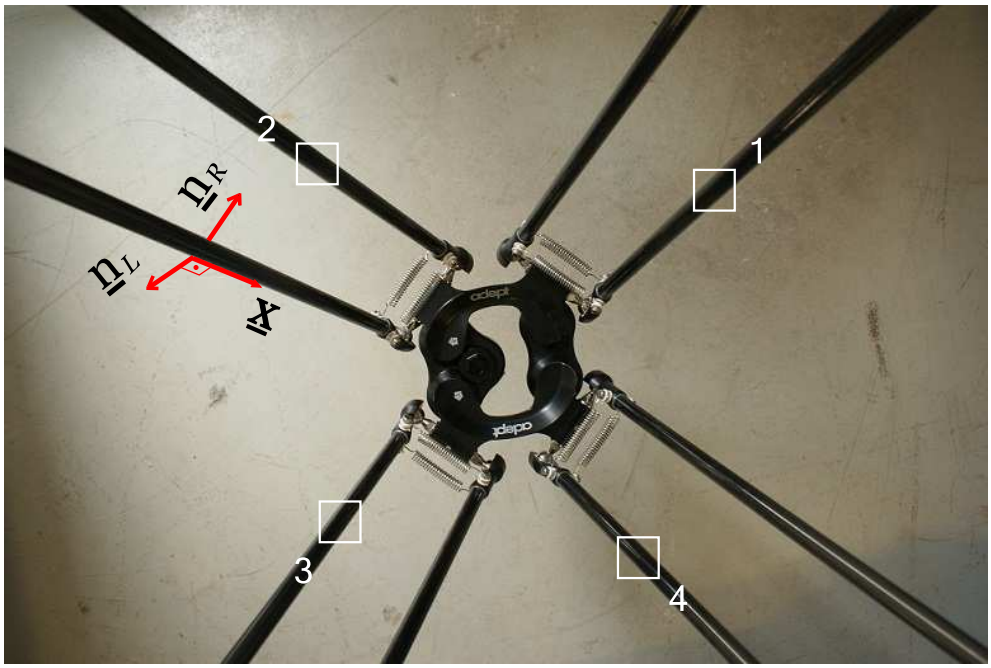


Figure 3.6 – A full image of lower-legs with their sub-images from the base-mounted camera of the Quattro robot. These sub-images are consecutively grabbed at discrete time instants and given to the virtual visual servoing as a reference. In this image, the Quattro robot is static.

3.2.3.1 On the Observability of the Legs

Furthermore, while even the lower-leg rod being observed is partially out of the field of view of the camera, we can still obtain meaningful information by keeping the sub-image in

the visible region via the parameter d_j, which slides the sub-image along the observed leg, and/or by observing the other rod if the leg is a parallelogram type.

In the case of a detected occlusion in the sub-image, one should relocate another sub-image on an unobstructed region of the same leg without forgetting about the change in the acquisition period T_c for the next state estimation. Even better, not to waste time for the next estimation, one can foresee the occlusion and can locate the sub-image directly on the unobstructed region of the leg. Consequently, this discussion poses the following questions: (i) How can one detect the occlusion in the sub-image? (ii) How can one foresee the occlusion before grabbing the sub-image?

3.2.3.2 End-Effector Pose Representation

The end-effector of the Quattro parallel robot has 4 degrees of mobility: 3 translational mobility along the x, y, z axes, and 1 rotational mobility (θ_z) around the z axis. These motion axes are decoupled and they form a linear vector space. Since this representation is linear, it eases algebra and increases accuracy of the edge evolution models. Since this representation is minimal, it accelerates computations through smaller size of matrix multiplications. Thus, in high-speed dynamic state estimation of the Quattro parallel robot, we use minimal pose representation of the end-effector:

$$\mathbb{X} = \begin{bmatrix} \mathbf{E}^T & \theta_z \end{bmatrix}^T \in \Re^{4\times 1} \tag{3.50}$$

rather than the redundant pose representation which is used in Chapter 2 for linear modeling purposes:

$$\mathbb{X} = \begin{bmatrix} \mathbf{E}^T & \underline{\mathbf{x}}_e^T \end{bmatrix}^T \in \Re^{6\times 1} \tag{3.51}$$

where $\mathbf{E}$ is the origin and $\underline{\mathbf{x}}_e$ is the x axis of the end-effector frame. The velocity of the minimal pose representation $\mathbb{X} \in \Re^{4\times 1}$ is directly in the 4 dof $se(3)$ space:

$$\dot{\mathbb{X}} = L_{\mathbb{X}} \begin{bmatrix} \boldsymbol{v} \\ \omega_z \end{bmatrix}, \qquad L_{\mathbb{X}} = I_{4\times 4} \tag{3.52}$$

where ω_z is rotational speed around the z axis (i.e., the third component of $\boldsymbol{\omega}$). Then, the dynamic state of the Quattro parallel robot can be simply updated as follows:

$$\mathbb{X}_{\bar{t}_c} = \mathbb{X}_{(\bar{t}_c - \bar{T}_c)} + \bar{T}_c \, \dot{\mathbb{X}}_u \tag{3.53}$$

$$\dot{\mathbb{X}}_{\bar{t}_c} = \dot{\mathbb{X}}_{(\bar{t}_c - \bar{T}_c)} + \bar{T}_c \, \ddot{\mathbb{X}}_u \tag{3.54}$$

3.2.3.3 Validation By Simulations

The proposed high speed state estimation approach is verified by simulation results on Matlab software. Figures 3.7 and 3.8 show the simple block representations of the estimation algorithm and of the validation process. Each estimation is done with a single-iteration virtual visual servoing. We remark that the computation of the velocity control law $\ddot{\mathbb{X}}_u$ is very ill-conditioned. Therefore, the pose velocity estimation is directly assigned equal to the pose update control law $\dot{\mathbb{X}}_u$.

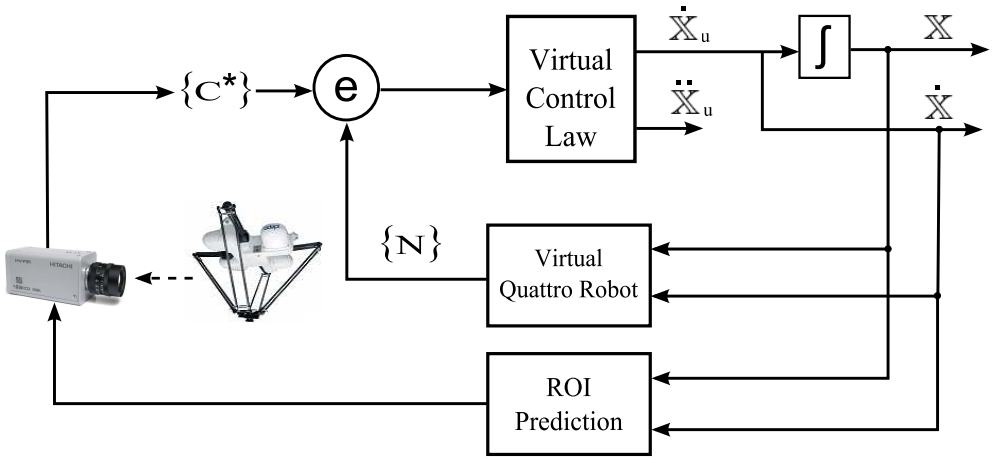


Figure 3.7 – Single-iteration virtual visual servoing for fast dynamic state estimation of the Quattro robot.

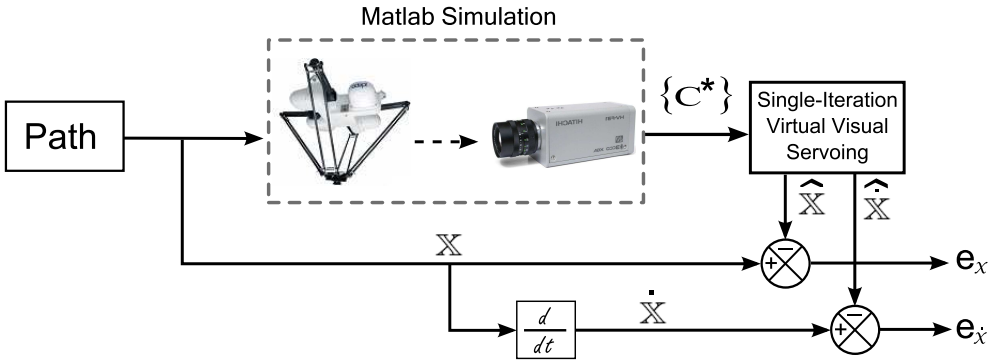


Figure 3.8 – Validation of the estimated state variables.

Acquisition Scenario: The camera sub-image acquisition frequency is set to $500\ Hz$. Then, the reference contour set will be composed of the last successively grabbed 4 sub-images of the corresponding lower-legs while the Quattro robot is being moved on a defined test trajectory. For example, the first estimation will use the set of sub-images of the lower-legs $\{1, 2, 3, 4\}$, the second will use the set of $\{2, 3, 4, 1\}$ and so on. Each sub-image is a 40×40 pixel^2 region and contains approximately 25 pixels for each side (left and right) of the observed lower-leg rod. Figure 3.9 explains the acquisition scenario of the dynamic state estimation.

Performance Metrics: The end-effector pose of the Quattro robot is composed of 3D positional part (xyz) and 1D orientational part (θ). Thus, we will evaluate the performances in these two parts. In order to evaluate the performance of the estimated states, we will use two different accuracy metrics: root-mean-square of residuals (RMSE) and Hausdorff distance. RMSE is the

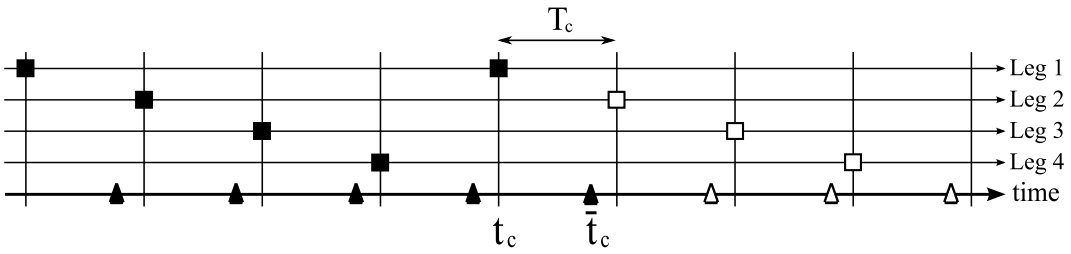


Figure 3.9 – The scenario of the sub-image acquisition instants of the legs of the Quattro robot for its dynamic state estimation. Black squares represent instants of already grabbed sub-images. White squares are future instants of sub-image grabbing. t_c is the last acquisition instant of the camera. $\bar{t}_c$ is the last estimation instant. Black triangles represent instants of already made estimations for the dynamic state of the robot. White triangles are future estimation instants. T_c is a time period of successive acquisitions of the camera.

most well-known metric to assess the tracking errors:

$$RMSE \;=\; \sqrt{\frac{\sum_{i=1}^{n} (x \;-\; x_e)_i^2}{n}} \tag{3.55}$$

where x is a known state, x_e is an estimated state which corresponds to the known state, and n is the number of estimations. However, if there is a latency in tracking, RMSE might not tell enough about the similarity of two trajectories in space. More explicitly, even if the performed trajectory by the real robot and the estimated trajectory by the single-iteration virtual visual servoing are perfectly aligned in 3D Euclidean space, RMSE might yield errors due to existing tracking latency. Therefore, we will also use Hausdorff distance metric, which can compare how close the two space curves are, without concern of the tracking latency. Let $A \;=\; \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$ and $B \;=\; \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ be the two curves of points, then Hausdorff distance is defined as follows:

$$H(A, B) \;=\; \max\{\, h(A, B),\; h(B, A) \,\} \tag{3.56}$$

where $h(A, B)$ is the maximum distance of a curve to the nearest point in the other curve:

$$h(A, B) \;=\; \max_{\mathbf{a} \in A} \{\, \min_{\mathbf{b} \in B} \|\mathbf{a} \;-\; \mathbf{b}\| \,\} \tag{3.57}$$

namely (3.57) says that for every point $\mathbf{a}$ of A, find its smallest distance to any point $\mathbf{b}$ of B; finally keep the maximum distance found among all points $\mathbf{a}$.

Test Trajectory: The test trajectory is a $0.2\ m$ diameter half-circle motion with $2\ m/s$ maximum velocity and $4G$ maximum acceleration. It is planned to span XY, XZ and YZ planes. Figure 3.10 shows traces of the test trajectory in time space. Table 3.3 tabulates accuracies of the position and orientation estimations without any noise in the system. In Table 3.3, even though there is no noise at all, the source of estimation errors are due to the following three reasons:

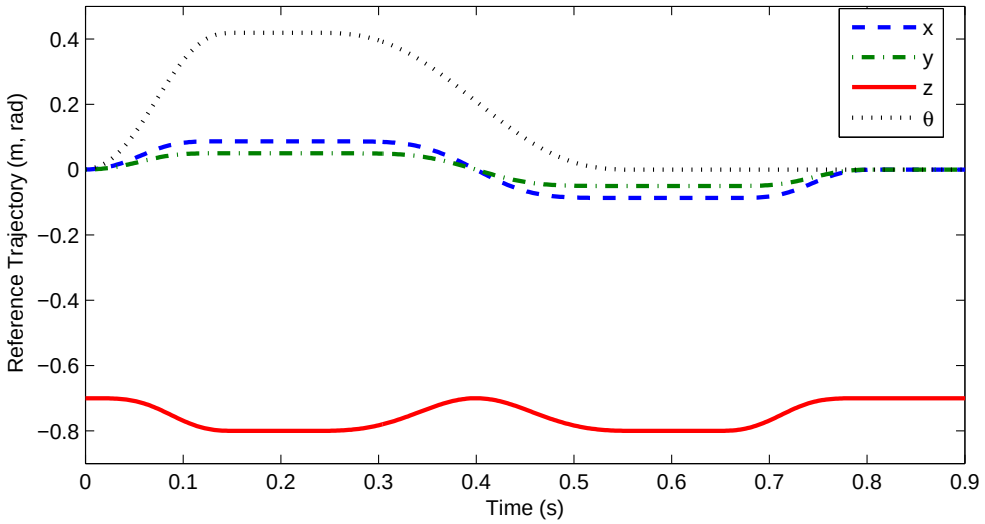


Figure 3.10 – The evolution of the reference trajectory in time in the robot base frame.

Table 3.3 – Dynamic state estimation errors (without noise).

	Pose Errors		Velocity Errors	
	$xyz\ (m)$	$\theta\ (rad)$	$\dot{x}\dot{y}\dot{z}\ (m/s)$	$\dot{\theta}\ (rad/s)$
RMSE	0.0090	0.047	0.216	1.049
Hausdorff	0.0024	0.007	0.095	0.887

– the constant velocity model being used for evolution of the feedback edges of the virtual robot while the movement of the real robot is accelerating.
– use of an approximated spatiotemporal edge evolution model for fast estimation.
– using the pose update control law for the pose velocity estimation, since the computation of the velocity update control law is very ill-conditioned.

Robustness To Noise: We imitate the following sensor noises to test the robustness of the estimation method:

– *Calibration noise:* Camera extrinsic parameters, orientation matrix R and position vector $\mathbf{t}$ with respect to robot base frame, are subjected to noise. Orientation matrix R is deflected with 1° degree around an arbitrary axis. The position vector $\mathbf{t}$ is displaced $0.005\,m$ away along an arbitrary direction.
– *Image noise:* The reference contour pixels are orthogonally and uniformly perturbed (with respect to their corresponding edges) by $[-1, +1]$ pixel.

Table 3.4 tabulates accuracies of the position and orientation estimations under the aforementioned noises. Figure 3.11 depicts the reference and estimated Cartesian space curves. Figure 3.12 plots estimated Cartesian velocities of the end-effector versus time. Figure 3.13 plots positional and orientational estimation errors versus time.

Table 3.4 – Dynamic state estimation errors (with noise).

	Pose Errors		Velocity Errors	
	$xyz\ (m)$	$\theta\ (rad)$	$\dot{xyz}\ (m/s)$	$\dot{\theta}\ (rad/s)$
RMSE	0.0108	0.083	0.221	1.143
Hausdorff	0.0052	0.023	0.163	0.792

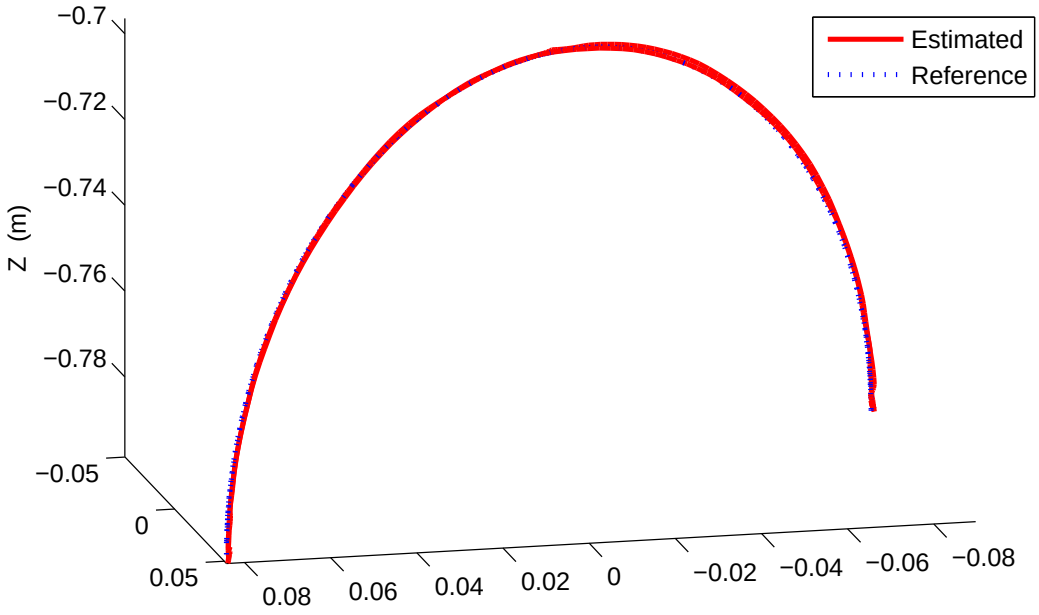


Figure 3.11 – Reference (blue dotted line) and estimated (red solid line) Cartesian space curves in the robot base frame (for the results of Table 3.4).

When Hausdorff distance results of Tables 3.3 and 3.4 are compared, we see that the calibration noise of the camera is directly pronounced on the estimated 3D trajectory, while the uniform distribution of the image noise almost has no effect. RMSE results show that we have a certain latency in tracking which remains about at the same distance without noise and with noise.

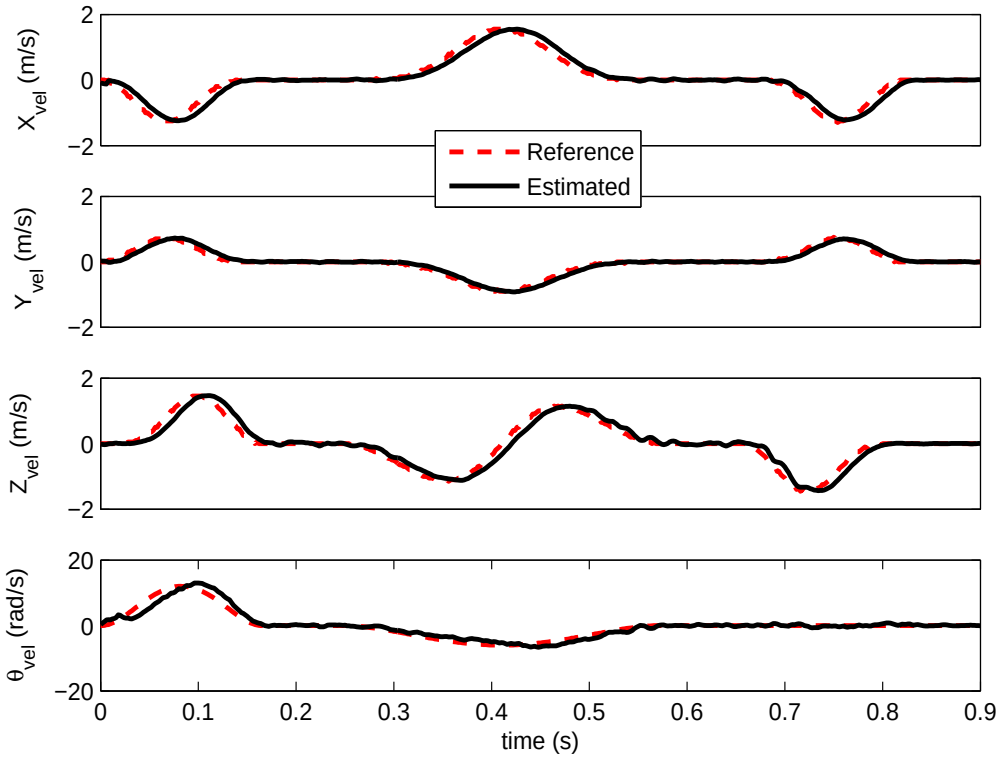


Figure 3.12 – Superimposed reference (red dotted line) and estimated (black solid line) Cartesian velocities in the camera frame (for the results of Table 3.4).

3.2.3.4 Conclusions

We validated by simulations correctness of the high-speed integrated dynamic MICMAC observer on the Quattro parallel robot. At the same time, this dynamic state observer meets the second objective of the thesis which is stated as « Fast estimation of the dynamic state (position and velocity) of the parallel robot from its leg observations. » at the end of the Chapter 1. The obtained results are promising, and we shall see what this high-speed dynamic state observer will give in experiments in Chapter 4. The next section discusses on some possible control scenarios built upon the inverse dynamic model presented in Chapter 2 and on the fast dynamic state observer explained here.

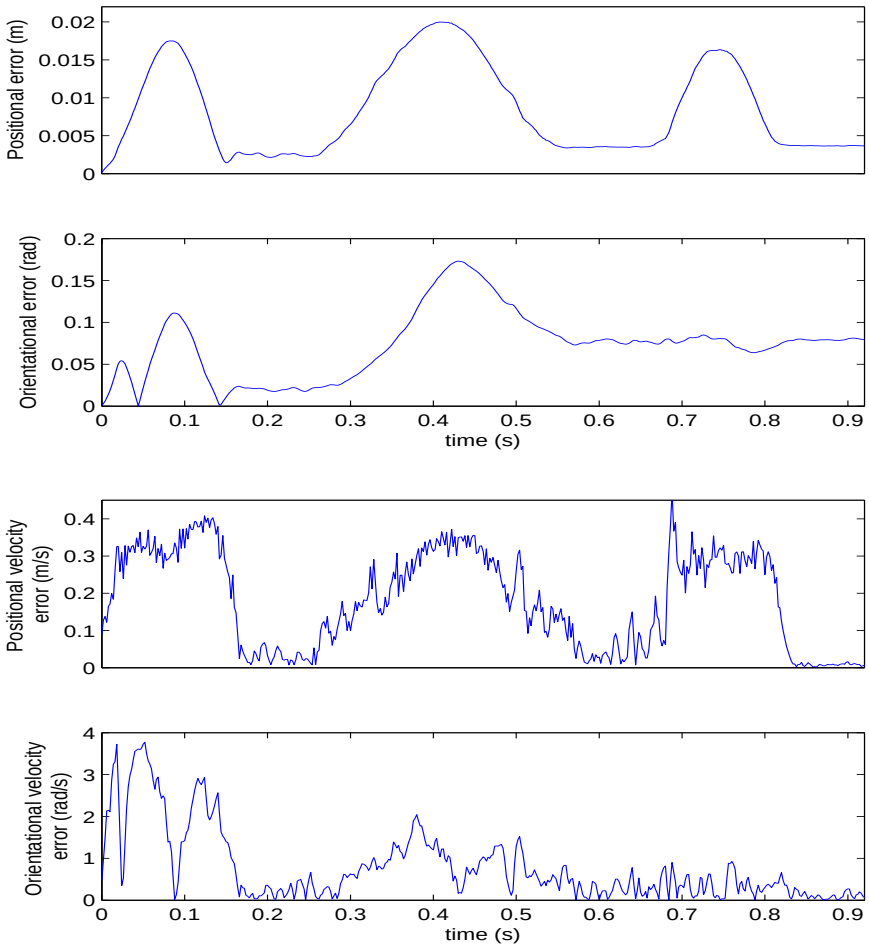


Figure 3.13 – Pose and pose velocity tracking errors versus time (for the results of Table 3.4).

3.3 Sensor-Based Computed-Torque Control based on Sequential Leg Observations

In this part, we integrate the linear inverse dynamic model and the high-speed dynamic state observer in order to define various computed-torque control laws. The objective of this part is to explore the regulation of different error functions and to discover the formative points of better control of parallel robots.

3.3.1 Control Variable

The actual sensor signal is the set of the *contours*, $\{\mathbf{C}^*\}$, of the cylindrical kinematic elements extracted from sequentially grabbed sub-images. These contours are then exploited to compute the dynamic state of the real robot through a virtual robot imitating its motion.

This virtual robot is controlled by a single-iteration virtual visual servoing scheme at each sub-image acquisition instant as it is explained in Section 3.2.2. This virtual robot can deliver every variable and its velocity needed for efficient modeling and control of the real robot. Thus, we can test our linear inverse dynamic model with different control spaces. Figure 3.14 illustrates the selection of a variable set for a control space. The chosen control space $\mathbf{s}_c$ must represent the state of the parallel robot. So, one can have different candidates for $\mathbf{s}_c$ depending on the architecture of the parallel robot. For example, one can use the following variables as a control space of the Quattro robot:

- the edges of the kinematic elements: $\{\, \underline{\mathbf{n}}_{\,Left}\,,\, \underline{\mathbf{n}}_{\,Right}\,\}$
- the orientation unit vectors of the kinematic elements: $\underline{\mathbf{x}}$
- the pose of the end-effector: $\mathbb{X}$
- the articular positions: $\mathbf{q}$

More precisely, the control space $\mathbf{s}_c$ can be noted as follows:

$$\mathbf{s}_c \in \{\, \{\, \underline{\mathbf{n}}_{\,Left}\,,\, \underline{\mathbf{n}}_{\,Right}\,\}\,,\, \underline{\mathbf{x}}\,,\, \mathbb{X}\,,\, \mathbf{q}\,,\, \ldots\,\} \tag{3.58}$$

The $\mathbf{s}_c$ can also be chosen as a combination set of those variables for optimal control purposes. Note that only $\mathbf{q}$ itself should not be $\mathbf{s}_c$, because $\mathbf{q}$ alone cannot define always the state of a parallel robot but it can be taken in any combination set of the $\mathbf{s}_c$.

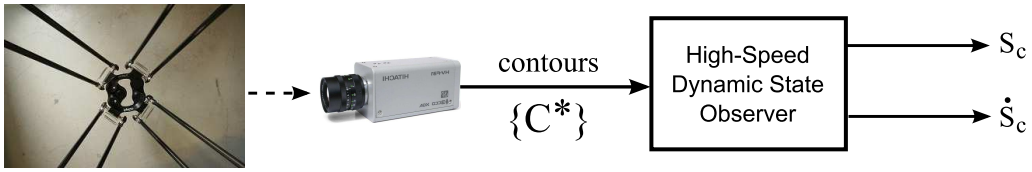


Figure 3.14 – Control space selection.

3.3.2 Versatile Control Law

In order to define a versatile control law, which can regulate any control space, we will consider that the chosen control space $\mathbf{s}_c$ is different than the state space $\mathbf{s}$ of the linearized dynamics of the robot:

$$IDM(\,\ddot{\mathbf{s}}\,,\,\dot{\mathbf{s}}\,,\,\mathbf{s}\,) \triangleq A(\mathbf{s})\,\mathbf{u} + \mathbf{h}(\mathbf{s}\,,\,\dot{\mathbf{s}}) \tag{3.59}$$

where $\ddot{\mathbf{s}}$ is replaced by the linear control law $\mathbf{u}$. Hence, we should first relate this actual control law $\mathbf{u}$ to a pseudo-control law ω. This pseudo-control law ω linearizes the dynamics of the chosen control space $\ddot{\mathbf{s}}_c = \omega$ and it is built upon an error function f_e which is written in terms of the control space variables rather than the linearized state space variables:

$$\mathbf{e} = f_e(\,\mathbf{s}_c^*(t),\, \mathbf{s}_c(t)\,) \tag{3.60}$$

where $\mathbf{s}_c^*$ is a reference state in the control space at time instant t.

In order to find the relation between the actual control law $\mathbf{u}$ and the pseudo-control law ω, we assume that there is a second-order diffeomorphism [1] between the state space $\mathbf{s}$ and the control space $\mathbf{s}_c$. If so, one can write the zeroth-order bijective mapping as follows:

$$\mathbf{s} \;=\; f_{s_c}(\,\mathbf{s}_c\,) \tag{3.61}$$

where f_{s_c} is a bijective function which maps control space to state space. Then, let the first-order diffeomorphism be noted as below:

$$\dot{\mathbf{s}} \;=\; L_{s_c}\,\dot{\mathbf{s}}_c \tag{3.62}$$

where L_{s_c} is a differential model which maps the differential control space to the differential state space. Finally, the second-order diffeomorphism is written by differentiating the first-order diffeomorphism (3.62) with respect to time:

$$\ddot{\mathbf{s}} \;=\; \dot{L}_{s_c}\,\dot{\mathbf{s}}_c + L_{s_c}\,\ddot{\mathbf{s}}_c \tag{3.63}$$

Using the second-order diffeomorphism given in (3.63), one can find the relation between pseudo and actual control laws by replacing their dynamics with themselves of the control laws $\ddot{\mathbf{s}} \;=\; \mathbf{u}$ and $\ddot{\mathbf{s}}_c \;=\; \omega$, respectively:

$$\mathbf{u} \;=\; f_u(\,L_{s_c}, \dot{L}_{s_c},\, \dot{\mathbf{s}}_c,\, \omega\,) \;=\; \dot{L}_{s_c}\,\dot{\mathbf{s}}_c \;+\; L_{s_c}\,\omega \tag{3.64}$$

where f_u is a function of the differential model L_{s_c} and its derivative, of the derivative of the control-variable $\mathbf{s}_c$ and the pseudo-control law ω. Consequently, one has to know the set $\{\,\mathbf{s}_c,\, \dot{\mathbf{s}}_c,\, L_{s_c}, \dot{L}_{s_c}\}$ to calculate the actual control law $\mathbf{u}$ by the pseudo-control law ω.

In order to show that the error f_e can be regulated too, one should prove that the linearized dynamics in the state space $(\ddot{\mathbf{s}} \;=\; \mathbf{u})$ is equivalent to the linearized dynamics in the control space $(\,\ddot{\mathbf{s}}_c = \omega)$. In order to prove this equivalence, one can rewrite the linearized dynamics of the state space using the right sides of the last two expressions in (3.63) and (3.64):

$$\dot{L}_{s_c}\,\dot{\mathbf{s}}_c \;+\; L_{s_c}\,\ddot{\mathbf{s}}_c \;=\; \dot{L}_{s_c}\,\dot{\mathbf{s}}_c \;+\; L_{s_c}\,\omega \tag{3.65}$$

and this boils down to the linearized dynamics of the control space $(\,\ddot{\mathbf{s}}_c = \omega)$, on the condition that *good approximations of the models exist* and $(\,\ddot{\mathbf{s}}_c - \omega)$ does not lie in the null-space of L_{s_c}. Finally, the pseudo-control law ω can be extracted from the following second-order error dynamics:

$$\mathbf{0} = K_{_P}\;\mathbf{e} \;+\; K_{_D}\;\dot{\mathbf{e}} \;+\; \ddot{\mathbf{e}} \tag{3.66}$$

where $K_{_P}$ and $K_{_D}$ are the proportional and derivative positive controller gains, respectively. The derivative of the error vector $\mathbf{e}$ with respect to time is written as follows:

$$\dot{\mathbf{e}} \;=\; \frac{d}{dt}\;(\,f_e(\,\mathbf{s}_c^*(t),\, \mathbf{s}_c(t)\,)\,) \;=\; \frac{\partial\, f_e}{\partial\, \mathbf{s}_c^*}\;\dot{\mathbf{s}}_c^* \;+\; \frac{\partial\, f_e}{\partial\, \mathbf{s}_c}\;\dot{\mathbf{s}}_c \tag{3.67}$$

1. invertible and differentiable smooth functions

The pseudo-control law ω which we look for is hidden in the $\ddot{\mathbf{e}}$. In order to make it appear, we differentiate the velocity of the error vector $\mathbf{e}$ with respect to time, too. This yields:

$$\ddot{\mathbf{e}} \;=\; \frac{d^2}{dt^2}\;(\,f_e(\,\mathbf{s}_c^*(t),\,\mathbf{s}_c(t)\,)\,) \;=\; \frac{d}{dt}\,(\frac{\partial\, f_e}{\partial\,\mathbf{s}_c^*})\;\dot{\mathbf{s}}_c^* \;+\; \frac{\partial\, f_e}{\partial\,\mathbf{s}_c^*}\;\ddot{\mathbf{s}}_c^* \;+\; \frac{d}{dt}\,(\frac{\partial\, f_e}{\partial\,\mathbf{s}_c})\;\dot{\mathbf{s}}_c \;+\; \frac{\partial\, f_e}{\partial\,\mathbf{s}_c}\;\omega \quad (3.68)$$

The term in front of the pseudo-control law ω in (3.68) can be expressed as below:

$$\frac{\partial\, f_e}{\partial\,\mathbf{s}_c} \;=\; C(\mathbf{s}_c^*(t)) \quad (3.69)$$

where $C(\,\mathbf{s}_c^*(t)\,)$ is a matrix written from the reference state $\mathbf{s}_c^*$ of the control space at time t. As long as this matrix $C(\,\mathbf{s}_c^*(t)\,)$ is non-singular, one can calculate the pseudo-control law ω:

$$\omega \;=\; -C^{\dagger}\;\left(\;K_P\;\mathbf{e} \;+\; K_D\;\dot{\mathbf{e}} \;+\; \frac{d}{dt}\,(\frac{\partial\, f_e}{\partial\,\mathbf{s}_c^*})\;\dot{\mathbf{s}}_c^* \;+\; \frac{\partial\, f_e}{\partial\,\mathbf{s}_c^*}\;\ddot{\mathbf{s}}_c^* \;+\; \frac{d}{dt}\,(\frac{\partial\, f_e}{\partial\,\mathbf{s}_c})\;\dot{\mathbf{s}}_c\;\right) \quad (3.70)$$

This yields a second-order convergence in $\mathbf{s}_c$. Figure 3.15 shows the block diagrams of the linearized dynamics in the control space and in the state space. Figure 3.16 describes the versatile computed-torque control scheme.

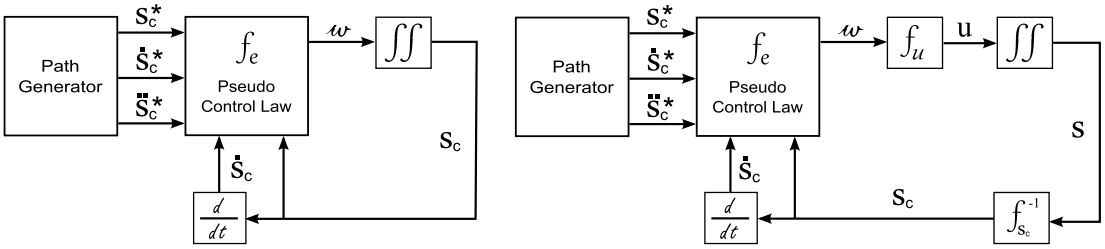


Figure 3.15 – Linearized dynamics in the control space (left) and linearized dynamics in the state space (right).

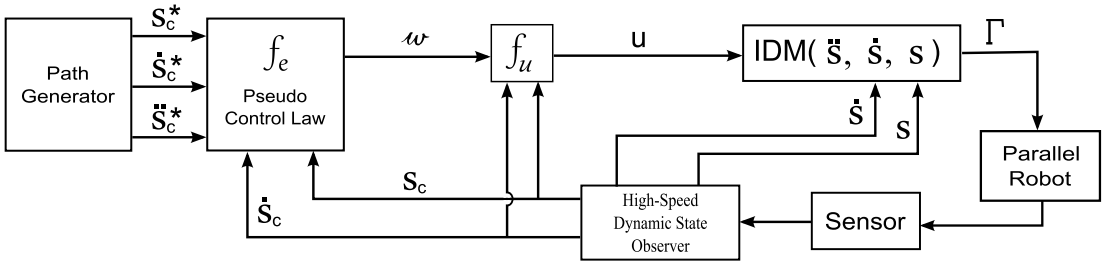


Figure 3.16 – Versatile computed-torque control scheme (V-CTC).

3.3.3 Variations Upon the Control Space

In the following parts, we propose three novel control laws for comparative purposes of the dynamic control of a parallel robot. Proposed novel control laws are derived from the aforementioned versatile computed-torque control scheme. These control laws differ from each

other by the choice of the control space (i.e., control variable $\mathbf{s}_c$) and consequently by the use of various transformations in the control scheme ($f_e,\ f_u$). Proposed control laws are applicable to any parallel robot. Here, we demonstrate the application of these novel control laws on the Quattro parallel robot in order to be consistent with the given examples throughout the thesis and to ease the following of the expressions. Before going into details of these control laws, we remind the inverse dynamic model of the Quattro parallel robot once more:

$$\mathbf{\Gamma} \ = \ IDM(\ddot{\underline{\mathbf{x}}},\ \dot{\underline{\mathbf{x}}},\ \underline{\mathbf{x}}) \tag{3.71}$$

where the inverse dynamics is expressed by means of the leg orientations $\underline{\mathbf{x}}$.

Proposition I: Body Orientation Space Computed-Torque Control (BS-CTC)

Since the $3D$ direction vectors of the kinematic elements stand almost at the heart of the inverse dynamic model of the Quattro robot ($\ddot{\underline{\mathbf{x}}} \ = \ \mathbf{u}$), the control variable set is chosen as $\mathbf{s}_c \ = \ \{\, \underline{\mathbf{x}}_{pi}\,, \underline{\mathbf{x}}_{ai}\,, \underline{\mathbf{x}}_{e}\,|_{i=1}^{4}\,\}$, and the error is directly regulated over them in order to have an efficient performance:

$$f_e(\underline{\mathbf{x}}^*,\ \underline{\mathbf{x}}) \ = \ \mathbf{e} \ = \ \begin{bmatrix} \underline{\mathbf{x}}_{p1}^* \\ \vdots \\ \underline{\mathbf{x}}_{p4}^* \\ \underline{\mathbf{x}}_{a1}^* \\ \vdots \\ \underline{\mathbf{x}}_{a4}^* \\ \underline{\mathbf{x}}_{e}^* \end{bmatrix} \ - \ \begin{bmatrix} \underline{\mathbf{x}}_{p1} \\ \vdots \\ \underline{\mathbf{x}}_{p4} \\ \underline{\mathbf{x}}_{a1} \\ \vdots \\ \underline{\mathbf{x}}_{a4} \\ \underline{\mathbf{x}}_{e} \end{bmatrix} \tag{3.72}$$

where $\{\, \underline{\mathbf{x}}_{pi}^*\,, \underline{\mathbf{x}}_{ai}^*\,, \underline{\mathbf{x}}_{e}^*\,|_{i=1}^{4}\,\}$ are the desired orientation vectors of the kinematic elements and $\mathbf{e} \in \Re^{27\times 1}$ is the orientations error vector. Afterwards, the pseudo-control law, $\omega \in \Re^{27\times 1}$, can be calculated through (3.72) and (3.70):

$$\omega \ = \ K_{_P}\ \mathbf{e} \ + \ K_{_D}\ \dot{\mathbf{e}} \ + \ \ddot{\underline{\mathbf{x}}}^* \tag{3.73}$$

One can directly use this pseudo-control law as the final control law $\mathbf{u} \ = \ \omega$, since the state space of linearized dynamics and the control space are the same space ($f_u \ = \ 1$). Figure 3.17 shows the block diagram of the body orientation space computed-torque control scheme.

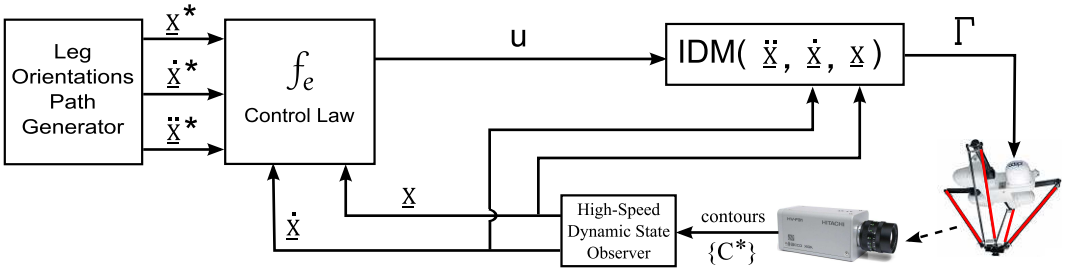


Figure 3.17 – Body orientation space computed-torque control (BS-CTC).

Note that we exploited all the direction vectors of the kinematic elements of the Quattro robot as a control signal to control its full posture. In another parallel robot, the full posture might be controlled by some of the direction vectors of the kinematic elements, such as using only the directions of the observed kinematic elements. In the case of the Quattro robot, only the directions of the observed lower-legs cannot provide a unique posture. For example, for every configuration of the lower-legs of the Quattro robot, which stays around a line that passes through the origin **O** and is parallel to the z-axis of its base frame, there is a second posture of the Quattro robot with the same configuration of the lower-legs. This second posture is where the upper-legs are symmetric to the first posture with respect to the plane passing through the motor positions at the base platform. Figure 3.18 shows examples of these postures.

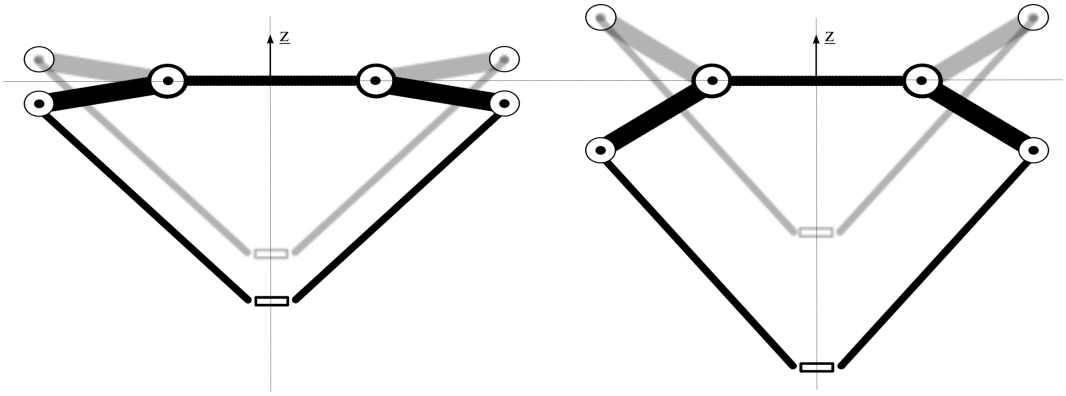


Figure 3.18 – The Quattro robot postures on the z-axis where lower-legs have the same 3D orientation vectors.

Proposition II: Leg-Based Cartesian-Space Computed-Torque Control (LCS-CTC)

In this control scheme, the minimal end-effector pose representation of the Quattro parallel robot is used as a Cartesian space control variable:

$$\mathbb{X} = \begin{bmatrix} \mathbf{E}^T & \theta_z \end{bmatrix}^T \in \Re^{4\times 1} \tag{3.74}$$

Figure 3.19 shows the block diagram of the leg-based Cartesian-space computed-torque control scheme.

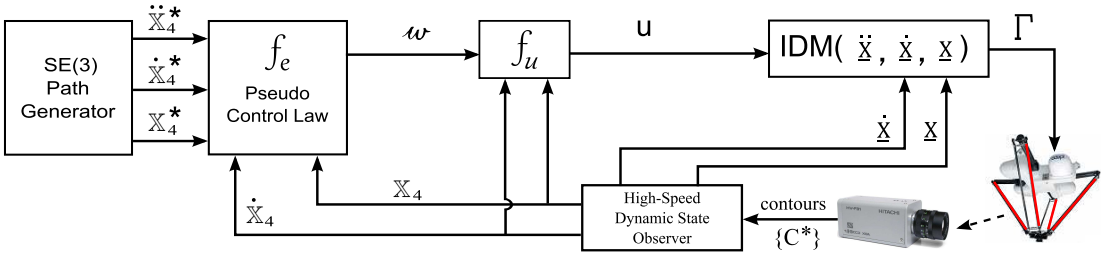


Figure 3.19 – Leg-based Cartesian-space computed-torque control scheme (LCS-CTC).

Here, we will note this minimal end-effector pose representation with $\mathbb{X}_4$ rather than $\mathbb{X}$ in order not to confuse it with the 6×1 redundant end-effector pose representation of the Quattro parallel robot. Subsequently, we will note the redundant end-effector pose representation with $\mathbb{X}_6$ for consistency of the notation. Since this minimal pose representation has decoupled components, the error can be defined as a direct difference in the Cartesian space:

$$f_e(\,\mathbb{X}_4^*,\,\mathbb{X}_4\,) \;=\; \mathbf{e} \;=\; \mathbb{X}_4^* \;-\; \mathbb{X}_4 \tag{3.75}$$

where $\mathbb{X}_4^*$ and $\mathbb{X}_4$ are the desired and the current minimal pose vectors. Afterwards, the pseudo-control law ω can be calculated through (3.75) and (3.70) as follows:

$$\omega \;=\; K_{_P}\;\mathbf{e} \;+\; K_{_D}\;\dot{\mathbf{e}} \;+\; \ddot{\mathbb{X}}_4^* \tag{3.76}$$

As a consequence of (3.76), the control law $\mathbf{u}$ can now be calculated from (3.64) as below:

$$\mathbf{u} \;=\; \dot{L}_{_{\mathbb{X}}}\;\dot{\mathbb{X}}_4 \;+\; L_{_{\mathbb{X}}}\;\omega \;\;=\;\; f_u(\,L_{_{\mathbb{X}}},\,\dot{L}_{_{\mathbb{X}}},\,\dot{\mathbb{X}}_4,\,\omega\,) \tag{3.77}$$

where $L_{_{\mathbb{X}}} \in \Re^{27\times 4}$ is the inverse differential kinematic model between the end-effector pose $\mathbb{X}_4$ and the $3D$ direction vectors of the kinematic elements. Using the differential kinematic models defined in (2.147), (2.149) and (2.152) of Chapter 2, we can write $L_{_{\mathbb{X}}}$ as follows:

$$L_{_{\mathbb{X}}} \;=\; \left[\begin{array}{c} M_p\,{}^6T_4 \\ M_a\,{}^6T_4 \\ M_e\,{}^6T_4 \end{array}\right] \tag{3.78}$$

where M_p, M_a and M_e are the differential kinematic models of the direction vectors of the upper-legs, the lower-legs, and the rotational-bar of the nacelle, respectively. ${}^6T_4 \in \Re^{6\times 4}$ is the transition matrix which maps velocity of the minimal end-effector pose representation to the redundant pose representation:

$$\dot{\mathbb{X}}_6 \;=\; {}^6T_4\,\dot{\mathbb{X}}_4\,, \qquad {}^6T_4 \;=\; \left[\begin{array}{cc} I_{3\times 3} & \mathbf{0}_{3\times 1} \\ \mathbf{0}_{3\times 1} & \underline{\mathbf{y}}_e \end{array}\right] \tag{3.79}$$

where $\mathbb{X}_6 \;=\; \left[\mathbf{E}^T\;\underline{\mathbf{x}}_e^T\right]^T \in \Re^{6\times 1}$ is the redundant pose representation which is used in Chapter 2 for linear modeling purposes, and all the differential kinematic models brought from Chapter 2 were written with respect to this redundant pose $\mathbb{X}_6$.

Proposition III: Edge-Space Computed-Torque Control (ES-CTC)

The left and the right edge equations in pixel-units $\{(\,{}^{im}\underline{\mathbf{n}}_{Left},\,{}^{im}\underline{\mathbf{n}}_{Right}\,)_i\}|_{i=1}^4 \in \Re^{3\times 1}$ of the image projections of the lower-leg rods (see Fig. 3.6) of the Quattro parallel robot are exploited directly as a control variable set. These visual edge vectors of the lower-legs are enough to define the state of the Quattro robot. Therefore, we do not need to use in addition the visual edge vectors of the other kinematic elements (e.g., upper-legs). Figure 3.20 shows the block diagram of this edge-space computed-torque control scheme. In this scheme, the error

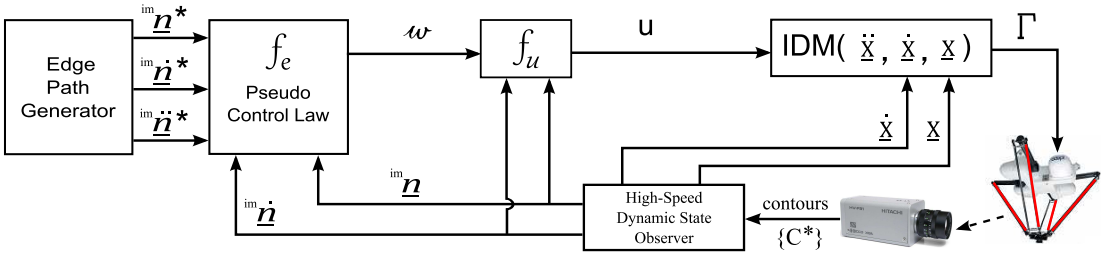


Figure 3.20 – Edge-space computed-torque control scheme (ES-CTC).

vector for each lower-leg, $\mathbf{e}_i \in \Re^{6\times 1}$, is defined as follows:

$$\mathbf{e}_i \;=\; \left[\begin{array}{c} \mathbf{e}_{Li} \\ \mathbf{e}_{Ri} \end{array}\right] = \left[\begin{array}{c} (\,{}^{im}\underline{\mathbf{n}}^{*}_{Left} \;-\; {}^{im}\underline{\mathbf{n}}_{Left}\,)_i \\ (\,{}^{im}\underline{\mathbf{n}}^{*}_{Right} \;-\; {}^{im}\underline{\mathbf{n}}_{Right}\,)_i \end{array}\right] \tag{3.80}$$

where $\{\,{}^{im}\underline{\mathbf{n}}^{*}_{Left},\, {}^{im}\underline{\mathbf{n}}^{*}_{Right}\,\}$ are the left and right desired projection-edges of a lower-leg rod. The complete error vector $\mathbf{e}$ of all the lower-legs is noted in a stacked form using (3.80) as follows:

$$f_e(\,{}^{im}\underline{\mathbf{n}}^{*},\, {}^{im}\underline{\mathbf{n}}\,) \;=\; \mathbf{e} \;=\; [\,\mathbf{e}_1^T,\, \mathbf{e}_2^T,\, \mathbf{e}_3^T,\, \mathbf{e}_4^T\,]^T \tag{3.81}$$

Hence, one can derive the differential relation between the orientation vector of a lower-leg rod and its projection-edges in pixel coordinates by differentiating the expressions below brought from (1.96) and (1.92):

$${}^{c}\underline{\mathbf{x}}_{ai} \;=\; \frac{{}^{c}\underline{\mathbf{n}}_{Left} \;\times\; {}^{c}\underline{\mathbf{n}}_{Right}}{\|\,{}^{c}\underline{\mathbf{n}}_{Left} \;\times\; {}^{c}\underline{\mathbf{n}}_{Right}\,\|}\;, \qquad {}^{c}\underline{\mathbf{n}} \;=\; \frac{K^T\;{}^{im}\underline{\mathbf{n}}}{\|\,K^T\;{}^{im}\underline{\mathbf{n}}\,\|} \tag{3.82}$$

After some algebraic calculus, the following expression appears:

$${}^{c}\dot{\underline{\mathbf{x}}}_{ai} \;=\; M_{nx_{ai}} \left[\begin{array}{c} ({}^{im}\dot{\underline{\mathbf{n}}}_{Left})_i \\ ({}^{im}\dot{\underline{\mathbf{n}}}_{Right})_i \end{array}\right] \tag{3.83}$$

where $M_{nx_{ai}} \in \Re^{3\times 6}$ is the interaction matrix between the velocities of a lower-leg rod $3D$ direction and its projection-edges:

$$M_{nx_{ai}} \;=\; \frac{\boldsymbol{\pi}({}^{c}\mathbf{x}_{ai})}{\|{}^{c}\mathbf{x}_{ai}\|} \left[\begin{array}{cc} [\,({}^{c}\underline{\mathbf{n}}_{\,Right})_i\,]_{\times}^{T}\; \frac{\boldsymbol{\pi}(({}^{c}\mathbf{n}_{Left})_i)}{\|({}^{c}\mathbf{n}_{Left})_i\|}\; K^T & [\,({}^{c}\underline{\mathbf{n}}_{\,Left})_i\,]_{\times}\; \frac{\boldsymbol{\pi}(({}^{c}\mathbf{n}_{Right})_i)}{\|({}^{c}\mathbf{n}_{Right})_i\|}\; K^T \end{array}\right] \tag{3.84}$$

and where ${}^{c}\mathbf{x}_{ai}$ and $({}^{c}\mathbf{n}_{Left/Right})_i$ are respectively the non-unit vectors of the direction of a cylindric leg and the visual edges of the same leg:

$${}^{c}\mathbf{x}_{ai} \;=\; ({}^{c}\underline{\mathbf{n}}_{Left})_i \;\times\; ({}^{c}\underline{\mathbf{n}}_{Right})_i\,, \qquad ({}^{c}\mathbf{n}_{Left/Right})_i \;=\; K^T\;({}^{im}\underline{\mathbf{n}}_{\,Left/Right})_i \tag{3.85}$$

The $\boldsymbol{\pi}(\cdot) \in \Re^{3\times 3}$ denotes an orthogonal projection matrix with respect to an associated vector. This orthogonal projection matrix $\boldsymbol{\pi}(\cdot)$ appears in the time derivation of a unit vector. For

instance, let $\mathbf{x}$ be an orientation vector of a leg of a robot, then time derivative of the normalized orientation vector can be expressed by the orthogonal projection matrix $\boldsymbol{\pi}(\mathbf{x})$ as follows:

$$\frac{d}{dt}\left(\frac{\mathbf{x}}{\|\,\mathbf{x}\,\|}\right) = \frac{1}{\|\,\mathbf{x}\,\|}\,\boldsymbol{\pi}(\mathbf{x})\,\dot{\mathbf{x}} = \frac{1}{\|\,\mathbf{x}\,\|}\left(\mathbf{I}_3 - \underline{\mathbf{x}}\,\underline{\mathbf{x}}^T\right)\,\dot{\mathbf{x}} \tag{3.86}$$

Then, we proceed by writing down all relations between the $3D$ directions of lower-legs and their projection-edges in a matrix-vector from:

$$\underbrace{\begin{bmatrix} {}^c\underline{\dot{\mathbf{x}}}_{a1} \\ {}^c\underline{\dot{\mathbf{x}}}_{a2} \\ {}^c\underline{\dot{\mathbf{x}}}_{a3} \\ {}^c\underline{\dot{\mathbf{x}}}_{a4} \end{bmatrix}}_{\underline{\dot{\mathbf{X}}}_a} = \underbrace{\begin{bmatrix} M_{nx_{a1}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & M_{nx_{a2}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & M_{nx_{a3}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & M_{nx_{a4}} \end{bmatrix}}_{M_{na}} \underbrace{\begin{bmatrix} ({}^{im}\underline{\dot{\mathbf{n}}}_{Left})_1 \\ ({}^{im}\underline{\dot{\mathbf{n}}}_{Right})_1 \\ ({}^{im}\underline{\dot{\mathbf{n}}}_{Left})_2 \\ ({}^{im}\underline{\dot{\mathbf{n}}}_{Right})_2 \\ ({}^{im}\underline{\dot{\mathbf{n}}}_{Left})_3 \\ ({}^{im}\underline{\dot{\mathbf{n}}}_{Right})_3 \\ ({}^{im}\underline{\dot{\mathbf{n}}}_{Left})_4 \\ ({}^{im}\underline{\dot{\mathbf{n}}}_{Right})_4 \end{bmatrix}}_{{}^{im}\underline{\dot{\mathbf{N}}}} \tag{3.87}$$

where $\underline{\dot{\mathbf{X}}}_a \in \Re^{12\times 1}$, $M_{na} \in \Re^{12\times 24}$ and ${}^{im}\underline{\dot{\mathbf{N}}} \in \Re^{24\times 1}$ are the stacked vector of ${}^c\underline{\dot{\mathbf{x}}}_{ai}$, the concatenated block diagonal matrix of $M_{nx_{ai}}$, and the stacked vector of $({}^{im}\underline{\dot{\mathbf{n}}}_{Left/Right})_i$, respectively. After that, we continue by writing the differential model which relates the velocities of lower-leg direction vectors to the rest of the velocities of the direction vectors of the kinematic elements:

$$\underbrace{\begin{bmatrix} {}^c\underline{\dot{\mathbf{x}}}_{p1} \\ \vdots \\ {}^c\underline{\dot{\mathbf{x}}}_{p4} \\ {}^c\underline{\dot{\mathbf{x}}}_{e} \end{bmatrix}}_{\underline{\dot{\mathbf{X}}}_{pe}} = \underbrace{\begin{bmatrix} M_p \\ M_e \end{bmatrix} M_a{}^{\dagger}}_{M_{ape}} \underbrace{\begin{bmatrix} {}^c\underline{\dot{\mathbf{x}}}_{a1} \\ \vdots \\ {}^c\underline{\dot{\mathbf{x}}}_{a4} \end{bmatrix}}_{\underline{\dot{\mathbf{X}}}_a} \tag{3.88}$$

where $M_{ape} \in \Re^{15\times 12}$ requires only a 4×4 linear system solving in its computation due to the pseudo-inverse of the $M_a \in \Re^{12\times 4}$. The M_p, M_a and M_e are again the differential kinematic models of the directions vectors of the upper-legs, the lower-legs, and the rotating rod of the nacelle, respectively. These differential kinematic models can be found in Chapter 2. The $\underline{\dot{\mathbf{X}}}_{pe} \in \Re^{15\times 1}$ is the stacked vector of ${}^c\underline{\dot{\mathbf{x}}}_{pi}$ and ${}^c\underline{\dot{\mathbf{x}}}_{e}$. Now at this point, we can write the first-order diffeomorphism between the differential control space and the differential state space:

$$\begin{bmatrix} \underline{\dot{\mathbf{X}}}_a \\ \underline{\dot{\mathbf{X}}}_{pe} \end{bmatrix} = \underbrace{\begin{bmatrix} M_{na} \\ M_{ape}\,M_{na} \end{bmatrix}}_{L_n} {}^{im}\underline{\dot{\mathbf{N}}} \tag{3.89}$$

where $L_n \in \Re^{27\times 24}$ is the differential model of the first-order diffeomorphism. Then, we write the control law $\mathbf{u} \in \Re^{27\times 1}$ from (3.64) as follows:

$$\mathbf{u} = f_u(\,L_n, \dot{L}_n,\, {}^{im}\underline{\dot{\mathbf{N}}},\, \omega\,) = \dot{L}_n\,{}^{im}\underline{\dot{\mathbf{N}}} + L_n\,\omega \tag{3.90}$$

where the pseudo-control law ω is computed by using (3.82) and (3.70):

$$\omega = K_P \, \mathbf{e} + K_D \, \dot{\mathbf{e}} + {}^{im}\ddot{\underline{\mathbf{N}}}^{*} \tag{3.91}$$

3.3.4 Validation By Simulations

The proposed vision-based computed-torque control laws are validated by simulations on the Quattro parallel robot. These simulations are conducted on the ADAMS & Simulink platform. The simulation frequency is set to $500\ Hz$. A $0.2\ m$ diameter reference circle motion with $2\ m/s$ maximum velocity and $4G$ maximum acceleration is planned such that it spans XY, XZ and YZ planes. Figure 3.21 shows this reference circle motion versus time. The simulations are executed for previously explained three computed-torque control laws: BS-CTC, LCS-CTC, and ES-CTC. Afterwards, the results are compared.

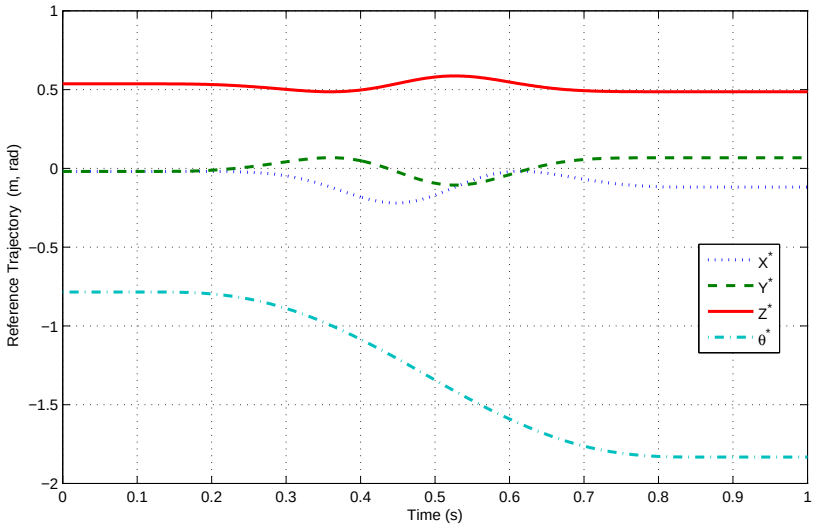


Figure 3.21 – Cartesian space reference trajectory expressed in the camera frame.

3.3.4.1 Feedback Sensing

The high-speed dynamic state observer is not integrated here for computation of the feedback signals. We did so to analyse the respective properties of the proposed control laws independently from the technological constraints. Indeed, the development of smart fast cameras in the coming years might enable fast tracking of all the legs in a simultaneous way. Moreover, our high-speed dynamic state observer can be easily adapted to the developing new sensing technologies so as to perform simultaneous posture and velocity estimations of all the legs. As a consequence, we deliberately take the following assumption:

Assumption 1 *Image projection lines* $\underline{\mathbf{n}}$ *of the lower-legs and their velocities* $\dot{\underline{\mathbf{n}}}$ *can be precisely measured at high speed and simultaneously.*

Here, we take the opportunity to propose another tracking algorithm [OAM10] based on the Assumption 1, although it is not feasible in practice until the smart fast cameras become a reality. On the other hand, this algorithm is perfectly in step with our methodology in the sense of linearity and codability. The additional advantage of this new tracking algorithm is that it forms a constructive proof of:

Lemma 2 *The edge set* $(\,\forall j\,,\ \forall i\,,\ \{\,\underline{\mathbf{n}}_{ji}\,,\,\dot{\underline{\mathbf{n}}}_{ji}\,\}\,|\,j\in\{(L)eft,(R)ight\},\ i\in\{1,2,3,4\}\,)$ *of the first cylindric rods of the lower-legs implies the dynamic state of the Quattro parallel robot, and the required variable set for kinematics and dynamics can be calculated from this set.*

Cylindrical Kinematic Element Constraints We remind once more the two profitable geometric constraints of a cylindrical rod in a lower-leg:

$$\mathbf{B}_{i1}{}^{T}\,\underline{\mathbf{n}}_{ji}\;=\;-r \tag{3.92}$$

$$\underline{\mathbf{x}}_{ai}\;=\;\frac{\underline{\mathbf{n}}_{Li}\times\underline{\mathbf{n}}_{Ri}}{\|\underline{\mathbf{n}}_{Li}\times\underline{\mathbf{n}}_{Ri}\|} \tag{3.93}$$

where $\underline{\mathbf{x}}_{ai}\,,\,\underline{\mathbf{n}}_{ji}$ and r are the direction, a projection-line and the radius of the cylindrical rod in a lower-leg, respectively. Fore further details on the robotic leg projection and 3D construction, the reader is referred to the integrated MICMAC part of the Chapter 1.

Computation of the Attachment Points Recalling the assumption that the attachment point $\mathbf{B}_{i1}$ (i.e., the point located on the nacelle) is lying on the revolution axis of the lower-leg rod with radius r, the geometric constraint in (3.92) is applied on the both of projection-lines (i.e., left and right sides) of the first rods of the lower-legs 1 and 2. This yields:

$$\begin{array}{ll}\underline{\mathbf{n}}_{L1}{}^{T}\,\mathbf{B}_{11}\;=\;-r & \underline{\mathbf{n}}_{L2}{}^{T}\,\mathbf{B}_{21}\;=\;-r\\ \underline{\mathbf{n}}_{R1}{}^{T}\,\mathbf{B}_{11}\;=\;-r & \underline{\mathbf{n}}_{R2}{}^{T}\,\mathbf{B}_{21}\;=\;-r\end{array} \tag{3.94}$$

Taking into account the nacelle parameters, one can have the following relation:

$$\mathbf{B}_{11}\;=\;\mathbf{B}_{21}\;+\;\overrightarrow{\mathbf{B}_{21}\mathbf{B}_{11}} \tag{3.95}$$

where $\overrightarrow{\mathbf{B}_{21}\mathbf{B}_{11}}$ is a constant vector, which can be retrieved by calibration or from the CAD model:

$$\overrightarrow{\mathbf{B}_{21}\mathbf{B}_{11}}\;=\;\frac{H}{2}\,\underline{\mathbf{z}}_{p2}\;-\;(d+2d_x)\underline{\mathbf{x}}_b\;-\;\frac{H}{2}\,\underline{\mathbf{z}}_{p1} \tag{3.96}$$

By replacing $\mathbf{B}_{11}$ in (3.94) with (3.95), the following linear system can be obtained from the image information:

$$\underbrace{\begin{bmatrix}\underline{\mathbf{n}}_{L1}{}^{T}\\ \underline{\mathbf{n}}_{R1}{}^{T}\\ \underline{\mathbf{n}}_{L2}{}^{T}\\ \underline{\mathbf{n}}_{R2}{}^{T}\end{bmatrix}}_{N_{j_{21}}}\mathbf{B}_{21}\;=\;\underbrace{\begin{bmatrix}-r\;-\;\underline{\mathbf{n}}_{L1}{}^{T}\,(\overrightarrow{\mathbf{B}_{21}\mathbf{B}_{11}})\\ -r\;-\;\underline{\mathbf{n}}_{R1}{}^{T}\,(\overrightarrow{\mathbf{B}_{21}\mathbf{B}_{11}})\\ -r\\ -r\end{bmatrix}}_{\boldsymbol{\beta}_{21}} \tag{3.97}$$

The least-square solution, $\mathbf{B}_{21}$, of this 4×3 linear system is unique provided that 3 of the interpretation planes are linearly independent:

$$\mathbf{B}_{21} \;=\; N_{j_{21}}^{\dagger}\, \boldsymbol{\beta}_{21} \tag{3.98}$$

where $N_{j_{21}}^{\dagger}$ is the pseudo-inverse of $N_{j_{21}} \in \Re^{4\times 3}$ and needs only 3×3 matrix inversion which can be algebraically computed. Using (3.95), one can also arrive at $\mathbf{B}_{11}$.

After that, a second linear system can be built to compute $\mathbf{B}_{31}$ and $\mathbf{B}_{41}$ by repeating the same procedure on lower-legs 3 and 4. We would like to point out that this estimation is performed in a single image. Note that this result was already verified in [DAM07] on a real I4R robot, and was adapted here for the end-effector of the Quattro robot.

Computation of the Attachment Point Velocities The velocities of the attachment points $\mathbf{B}_{i1}$ can be computed by differentiating the constraints in (3.97) and solving the linear systems for $\dot{\mathbf{B}}_{21}$ and $\dot{\mathbf{B}}_{31}$. In order to calculate $\dot{\mathbf{B}}_{21}$ the new linear system is written as follows:

$$\begin{bmatrix} \underline{\mathbf{n}}_{\,L1}^{\,T} \\ \underline{\mathbf{n}}_{\,R1}^{\,T} \\ \underline{\mathbf{n}}_{\,L2}^{\,T} \\ \underline{\mathbf{n}}_{\,R2}^{\,T} \end{bmatrix} \dot{\mathbf{B}}_{21} \;=\; \begin{bmatrix} -\underline{\dot{\mathbf{n}}}_{\,L1}^{\,T}\, \mathbf{B}_{11} \\ -\underline{\dot{\mathbf{n}}}_{\,R1}^{\,T}\, \mathbf{B}_{11} \\ -\underline{\dot{\mathbf{n}}}_{\,L2}^{\,T}\, \mathbf{B}_{21} \\ -\underline{\dot{\mathbf{n}}}_{\,R2}^{\,T}\, \mathbf{B}_{21} \end{bmatrix} \tag{3.99}$$

while $\dot{\mathbf{B}}_{31}$ can be computed similarly. Then, velocities of the (attachment) points that are located on the same rigid part of the nacelle will be equal:

$$\dot{\mathbf{C}}_1 \;=\; \dot{\mathbf{C}}_2 \;=\; \dot{\mathbf{B}}_{21} \;=\; \dot{\mathbf{B}}_{11} \tag{3.100}$$

$$\dot{\mathbf{C}}_3 \;=\; \dot{\mathbf{C}}_4 \;=\; \dot{\mathbf{B}}_{31} \;=\; \dot{\mathbf{B}}_{41} \tag{3.101}$$

Required Variable Set Looking at carefully to the modeling of the Quattro robot at the end of the Chapter 2, one can list the required variable set for kinematics and dynamics as follows:

- $\{\, \underline{\mathbf{x}}_{pi},\ \dot{\underline{\mathbf{x}}}_{pi},\ \ddot{\underline{\mathbf{x}}}_{pi},\ \underline{\mathbf{y}}_{pi}\,\}$ the variables related to the active upper-legs.
- $\{\, \underline{\mathbf{x}}_{ai},\ \dot{\underline{\mathbf{x}}}_{ai},\ \ddot{\underline{\mathbf{x}}}_{ai}\}$ the variables related to the passive lower-legs.
- $\{\, \underline{\mathbf{x}}_{e},\ \dot{\underline{\mathbf{x}}}_{e},\ \ddot{\underline{\mathbf{x}}}_{e},\ \underline{\mathbf{y}}_{e}\,\}$ the variables related to the passive nacelle.

So, one can start computing the zero-order variables of the robot. The variables of the upper-legs are given as follows:

$$\underline{\mathbf{x}}_{pi} \;=\; \frac{1}{\ell_{pi}} \left(\mathbf{B}_{i1} \;-\; \mathbf{P}_i \;-\; \overrightarrow{\mathbf{A}_i\mathbf{A}_{i1}} \;-\; \ell_{ai}\, \underline{\mathbf{x}}_{ai} \right) \tag{3.102}$$

$$\underline{\mathbf{y}}_{pi} \;=\; \underline{\mathbf{z}}_{pi} \;\times\; \underline{\mathbf{x}}_{pi} \tag{3.103}$$

where ℓ_{pi}, ℓ_{ai}, $\mathbf{P}_i$, $\overrightarrow{\mathbf{A}_i\mathbf{A}_{i1}}$ and $\underline{\mathbf{z}}_{pi}$ are the constant parameters and vectors. The nacelle variables are expressed as below:

$$\underline{\mathbf{x}}_{e} \;=\; (\mathbf{C}_2 \;-\; \mathbf{C}_3)/h \tag{3.104}$$

$$\underline{\mathbf{y}}_{e} \;=\; \underline{\mathbf{z}}_{e} \;\times\; \underline{\mathbf{x}}_{e} \tag{3.105}$$

where $\underline{\mathbf{z}}_e$ and h are constants, and the $\mathbf{C}_2$ and $\mathbf{C}_3$ can be represented as follows:

$$\mathbf{C}_2 \;=\; \mathbf{B}_{21} \;+\; \overrightarrow{\mathbf{B}_{21}\mathbf{C}_2} \tag{3.106}$$

$$\mathbf{C}_3 \;=\; \mathbf{B}_{31} \;+\; \overrightarrow{\mathbf{B}_{31}\mathbf{C}_3} \tag{3.107}$$

with $\{\, \overrightarrow{\mathbf{B}_{21}\mathbf{C}_2}\,, \overrightarrow{\mathbf{B}_{31}\mathbf{C}_3}\}$ constant vectors and the $\{\, \mathbf{B}_{21}\,, \mathbf{B}_{31}\}$ attachment points of the lower-legs to the nacelle.

The first-order variables of the passive nacelle can be calculated as follows:

$$\dot{\underline{\mathbf{x}}}_e \;=\; (\dot{\mathbf{C}}_2 \;-\; \dot{\mathbf{C}}_3)/h \tag{3.108}$$

$$\dot{\underline{\mathbf{y}}}_e \;=\; \underline{\mathbf{z}}_e \;\times\; \dot{\underline{\mathbf{x}}}_e \tag{3.109}$$

After that, the end-effector pose $\mathbb{X}$ and its velocity $\dot{\mathbb{X}}$ can be expressed as below:

$$\mathbb{X} \;=\; \begin{bmatrix} (\mathbf{B}_{21} \;+\; \overrightarrow{\mathbf{B}_{21}\mathbf{C}_2} \;-\; \frac{h}{2}\,\underline{\mathbf{x}}_e \;-\; a\,\underline{\mathbf{y}}_e) \\ \underline{\mathbf{x}}_e \end{bmatrix} \tag{3.110}$$

$$\dot{\mathbb{X}} \;=\; \begin{bmatrix} (\dot{\mathbf{B}}_{21} \;-\; \frac{h}{2}\,\dot{\underline{\mathbf{x}}}_e \;-\; a\,\dot{\underline{\mathbf{y}}}_e) \\ \dot{\underline{\mathbf{x}}}_e \end{bmatrix} \tag{3.111}$$

Then, the rest of the first-order variables (upper-legs, lower-legs, parts of nacelle) is obtained as below:

$$\dot{\underline{\mathbf{X}}} \;=\; \begin{bmatrix} M_p \\ M_a \\ M_e \end{bmatrix} \dot{\mathbb{X}} \;=\; M_{\mathbb{X}}\, \dot{\mathbb{X}} \tag{3.112}$$

where $\underline{\mathbf{X}} \in \Re^{27\times 1}$ is the variable vector of the system:

$$\underline{\mathbf{X}} \;=\; \begin{bmatrix}\; \underline{\mathbf{x}}_{p1}^T, \ldots, \underline{\mathbf{x}}_{p4}^T,\; \underline{\mathbf{x}}_{a1}^T, \ldots, \underline{\mathbf{x}}_{a4}^T,\; \underline{\mathbf{x}}_e^T \;\end{bmatrix}^T \tag{3.113}$$

and where $M_{\mathbb{X}} \in \Re^{27\times 6}$ is the interaction matrix between the end-effector pose and the system variables. The second-order variables can be computed by differentiating (3.112) as follows:

$$\ddot{\underline{\mathbf{X}}} \;=\; \dot{M}_{\mathbb{X}}\, \dot{\mathbb{X}} \;+\; M_{\mathbb{X}}\, \ddot{\mathbb{X}} \tag{3.114}$$

The actuated joint speeds $\dot{\mathbf{q}}$ and their accelerations $\ddot{\mathbf{q}}$ which are required for computation of the actuator inertial and frictional torques can be also obtained as below:

$$\dot{\mathbf{q}} \;=\; M_q\, \dot{\mathbb{X}} \tag{3.115}$$

$$\ddot{\mathbf{q}} \;=\; \dot{M}_q\, \dot{\mathbb{X}} \;+\; M_q\, \ddot{\mathbb{X}} \tag{3.116}$$

where $\dot{M}_{\mathbb{X}}$, M_q and $\dot{M}_q$ are written from the zero-order and the first-order variables which already exist since (3.112). The second-order time derivative of the end-effector pose will be coming directly from the control-law (if the control-variable is chosen as the end-effector pose) or it will be computed through the second-order diffeomorphism between the chosen control-variable and the end-effector pose.

Thereby, at this point *we substantiate that exploiting only the image edges* $\underline{\mathbf{n}}$ *and their velocities* $\dot{\underline{\mathbf{n}}}$ *of the lower-legs of the Quattro parallel robot, it is possible to figure out the whole variable set for dynamic control.* Note that this confluence is made easy, thanks to the vector-based formulation of both the dynamics and the differential geometry in the image. □

Figure 3.22 shows the block diagram of the versatile computed-torque control scheme integrated with this edge-based linear dynamic state observer.

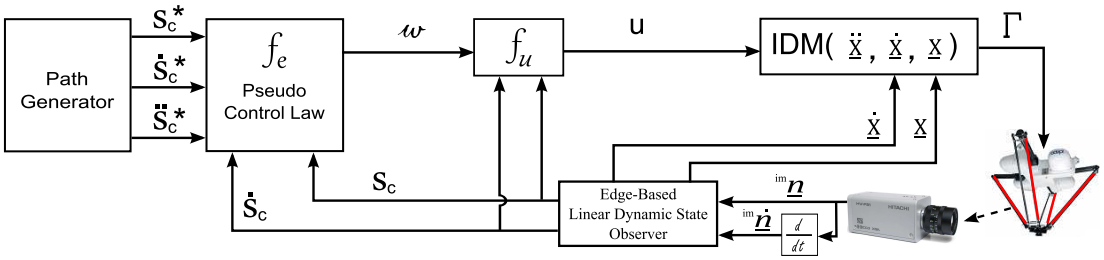


Figure 3.22 – Versatile computed-torque control scheme integrated with the edge-based linear dynamic state observer.

3.3.4.2 Noises for Robustness Test

The following two noise types are contaminated to the system to test the robustness of the control laws:

- **Mechanical noise**: Firstly, $100\mu m$ of uncertainty is injected on the 3D coordinates of the extremity points $\{\mathbf{A}_{i1}, \mathbf{B}_{i1}\}$ of the lower-legs of the Quattro robot so as to imitate the effects of clearances in passive joints, assembly errors, etc. This noise has a great impact on the orientations of the lower-legs. A good calibration is a must in the case of ignorance of that kind of mechanical errors.
- **Sensory noise**: Afterwards, for sensory noise, the locations of the visual contours of a lower-leg are orthogonally perturbed (with respect to its noiseless projection-line) in between $[-2, +2]$ pixels. This noise makes the new fitted line take a slight deflection off the previous noiseless one.

3.3.4.3 Performance Metric

The accuracy of the proposed control laws is assessed in terms of mean and standard deviation values of the position (xyz) and orientation (θ) tracking errors of the end-effector pose. Table 3.5 lists these accuracy results for each control law obtained under previously explained noise types. The mean values are shown in **bold** font and the standard deviation values are shown in *italic* font.

3.3.4.4 End-Effector Computed-Torque Control (EE-CTC)

We also performed another computed-torque control with a feedback pose estimated by the direct observation of the end-effector pose (EE-CTC) instead of computing this pose from the lower-leg edges. This feedback pose $\mathbb{X}_4$ is corrupted with a $\{100\,\mu m,\ 0.01^\circ\}$ noise which corresponds to state-of-the-art accuracy of high-speed vision. The results of this EE-CTC are shown in the last row of the Table 3.5 below of LCS-CTC column. Figure 3.23 shows the block diagram of the end-effector computed-torque control scheme.

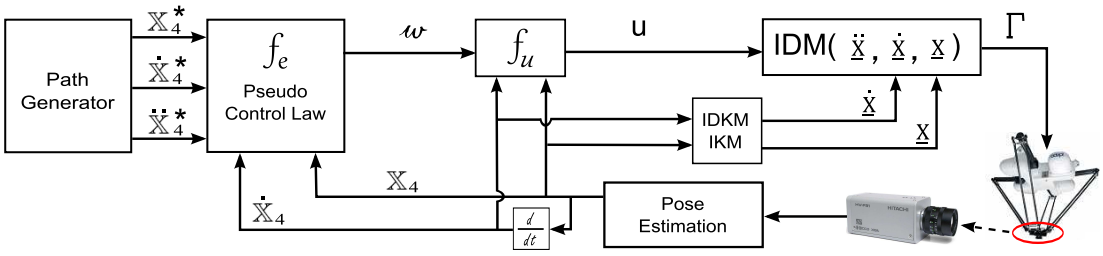


Figure 3.23 – End-effector computed-torque control scheme (EE-CTC). IDKM is the inverse differential kinematic model. IKM is the inverse kinematic model.

3.3.4.5 Results

Figures 3.24, 3.25, and 3.26 depict traces of the performed trajectories and applied torques obtained under the noises given in the fourth row of the Table 3.5. Observing results in Table 3.5, one can immediately conclude that LCS-CTC performs better and ES-CTC performs worse than the others. It is surprising to have that result while our expectations are put on the ES-CTC since the control variable ${}^{im}\underline{\mathbf{n}}$ is directly defined in the very sensor-space. However, differences on the orders of magnitudes of the errors are not so decisive to promote one over the others.

Going into details of results given in Table 3.5, one can end up that: ES-CTC and BS-CTC seem robust only to the noises in the sensor space (line fitting easily smooths out the 2D sensory noise), while being sensitive to the mechanical errors. They are slightly better in rotation but slightly worse in translation than LCS-CTC. It seems that, the closer the control space to the operational space of the robot is, the better the results are. Moreover, the superior robustness of LCS-CTC to both types of noise (i.e., mechanical and sensory) can be explained by the fact that the pose is calculated from the projection-lines of the lower-legs. This imposes explicitly the closed-loop kinematic constraint that is helping to smooth out the 3D mechanical noise.

In the applied torques LCS-CTC performs better too, while the others are more oscillatory and peaky. One can observe these oscillations and peaks in Figures 3.24 and 3.25.

Let us finally remark that EE-CTC is worse than any other proposed controls, which confirms that *observing the lower-legs is probably one good way to the enhanced accuracy*.

Note that those results were achieved with a PD+FF controller under the assumption of a perfect decoupling and linearizing of the dynamics. In practice, due to noise, this assumption might not be valid and the actual performance of the system should be improved by advanced control techniques.

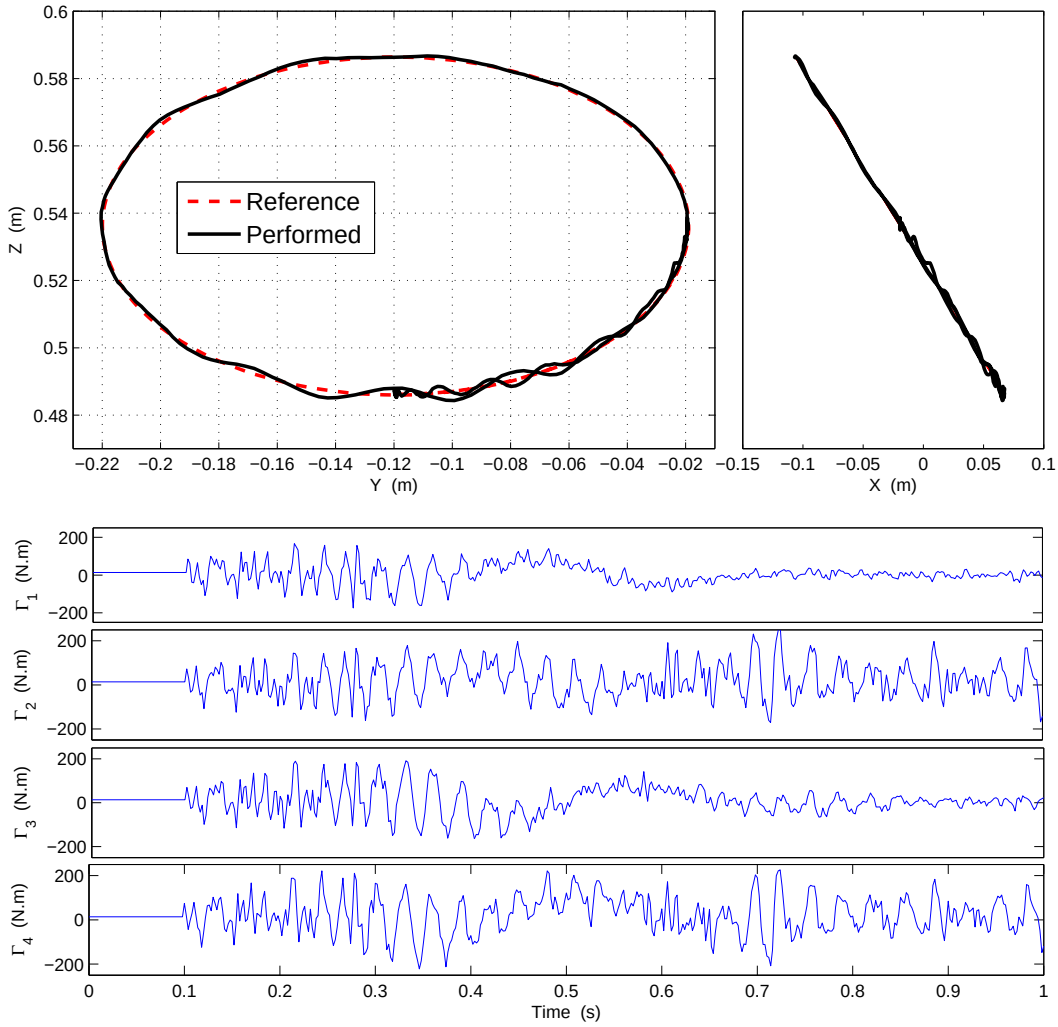


Figure 3.24 – Reference (red) and performed (black) superimposed trajectories in ZY and ZX planes (top), motor torques (bottom) for the ES-CTC.

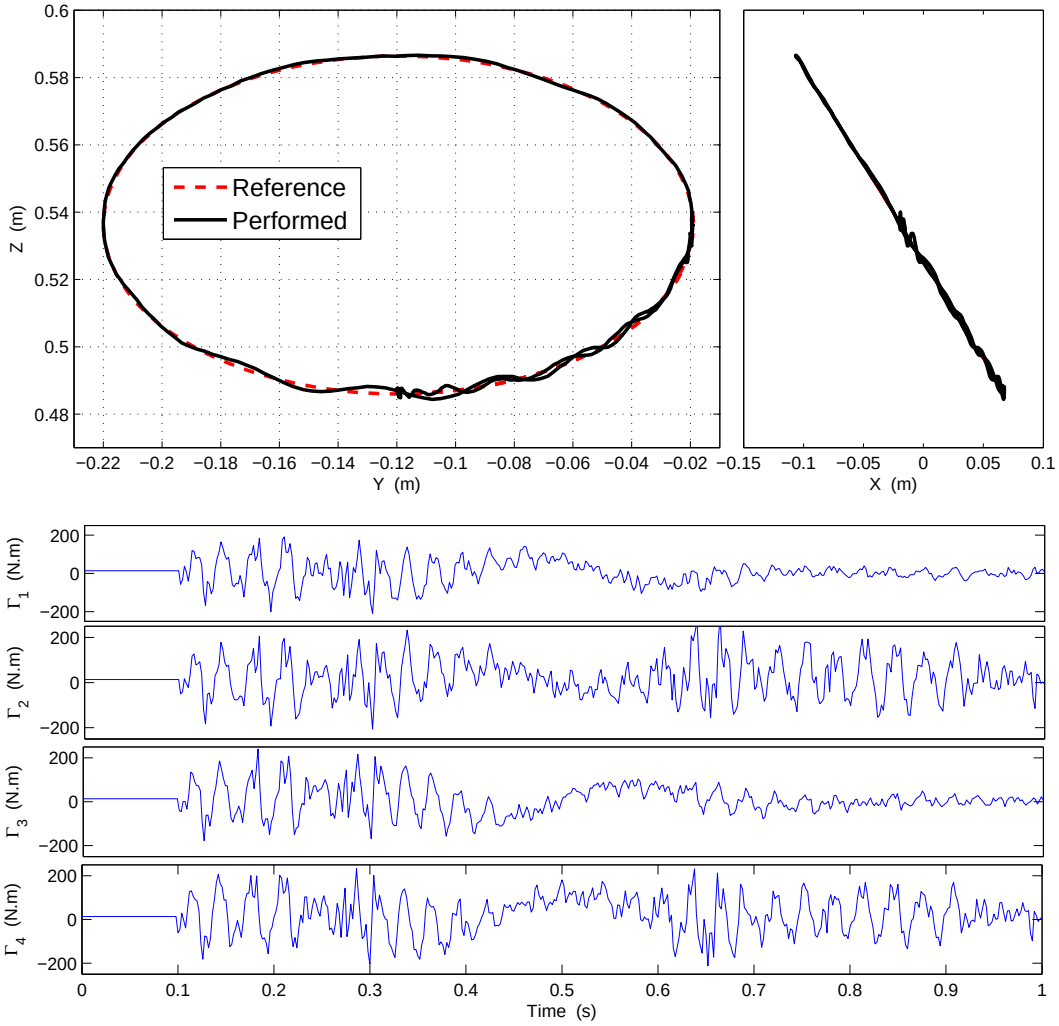


Figure 3.25 – Reference (red) and performed (black) superimposed trajectories in ZY and ZX planes (top), motor torques (bottom) for the BS-CTC.

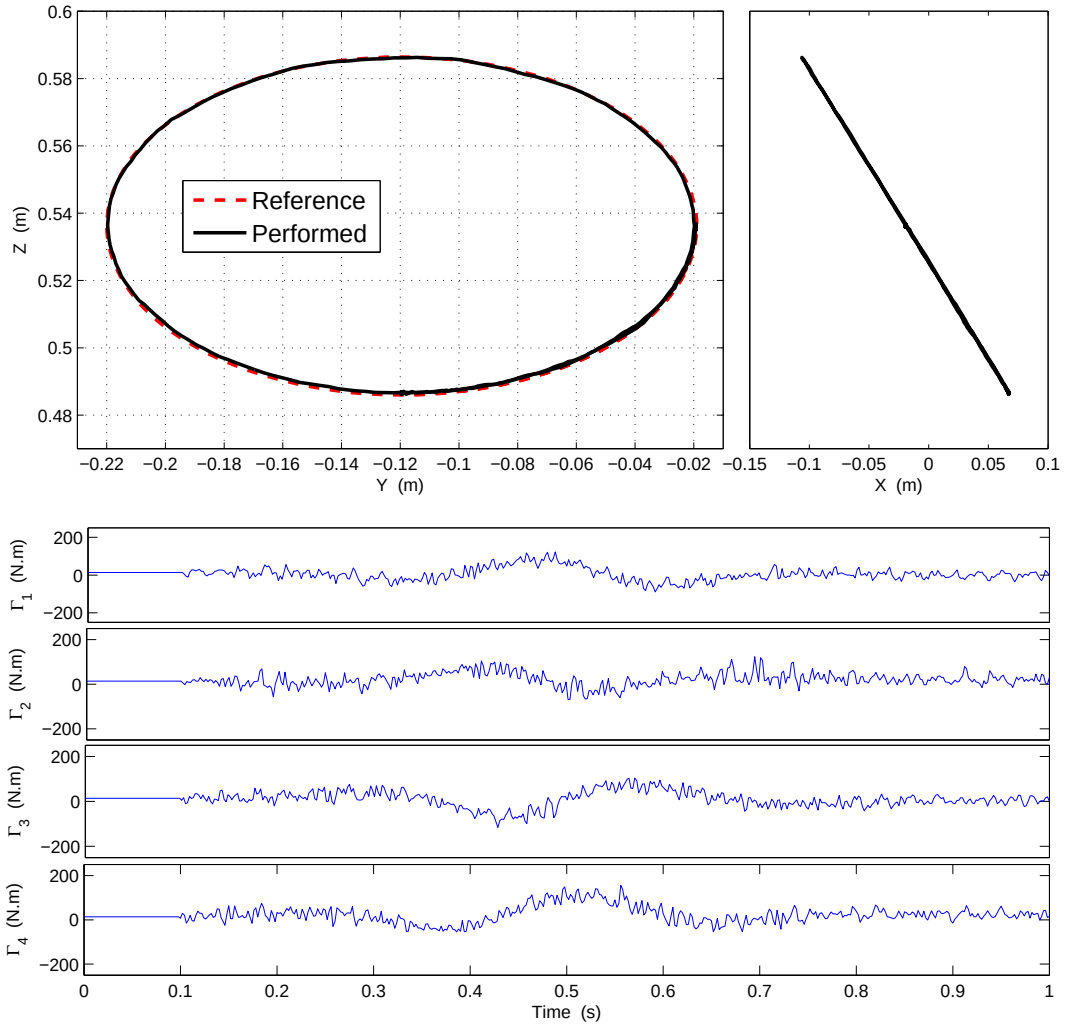


Figure 3.26 – Reference (red) and performed (black) superimposed trajectories in ZY and ZX planes (top), and motor torques (bottom) for the LCS-CTC.

Table 3.5 – Tracking errors versus noises in different control spaces.

	ES-CTC		BS-CTC		LCS-CTC	
	xyz (μm)	θ (deg)	xyz (μm)	θ (deg)	xyz (μm)	θ (deg)
$no\ noise$	**408**	**0.23°**	**356**	**0.23°**	**359**	**0.23°**
	239	*0.19°*	*190*	*0.16°*	*182*	*0.16°*
$100\,\mu m$	**674**	**0.32°**	**652**	**0.37°**	**553**	**0.36°**
	447	*0.26°*	*385*	*0.26°*	*269*	*0.25°*
$\pm 2\,pixels$	**553**	**0.22°**	**522**	**0.22°**	**529**	**0.34°**
	428	*0.18°*	*371*	*0.16°*	*236*	*0.23°*
$100\,\mu m$	**881**	**0.28°**	**899**	**0.29°**	**560**	**0.36°**
$\pm 2\,pixels$	*647*	*0.23°*	*703*	*0.24°*	*264*	*0.25°*
$100\,\mu m$	–		–		**862**	**0.56°**
$0.01°$					*400*	*0.38°*

3.3.4.6 Conclusions

In this part, for a competent control performance of a parallel robot, the control spaces have been explored regarding a specific inverse dynamic model expressed in leg orientations. The prevailing results are brought by the LCS-CTC. This outcome suggests the following 3 important formative points in order to improve the performance of parallel robots:

- (i) the control space should be in the operational space of the robot;
- (ii) the control space should be also as close as possible to the sensor space;
- (iii) the models should be linearly and compactly expressed by the measurements of the sensor space;

The above conclusions once more clearly distinguish « vision » as one of the best options in the sense of allowing us to satisfy all of the 3 formative points at the same time.

Finally, we would like to remark that the presented versatile computed-torque control scheme meets the third objective of the integrated dynamic MICMAC part which is stated as at the end of the Chapter 1 as follows: « Proposing a vision-based framework for the dynamic control of parallel robots from their leg observations. »

3.4 Conclusions

We can give a brief summary of this chapter as follows:

- Firstly, we presented a vision based high-speed dynamic state observer. This observer can provide all the necessary variables which keep the modeling and as well as the control in linear form. It computes the position and the velocity of a parallel robot simultaneously, because it uses sequential observation information of the legs of a parallel robot which encodes the state of motion. It is fast, because it observes small portions of the legs with relatively small sub-images, and because it uses a single-iteration virtual visual servoing to compute the position and the velocity.
- Secondly, we proposed a versatile computed-torque control scheme based on the leg observations of a parallel robot. This control scheme allowed us easily to define control laws for different control spaces. Then, we explored the effects of error regulations in different control spaces and as a consequence we discovered some formative points of better control of parallel robots.

Correctness of these new theories are validated by simulations. Now, in the next Chapter, it is time to prove the feasibility of these theories on the real experimental setups.

Chapter 4

Experiments

This chapter experimentally validates the feasibility of the presented theories for dynamic modeling, for high-speed dynamic state estimation and for dynamic control of parallel robots.

4.1 High-Speed Integrated Dynamic MICMAC Observer

The high-speed dynamic state tracking algorithm which is described in Chapter 3 Section 3.2 will be tested on the Quattro parallel robot. First, we will give the details of the test-bed, then we will explain an experimental scenario.

4.1.1 Test-Bed Setup

Figure 4.1 shows test-bed of the Quattro parallel robot. In this test-bed the following points are important to note:

Choosing a camera In order to do high-speed estimation, camera frame rate should be fast enough. A Photon Focus CMOS CamLink TrackCam is a relevant sensor for our state estimation algorithm. It allows for fast sequential sub-image acquisition. One should also choose a short focal lens for better perspective effect on the cylindrical legs of the Quattro robot. The more perspective effect there is, the better the estimation is. Unfortunately, short focal lens brings distortions on the image which should be carefully taken care of.

Positioning of the camera For a good observation of the legs, the camera should be located somewhere far away from occlusions. In the Quattro robot, the camera is placed onto the robot base, looking downwards to the legs and to the end-effector. In this location, the field of view of the camera is less cluttered than the space outside the legs. Figure 4.1 shows the base-mounted camera of the Quattro robot and an image taken by this camera.

Lighting If fast acquisition is desired, CMOS sensor of the camera should be exposed for a short time. This short exposure time causes dark images. Subsequently, to have clear images, the scene must be very strongly and properly illuminated. That is to say, a good visibility of the legs must be guaranteed while keeping the image noise at a moderate level. For instance, strong lighting can produce reflections causing false edges on the legs. A solution might be a white back light which allows one to observe the shadows

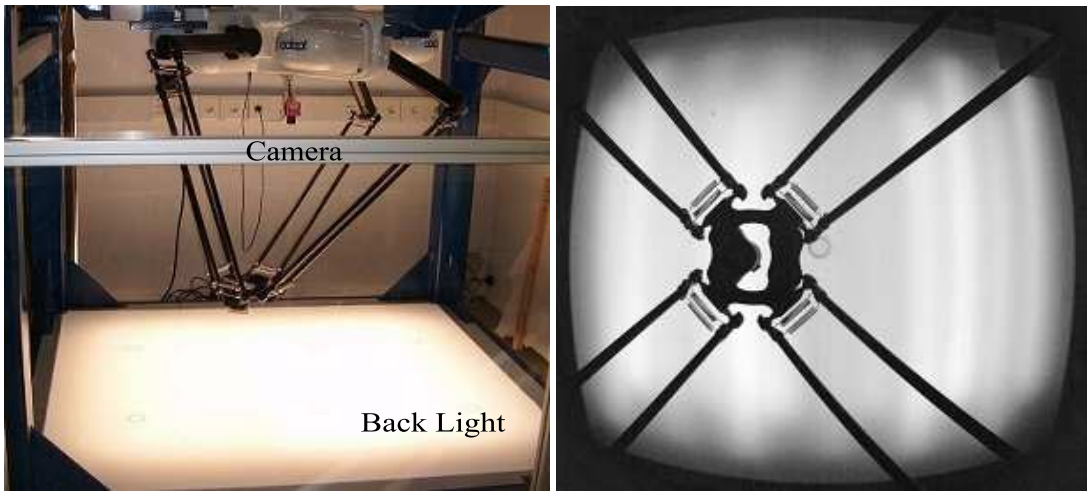


Figure 4.1 – *Left picture*: The Quattro robot and its cell. The camera is mounted onto the base looking downwards to the legs. The back light platform is below the legs. *Right picture*: An image of the legs from the base-mounted camera.

(i.e., silhouettes) of the legs on the image. This lighting provides almost a binary image where many of the unwanted false edges are removed. On the other hand, the white back light limits the visibility of the scene to its lighting surface. This also implies that the back light can limit the workspace to a smaller region than the field of view of the camera.

Camera-robot calibration Intrinsic calibration of the camera should be accurate. This must be performed with a method which can give a direct model for lens distortion correction (e.g., Visp) rather than an indirect (iterative) model. A direct model accelerates the state estimation. Afterwards, the pose of the camera frame with respect to robot base frame (i.e., extrinsic calibration) $[R, \mathbf{t}]$ should be calculated accurately. The accuracy and the speed of the estimation depend on all these parameters.

Synchronization In order to have a ground truth to compare the results, the motor encoders of the robot and the camera are synchronized to capture an information at the same time. So, motor encoders read the joint positions at the moment when the camera grabs a sub-image of a leg.

4.1.2 Experimental Scenario

To validate the feasibility of the dynamic state tracking algorithm, we proceed as follows:

1. *Calibration*: We shall first calibrate the extrinsic pose parameters of the camera with respect to the robot base frame. This extrinsic pose of the camera will be calculated by using the visual edges of the lower-legs of the robot.
2. *Reference Trajectory*: We will design a Cartesian space trajectory which expands all axes of motion of the robot, and we will calculate a region of interest (ROI) image trajectory of the lower-legs which corresponds to this designed reference Cartesian space trajectory.

3. *Data Collection*: The Quattro robot will be moved along the reference Cartesian space trajectory with simple Cartesian space kinematic control. During this motion, we will synchronously save the joint values measured by the motor encoders and as well as the sub-images of the lower-legs grabbed by the camera.
4. *Off-Line Dynamic State Computation*: We will then compute the dynamic state of the Quattro parallel robot and the ROI predictions through the registered sub-images of the lower-legs.
5. *Comparison*: Finally, we will compare calculated results with the performed reference motion of the Quattro parallel robot.

4.1.3 Calibration

Here, we shall calculate the pose (${}^{c}R_{o},\ {}^{c}\mathbf{t}_{o}$) of the camera frame with respect to the robot's base frame. It is assumed that the camera intrinsic matrix (K), the lens distortion correction coefficients, and the geometric parameters (ξ_{geo}) of the robot are known.

4.1.3.1 Camera-Quattro Parallel Robot Calibration from Leg Edges

In order to find the pose of the camera frame with respect to the base frame of the Quattro robot, we exploited a geometric constraint of the cylindrical legs as an objective function for minimization. The geometric constraint is as follows:

$$ {}^{c}\underline{\mathbf{n}}_{ij}{}^{T}\ {}^{c}\mathbf{B}_{i} \ = \ -\,r \tag{4.1} $$

where $\underline{\mathbf{n}}$ is a visual side edge of a cylindrical leg, $\mathbf{B}$ is the connection point of this cylindrical leg to the nacelle, r is the radius of the cylindrical leg, $i \in \{1, 2, \ldots, 8\}$ is the cylindrical leg index, and $j \in \{L,\ R\}$ is the left or right side visual edge index. Figure 4.2 depicts an image of the Quattro robot with its 16 edges of the 8 cylindrical legs. Therefore, from an image of a known robot posture, we can write 16 constraint equations.

Constraint (4.1) can be rewritten with the pose parameters of the camera frame as below:

$$ {}^{c}\underline{\mathbf{n}}_{ij}{}^{T}\ \left({}^{c}R_{o}\,{}^{o}\mathbf{B}_{i} \ + \ {}^{c}\mathbf{t}_{o} \right) \ = \ -\,r \tag{4.2} $$

where this time the connection point $\mathbf{B}$ is expressed in the robot's base frame, and it is known from the 3D CAD model of the robot. We rewrite (4.2) for the unknown pose parameters as follows:

$$ A_{ij}\ \mathbf{x} \ = \ -r \tag{4.3} $$

where the coefficient row vector $A_{ij} \in \Re^{1\times 12}$ and the unknown parameters column vector $\mathbf{x} \in \Re^{12\times 1}$ are as below:

$$ A_{ij} \ = \ {}^{c}\underline{\mathbf{n}}_{ij}{}^{T} \ \begin{bmatrix} {}^{o}\mathbf{B}_{i}^{T} & \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} & 1 & 0 & 0 \\ \mathbf{0}_{1\times 3} & {}^{o}\mathbf{B}_{i}^{T} & \mathbf{0}_{1\times 3} & 0 & 1 & 0 \\ \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} & {}^{o}\mathbf{B}_{i}^{T} & 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{x} \ = \ \begin{bmatrix} \mathbf{r}_{1}^{T} \\ \mathbf{r}_{2}^{T} \\ \mathbf{r}_{3}^{T} \\ {}^{c}\mathbf{t}_{o} \end{bmatrix} \tag{4.4} $$

Figure 4.2 – The 16 edges (red lines) from 8 cylindrical legs of the Quattro robot for a given pose in calibration.

and where $\mathbf{r}_{1,2,3} \in \Re^{1\times 3}$ are the row vectors of the orientation matrix ${}^{c}R_{o}$. Stacking all the linear constraint systems of all the observed cylindrical legs computed from k images, one can write the following complete system:

$$A\,\mathbf{x} \;=\; \mathbf{b} \tag{4.5}$$

where $A \in \Re^{16k\times 12}$ and $\mathbf{b} \in \Re^{16k}$ are as follows:

$$A \;=\; \begin{bmatrix} {}^{1}A_{1L} \\ {}^{1}A_{1R} \\ \vdots \\ {}^{k}A_{8R} \end{bmatrix}, \qquad \mathbf{b} \;=\; \begin{bmatrix} -r \\ \vdots \\ -r \end{bmatrix} \tag{4.6}$$

Linear solution of (4.5) yields the extrinsic pose parameters of the camera frame with respect to base frame of the robot. However, the computed orientation matrix ${}^{c}R_{o}$ may not be orthogonal. One can make it orthogonal using its SVD decomposition:

$${}^{c}R_{o} \;=\; U\,D\,V^{T} \tag{4.7}$$

assuming that the diagonal D matrix is identity. Then, the new orthogonal orientation matrix is calculated as below:

$${}^{c}R_{o} \;=\; U\,V^{T} \tag{4.8}$$

While calibrating the camera frame with respect to the robot frame, we have taken images of 25 different known poses of the robot expressed in its base frame with respect to its base

Table 4.1 – Calibration residual errors of the geometric constraint given in (4.1).

	Linear solution	Non-linear solution
RMSE	0.0022 m	0.0008 m

frame. We solved the extrinsic parameters through the linear system explained above. In order to compare the correctness of the results, we also solved the constraint equations written in (4.2) through the non-linear "trust-region-reflective" minimization method [CL96]. Table 4.1 depicts the root mean squares (RMSE) of the constraint residuals of the linear and non-linear solutions. When two solutions (i.e., linear and non-linear) of orientation matrices are compared with the geodesic distance metric, the difference is found $0.0016\,rad$. When two solutions of position vectors are compared with the Euclidean distance metric, the difference is found $0.0017\,m$. Figure 4.3 shows the 3 robot posture images with the back-projection of the connection points $\mathbf{B}$. Back-projection is performed with the extrinsic camera pose parameters which are computed linearly.

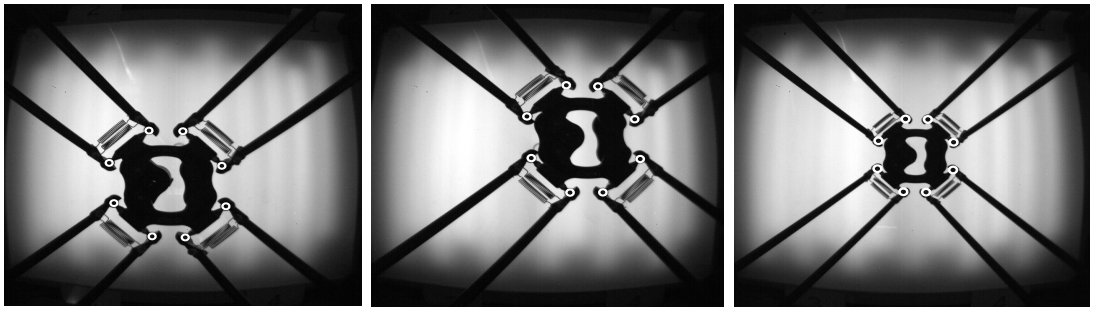


Figure 4.3 – Back-projection (white circles) of connection points $\mathbf{B}$ with linearly computed extrinsic pose parameters.

4.1.3.2 Coarse to Fine Calibration

One may start with a coarse set of the extrinsic parameters $[\,\widehat{R},\,\widehat{\mathbf{t}}\,]$ in order to perform the dynamic state estimation of the robot:

$$\widehat{R} \,=\, \Delta R\; R\,, \qquad \widehat{\mathbf{t}} \,=\, \mathbf{t} \,+\, \Delta\mathbf{t} \tag{4.9}$$

where $[\,R,\,\mathbf{t}\,]$ are the correct extrinsic parameters, and $[\,\Delta R,\,\Delta\mathbf{t}\,]$ are the errors of these parameters. As a consequence, the dynamic state estimations will give erroneous results. However, since the errors $[\,\Delta R,\,\Delta\mathbf{t}\,]$ of the extrinsic parameters of the camera stay constant during the dynamic state estimation of the robot, these errors will show themselves as an offset between the performed trajectory by the real robot and the estimated trajectory by the virtual robot. A solution for this offset can be approximated by a 3D pose estimation between these two space trajectories. Let $\{\mathbf{P}_1,\,\ldots,\,\mathbf{P}_m\} \in \Re^{3\times 1}$ and $\{\mathbf{P}^*_1,\,\ldots,\,\mathbf{P}^*_m\} \in \Re^{3\times 1}$ be two sub-sets of point correspondences in the estimated and performed trajectories, respectively. Hence, in order to

calculate this offset, the objective function for minimization is written as follows:

$$\min_{(\Delta R,\, \Delta \mathbf{t})} \sum_{i=1}^{m} \| \mathbf{P}_i^* - \Delta R\, \mathbf{P}_i - \Delta \mathbf{t} \|^2, \qquad m \geqslant 3 \tag{4.10}$$

where m is the minimum number of non-collinear corresponding points. Once this offset is computed, one can update the extrinsic parameters as follows:

$$R = (\Delta R)^T \widehat{R}, \qquad \mathbf{t} = \widehat{\mathbf{t}} - \Delta \mathbf{t} \tag{4.11}$$

where R and $\mathbf{t}$ are the new approximated parameters which will yield better results. So, this dynamic state estimation algorithm can also serve for a coarse to fine camera-robot calibration.

4.1.4 Reference Trajectory

The reference Cartesian space motion ($\mathbb{X}^*$) is a $0.08\,m$ by $0.08\,m$ square trajectory on the xy-plane. There is no rotation in this trajectory. The maximum velocity and acceleration of the motion are $0.25\,m/s$ and $1\,m/s^2$, respectively. This square trajectory is rotated by 60° degrees around the x-axis so as to cross the 3 axes of the motion space. Afterwards, a region of interest (ROI) image trajectory is created for the observation of the lower-legs of the Quattro robot using this reference Cartesian space motion ($\mathbb{X}^*$). The image trajectory contains the upper-left corner pixel coordinates of the sub-images that shall be acquired during the reference square motion of the robot. The size of sub-images are 48×48 pixel^2. Figure 4.4 shows this reference sub-image (ROI^*) trajectory of the lower-legs of the Quattro robot.

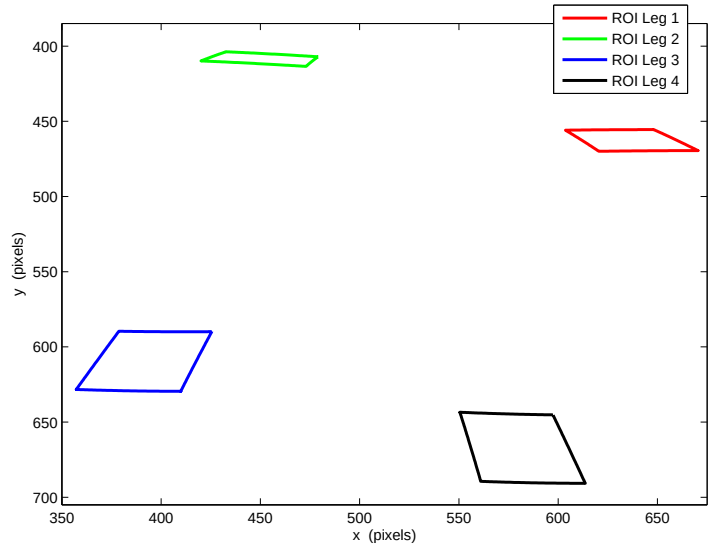


Figure 4.4 – Reference ROI^* upper-left corner positions on the image plane. Red, green, blue and black traces belong to the first, second, third and forth lower-legs, respectively.

4.1.5 Data Collection

The Quattro parallel robot is moved along the reference square trajectory through a simple Cartesian space kinematic control. During this motion, we recorded at $500\,Hz$ the joint positions ($\mathbf{q}^m$) of the motors given by the encoders and as well as the sub-images grabbed synchronously by the camera. Figure 4.5 shows the block diagram of this kinematic control and synchronous data acquisition. These measured joint positions ($\mathbf{q}^m$) are then used to calculate

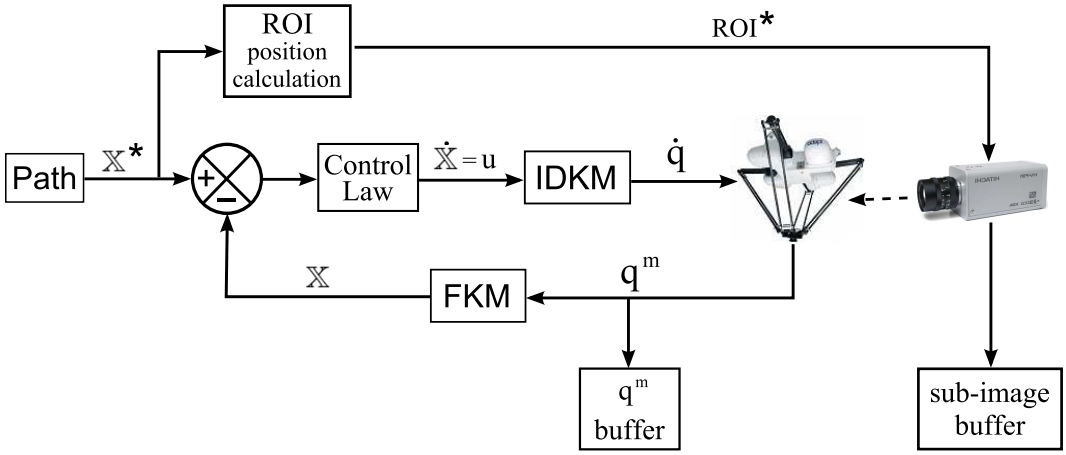


Figure 4.5 – Cartesian space kinematic control with synchronous data acquisition. FKM and IDKM are the iterative forward kinematic model and the inverse differential kinematic model of the Quattro robot, respectively.

the Cartesian space end-effector poses ($\mathbb{X}$) via the iterative forward kinematic model (FKM) of the Quattro parallel robot:

$$\mathbb{X} = FKM(\mathbf{q}^m, \mathbb{X}_o) \tag{4.12}$$

where $\mathbb{X}_o$ is an initial guess for the end-effector pose. Table 4.1.5 gives the pseudo-code of this iterative FKM.

$\mathbb{X} = \mathbb{X}_o$ // start with an initial pose
do{
$\mathbf{q} = IKM(\mathbb{X})$
$\Delta\mathbf{q} = \mathbf{q}^m - \mathbf{q}$
$\mathbb{X} = \mathbb{X} + M_{\mathbf{q}}^{-1}\,\Delta\mathbf{q}$
}while ($\|\Delta\mathbf{q}\| > \epsilon$)

Table 4.2 – Pseudo-code for the forward kinematic model (FKM) of the Quattro robot. $M_{\mathbf{q}}$ is the inverse differential kinematic model (IDKM) between the joint velocities and the end-effector pose velocity ($\dot{\mathbf{q}} = M_{\mathbf{q}}\,\dot{\mathbb{X}}$). IKM is the inverse kinematic model which relates Cartesian end-effector pose to the joint positions. ϵ is a threshold for the desired precision of the computed Cartesian end-effector pose. $\mathbf{q}^m$ and $\mathbf{q}$ are the measured and the computed joint positions, respectively.

Assuming that the known geometric parameters of the Quattro parallel robot are perfect, these computed poses ($\mathbb{X}$) will be used as ground truth (i.e., reference motion) for comparison. Figure 4.7 shows traces of the calculated Cartesian end-effector poses ($\mathbb{X}$) of the Quattro parallel robot which are expressed in its base frame. Figure 4.8 depicts velocities and accelerations of this performed square motion ($\mathbb{X}$). Velocities and accelerations are obtained by numerical differentiation of the Cartesian end-effector poses. Figure 4.6 shows simple block diagram for the computation of the ground truth states $\{\mathbb{X}, \dot{\mathbb{X}}, \ddot{\mathbb{X}}\}$.

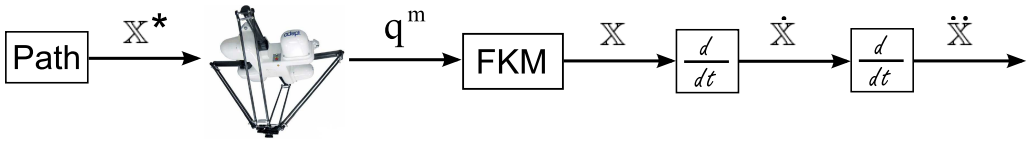


Figure 4.6 – The ground truth end-effector pose ($\mathbb{X}$), end-effector pose velocity ($\dot{\mathbb{X}}$) and end-effector pose acceleration ($\ddot{\mathbb{X}}$) generation.

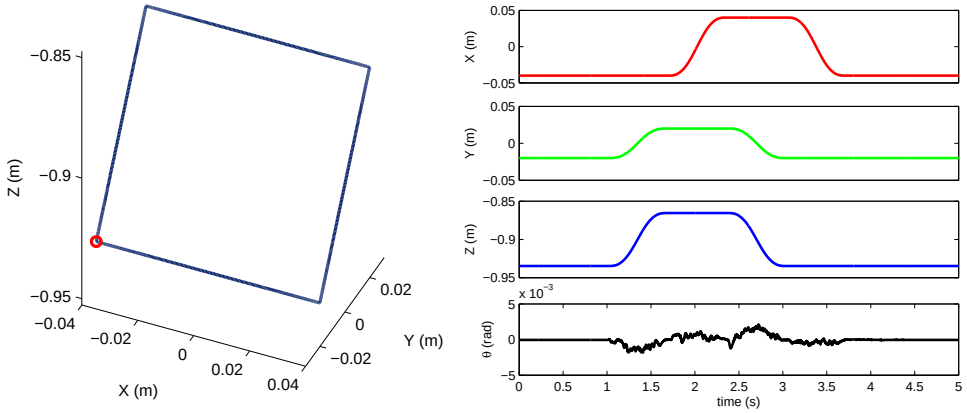


Figure 4.7 – Performed square motion ($\mathbb{X}$) during kinematic control. *Left figure*: 3D Cartesian end-effector pose trajectory with starting and ending point (red circle). *Right figure*: Evolution of the Cartesian end-effector poses versus time.

4.1.6 Off-Line Dynamic State Computation

The estimation algorithm is coded in C++ with nT_2 matrix library [FLCS07]. In order to validate the dynamic state estimation algorithm, we first detected and extracted the edge pixels (i.e., contours) of partially observed cylindrical legs from the registered sub-images. This edge extraction is performed with Canny edge-detection method [Can86]. Figure 4.9 shows a sequence of sub-images of the lower-legs of the Quattro parallel robot with their detected edge pixels. Finally, we conducted the dynamic state estimation algorithm with these detected edge pixels which are transformed to metric units. Figure 4.10 simply illustrates this dynamic state estimation process. Table 4.3 tabulates the accuracies of the position and orientation estimations. These accuracies are calculated with the root mean squares of the tracking errors (RMSE)

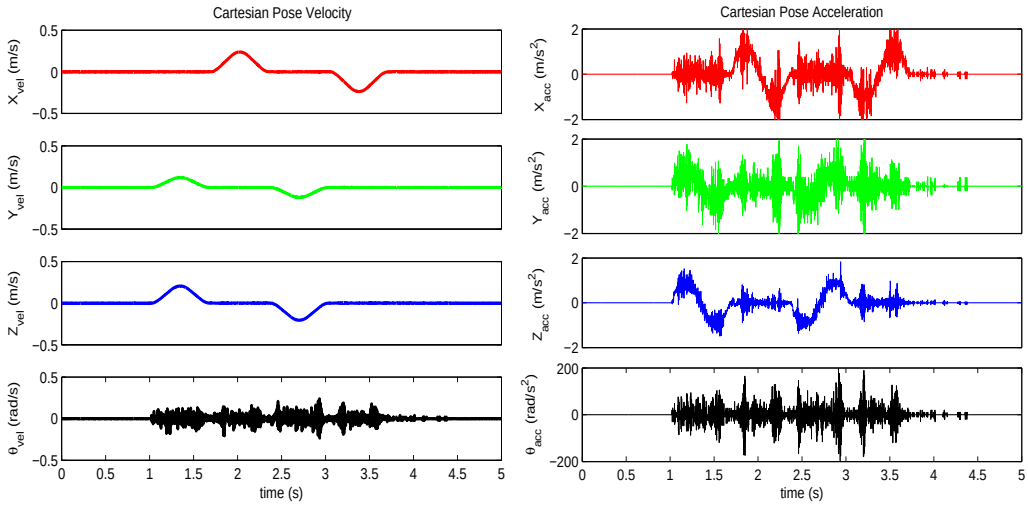


Figure 4.8 – *Left figure*: Cartesian end-effector pose velocity ($\dot{\mathbb{X}}$) prints of the square motion. *Right figure*: Cartesian end-effector pose acceleration ($\ddot{\mathbb{X}}$) prints of the square motion.

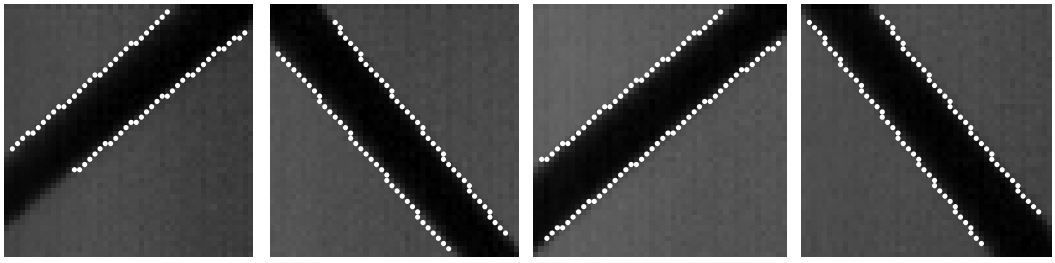


Figure 4.9 – From left to right, sequentially grabbed (48×48 pixel^2) sub-images of the lower-legs 1, 2, 3 and 4, of the Quattro parallel robot. White dots are the detected edge pixels for the dynamic state estimation algorithm. In this figure, the sub-images are zoomed.

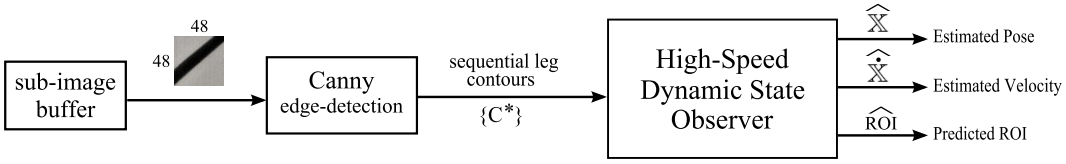


Figure 4.10 – High-speed dynamic state estimation scheme.

and Hausdorff distance metric regarding to ground truth trajectories. Table 4.4 tabulates the approximate average times taken for each of the processes used in a single dynamic state estimation of the Quattro parallel robot. These processes are, in turn, the exposure of ROI region on the CMOS sensor, the transfer of the ROI pixel information, the detection of the edge pixels of a cylindrical leg in this ROI, and the computation of the current dynamic state of the robot. Consequently, an estimation takes about $1400\,\mu s$ (microseconds) which comfortably allows to discover the current posture and velocity of the robot more than $500\,Hz$. To our best know-

Table 4.3 – Dynamic state estimation errors.

	Pose Errors		Velocity Errors	
	$xyz\ (m)$	$\theta\ (rad)$	$\dot{xyz}\ (m/s)$	$\dot{\theta}\ (rad/s)$
RMSE	0.004	0.027	0.098	0.31
Hausdorff	0.007	0.048	0.267	0.88

ledge, this method also implies the first proposed vision-based dynamic motion estimation of an articulated object at high speed by sequential sub-image acquisition.

Table 4.4 – Times taken for an estimation with 48×48 sub-images.

	ROI exposure	ROI transfer	Edge detection	Estimation
Time (μs)	500	100	200	600

4.1.7 Comparison

Figure 4.11 depicts the reference ($\mathbb{X}$) and the estimated ($\widehat{\mathbb{X}}$) Cartesian space curves.

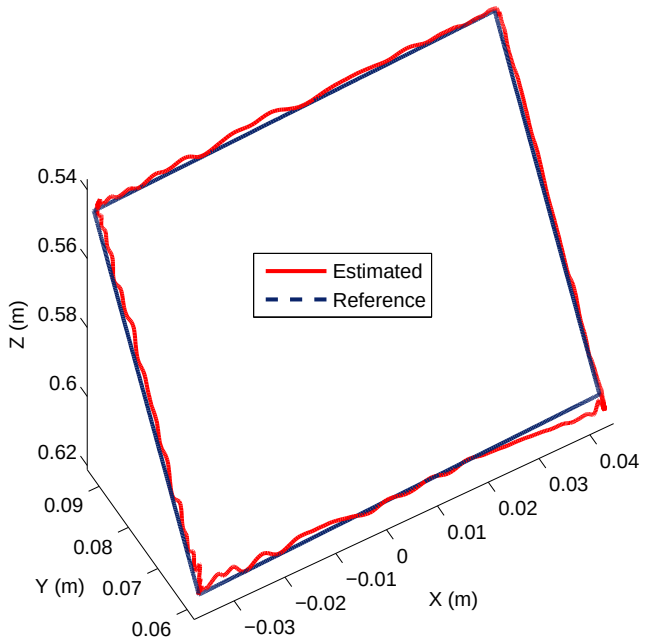


Figure 4.11 – Superimposed estimated ($\widehat{\mathbb{X}}$) and reference ($\mathbb{X}$) 3D trajectories expressed in the camera frame. Red solid line is the estimated trajectory and blue dashed line is the reference trajectory.

In Figure 4.11, the estimated 3D trajectory ($\widehat{\mathbb{X}}$) fits rather well to the reference ($\mathbb{X}$) trajectory. Note that in reality, the reference ($\mathbb{X}$) trajectory might differ from the real performed one by the robot, because the numerical FKM may not perfectly fit to the mechanics of the robot. The error between the reference ($\mathbb{X}$) and the estimated ($\widehat{\mathbb{X}}$) Cartesian poses is calculated as below:

$$\mathbf{e}_{\mathbb{X}} = \mathbb{X} - \widehat{\mathbb{X}} \tag{4.13}$$

Figure 4.12 plots these positional and orientational estimation errors versus time. In Figure 4.12, we observe that the estimated trajectory is not very smooth in the depth (z-axis) and as well as in the orientation (θ). Figure 4.13 plots the estimated ($\dot{\widehat{\mathbb{X}}}$) and the reference ($\dot{\mathbb{X}}$)

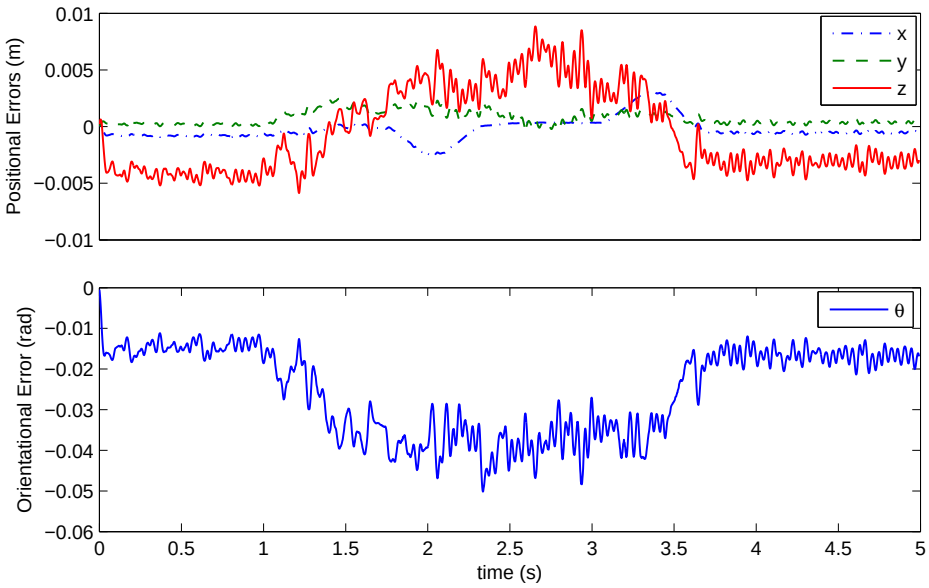


Figure 4.12 – Cartesian pose estimation errors ($\mathbf{e}_{\mathbb{X}}$) versus time for the Figure 4.11.

Cartesian velocities of the end-effector versus time. Estimated velocities look quite good for the x and y axes while they are very noisy for z-axis and θ. These noisy estimations of velocity in the depth and orientation can be directly deduced from the results of Figure 4.12. Figure 4.14 shows the superimposed reference (ROI^*) and predicted ($\widehat{ROI}$) sub-image upper-left corner position trajectories of the lower-legs of the Quattro parallel robot. Figure 4.15 plots the prediction errors of these sub-image positions on each of the lower-legs. A ROI prediction error is calculated as Euclidean distance between the reference and the predicted sub-image upper-left corner positions. The maximum prediction error for a sub-image position is calculated as 4.2 pixels through the whole tracking process. This maximum prediction error of 4.2 pixels for a 48×48 pixel^2 sub-image corresponds to a 6% error on the diagonal length of the sub-image. These predictions are quite enough to guarantee the observability of the edges during the motion of the Quattro parallel robot.

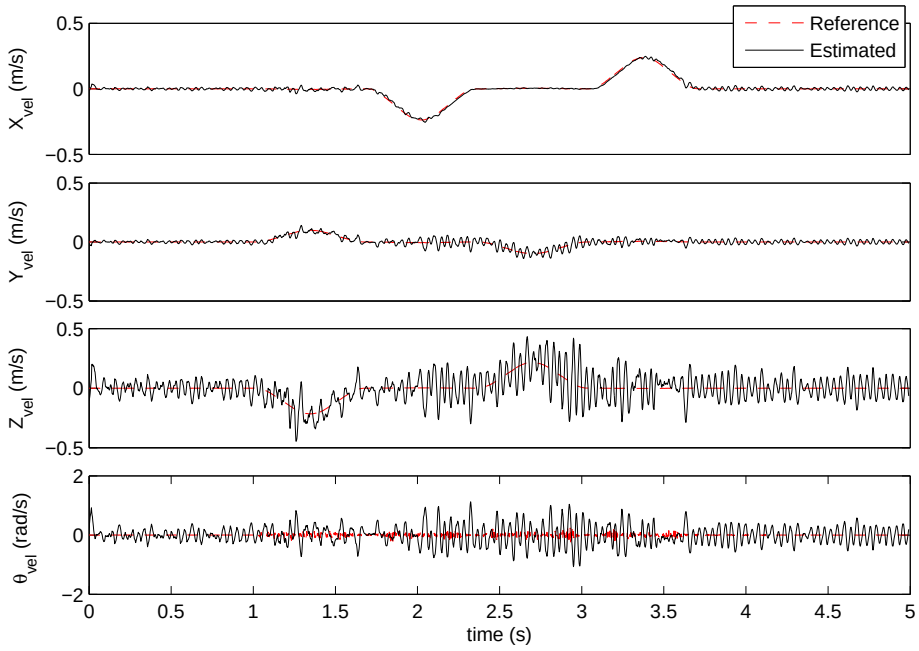


Figure 4.13 – Superimposed estimated ($\hat{\dot{\mathbb{X}}}$) and reference ($\dot{\mathbb{X}}$) Cartesian pose velocities versus time for the Figure 4.11.

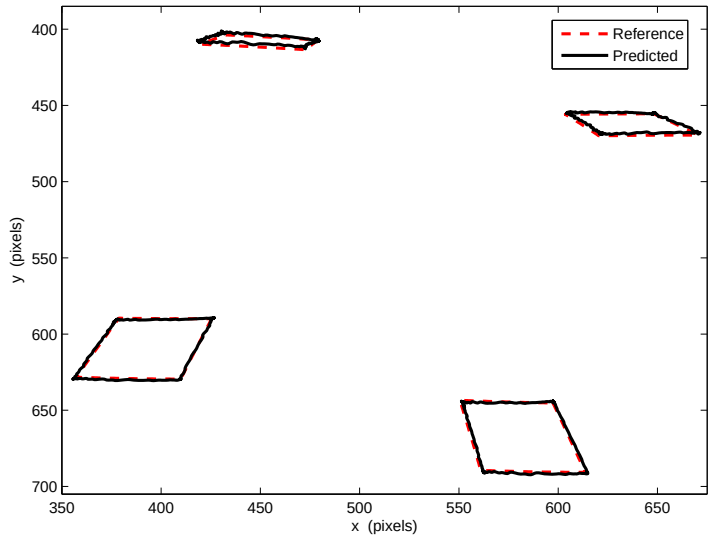


Figure 4.14 – Superimposed reference (ROI^*) and predicted ($\widehat{ROI}$) sub-image upper-left corner position trajectories on the image plane. Red dashed line is the reference and black solid line is the predicted.

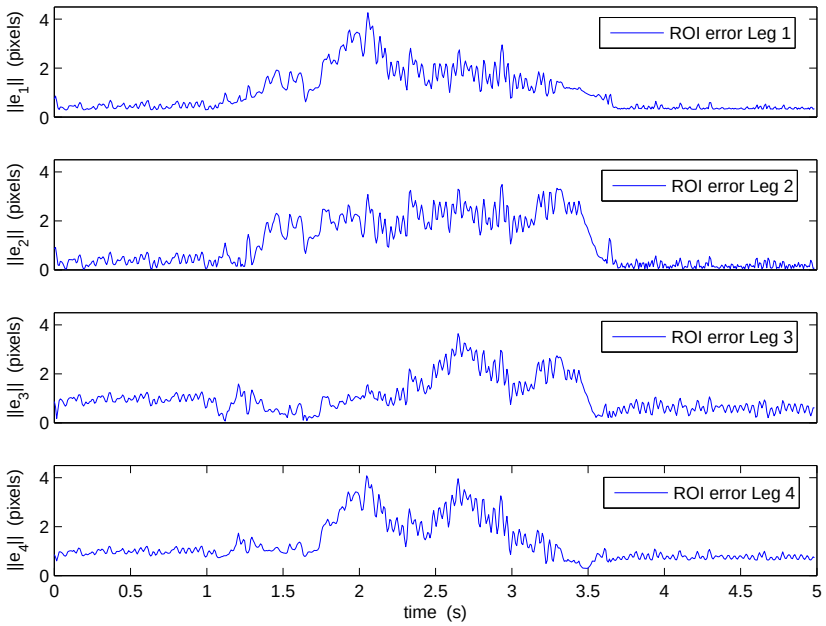


Figure 4.15 – ROI prediction errors on each of the observed legs of the Quattro parallel robot. A prediction error for a ROI is computed as the Euclidean distance between the reference ROI position and the predicted ROI position. The maximum prediction error is calculated as 4.2 pixels through the whole tracking process.

Discussion: This estimation method does not need a special artificial pattern as the legs are observed, and it is applicable to any parallel robot with slim prism-shaped legs. It is feasible by an edge detection in a quite small and well structured sub-image. The sub-image contains only a partial region of a slim leg. When the dynamic state estimation results are inspected, one can say that the depth (z-axis) and the orientation (θ) estimations are quite noisy (see Figs. 4.12 and 4.13). The errors, in the depth and the orientation estimations, might appear because of a cylindrical leg whose radius (r) is relatively smaller than its observational distance (d_o) from the camera. This makes estimations, especially the depth, more sensitive to small noises:

$$\tilde{z} = \epsilon \frac{d_o}{r} \tag{4.14}$$

where $\tilde{z}$ is the ambiguity in the depth and ϵ is a small observation error on the radius of a cylindrical leg ($\epsilon \ll r$). On the image:

- (i) if one observes a leg thinner ($r - \epsilon$) than it is, the depth of this leg is calculated farther from the camera;
- (ii) If one observes a leg thicker ($r + \epsilon$) than it is, the depth of this leg is calculated closer to the camera;

Subsequently, different depth errors on each of the legs also create orientation errors. Figure 4.16 illustrates this depth ambiguity.

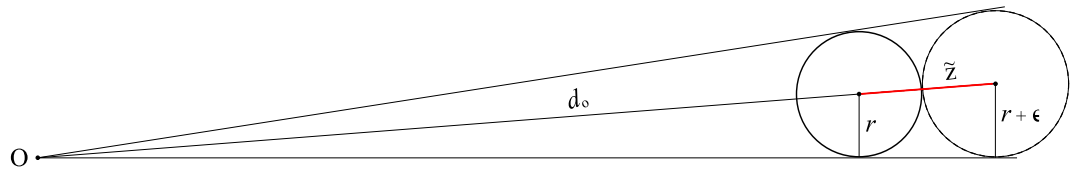


Figure 4.16 – A cylinder whose radius (r) is relatively smaller than its observational distance (d_o) from the camera. A small (ϵ) noise on the measured thickness of the cylinder on the image creates ambiguity on the depth of the revolution axis of the cylinder with respect to the optical center **O** of the camera. The red line shows this ambiguity ($\tilde{z}$) on the depth.

For example, the radius of a cylindrical leg of the Quattro robot is $r = 0.0078\,m$ and the approximate observation distance of the legs during this reference square motion was $d_o = 0.4\,m$. Hence, considering an image noise corresponding to $\epsilon = 0.0001\,m$ ambiguity on the radius of the observed legs causes approximately $0.005\,m$ depth errors in the estimations.

In our experimental test-bed, the source of this image noise can appear because of back lighting from a surface source. In the case of a point-source front light, the visible part of the cylindrical leg is the lighted region. On the other hand, in the case of a surface-source back light, the visible part of the cylindrical leg is the unlighted region and this unlighted region can be thinner than it is in front lighting. This means that it is possible to calculate the depth of the cylindrical legs farther than they are. We can observe this conclusion in Figure 4.12 from 0 to 1 seconds and from 4 to 5 seconds. In these time intervals, the robot is in a waiting pose (i.e., static) before the motion and after the motion, therefore the back light effect is homogenous on every leg which yields farther estimated positions. When the robot moves, the back light effect is irregular since the sub-images are taken sequentially at different time instants. Thus, this irregular back light effect causes different depth errors on each of the leg and as a consequence worse orientation estimations appear from 1 to 4 seconds. Figure 4.17 illustrates the effect of a surface-source back lighting on the observable thickness of a cylindrical leg.

Generally speaking, the source of the errors comes: (i) from the use of the approximated theoretical models; (ii) from the calibration of the camera extrinsic parameters; and (iii) from the image noise which creates false edges.

Future perspectives: In order to increase the accuracy and the speed of the estimations, the following perspectives are considered: (i) an investigation of the grabbing strategy with different number, size, location, and order of the sub-images; (ii) an exploration of the number of rods (4 or 8) and the number of sub-images on each cylindrical leg; (iii) an implementation of the edge detection algorithm on a smart camera which has an embedded FPGA / DSP / ARM hardware; and (iv) an investigation of the proper lighting of the legs;

Once all these problems are solved, one can also imagine: (i) robust tracking of the leg contours (e.g., treatment of occlusions); (ii) efficient implementation of the algorithm (e.g., reducing computation time); (iii) generalization to other parallel robots; and (iv) industrial applications.

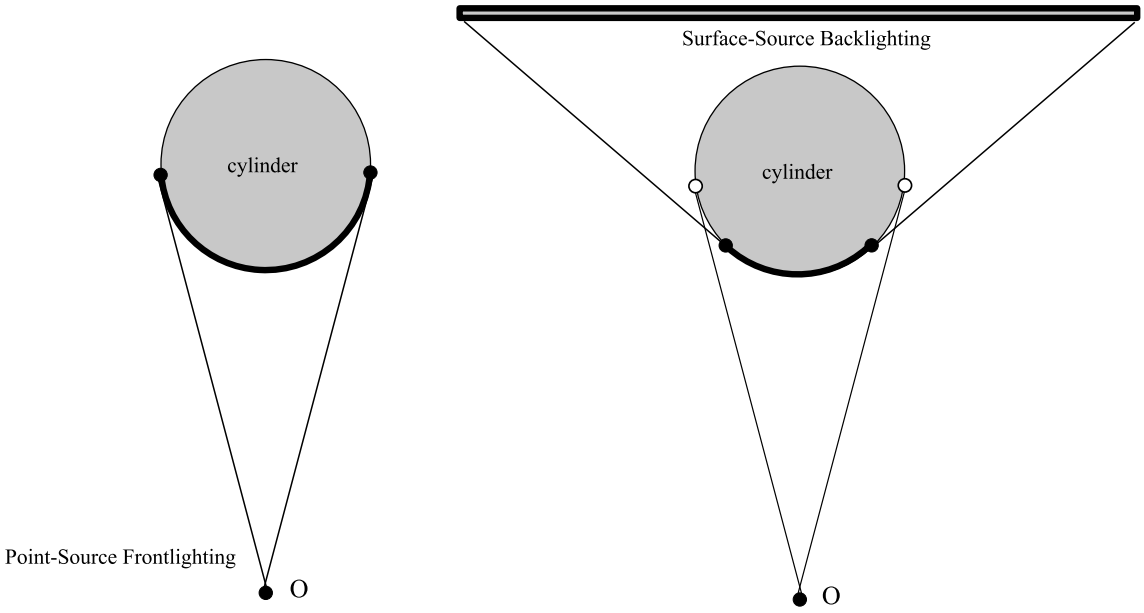


Figure 4.17 – Lighting effects and visibility of a cylindrical leg. **O** is the optical center of the camera. *Left*: Front lighting from a point source which is assumed at the proximity of the **O**. In the case of a point-source front light, the visible part of the cylinder is the lighted region (bold arc). *Right*: Back lighting from a surface source. In the case of a surface-source back light, the visible part of the cylinder is the unlighted region (bold arc) and this region is now thinner than it is in front lighting.

4.2 Inverse Dynamic Model

4.2.1 Validation of the Inverse Dynamic Model

The proposed inverse dynamic model (IDM) of the Quattro parallel robot, which is written in terms of the unit orientation vectors of the legs as explained in Chapter 2:

$$\mathbf{\Gamma} = -A^{\dagger}(\underline{\mathbf{x}})\ \mathbf{b}(\ddot{\underline{\mathbf{x}}}, \dot{\underline{\mathbf{x}}}, \underline{\mathbf{x}}) \tag{4.15}$$

is validated comparing to a reference inverse dynamic model presented in [NKC^{+}08].

A Reference Inverse Dynamic Model for the Quattro Parallel Robot

In [NKC^{+}08], a simplified inverse dynamic model is proposed for the LIRMM Par4 parallel robot, and it is shown that this simplified inverse dynamic model is almost correct as the complete dynamic model of the Par4 parallel robot. The correctness of this simplified inverse dynamic model is validated by simulations in Adams software and by experimentations on the LIRMM Par4 parallel robot. The Par4 parallel robot is the “father" prototype of the Quattro parallel robot. That is to say, the Quattro and the Par4 are architecturally the same parallel robots. The only difference in two robots is the slight variations in lengths and in weights of the

kinematic elements. Thus, we took this simplified inverse dynamic model as a reference inverse dynamic model for the Quattro parallel robot in order to compare it with our proposed inverse dynamic model. In this reference model, the simplifications are as follows: (i) the weight of a parallelogram lower-leg is considered as two point masses at each extremity; (ii) the inertia of two rotating rods of nacelle are neglected; and (iii) the weight of these two rotating rods are considered as two point masses at each extremity.

This reference IDM of the Quattro robot is written based on the motorized joint positions ($\mathbf{q}$) and poses ($\mathbb{X}_1$ and $\mathbb{X}_2$) of the mass centers of the two translational bars of the nacelle. The formulation of the reference IDM is as follows:

$$\mathbf{\Gamma}_{ref} = \mathcal{I}_{act}\,\ddot{\mathbf{q}} + J_1^T\,\mathcal{M}_1\,(\ddot{\mathbb{X}}_1 + \mathbf{g}) + J_2^T\,\mathcal{M}_2\,(\ddot{\mathbb{X}}_2 + \mathbf{g}) - \cos(\mathbf{q})\,g\,(\mathcal{M}_3\,\frac{\ell}{2} + \mathcal{M}_4\,\ell) + \mathbf{\Gamma}_f \quad (4.16)$$

where $\mathcal{I}_{act}$ is the inertia matrix which contains the inertias of the actuators, upper-legs and point mass lower-legs; J_1 and J_2 are the robot Jacobians written for each of the two poses of the translational bars of the nacelle; $\mathcal{M}_1$ and $\mathcal{M}_1$ are the mass matrices of these translational bars of the nacelle; $\mathbf{g} = [0\,0\,g]^T$ is the constant gravity vector; $\mathcal{M}_3$ and $\mathcal{M}_4$ are the mass matrices of the upper-legs and point mass lower-legs, respectively; ℓ is the length of an upper-leg; and $\mathbf{\Gamma}_f = f_v\,\dot{\mathbf{q}} + f_c\,sign(\dot{\mathbf{q}})$ is the friction term offering resistance on the the actuated joints, with f_v viscous and f_c Coulomb friction coefficient matrices.

Comparison

In comparison of the reference IDM and our proposed IDM of the Quattro parallel robot, we used the same geometric (ξ_{geo}) and dynamic (ξ_{dyn}) parameters in both of the models. The lengths of kinematic elements are obtained from the CAD model. The weights of the lower-legs and the moving platform are measured on a balance. The weights of the upper-legs and the inertias of the motors are obtained from Adept Company. Table 4.5 lists the length and weight of each of the identical kinematic elements of the Quattro parallel robot. All the required input

A Kinematic Leg

	$length$	$weight$
an upper-leg	$0.375\,m$	$1.5\,kg$
a lower-leg	$0.825\,m$	$0.48\,kg$

Nacelle

	$length$	$weight$
a translational-bar	$0.131\,m$	$0.555\,kg$
a rotational-bar	$0.06172\,m$	$0.555\,kg$

Table 4.5 – Geometric and dynamic parameters of the identical kinematic elements of the Quattro parallel robot. A motor inertia is obtained as $\mathcal{I}_m = 0.000043\,kg.m^2$ from Adept.

variables for comparison of the reference IDM and the proposed IDM are calculated from the measured joint positions ($\mathbf{q}^m$). The conversion between the variables is performed with the kinematic models of the Quattro robot and numerical differentiation with respect to time:

$$\begin{aligned}
&\mathbb{X} = FKM(\,\mathbf{q}^m\,), && \underline{\mathbf{x}} = IKM(\,\mathbb{X}\,),\\
&\dot{\mathbb{X}} = \tfrac{d}{dt}\,(\,FKM(\,\mathbf{q}^m\,)\,), && \underline{\dot{\mathbf{x}}} = IDKM(\,\dot{\mathbb{X}}\,), \qquad \dot{\mathbf{q}} = \tfrac{d}{dt}\,(\,\mathbf{q}^m\,)\\
&\ddot{\mathbb{X}} = \tfrac{d^2}{dt^2}\,(\,FKM(\,\mathbf{q}^m\,)\,), && \underline{\ddot{\mathbf{x}}} = IDKM_2(\,\ddot{\mathbb{X}}\,), \qquad \ddot{\mathbf{q}} = \tfrac{d^2}{dt^2}\,(\,\mathbf{q}^m\,)
\end{aligned} \quad (4.17)$$

where FKM is the forward kinematic model which relates joint positions to the end-effector pose; IKM is the inverse kinematic model which relates end-effector pose to the unit orientation vectors of the kinematic elements; $IDKM$ is the inverse differential kinematic model which relates velocity of the end-effector pose to the velocities of the unit orientation vectors of the kinematic elements; and $IDKM_2$ is the second-order inverse differential kinematic model which relates acceleration of the end-effector pose to the accelerations of the unit orientation vectors of the kinematic elements. Figure 4.18 depicts how comparison is made between the reference and proposed inverse dynamic models using the measured joint positions during a motion of the Quattro robot.

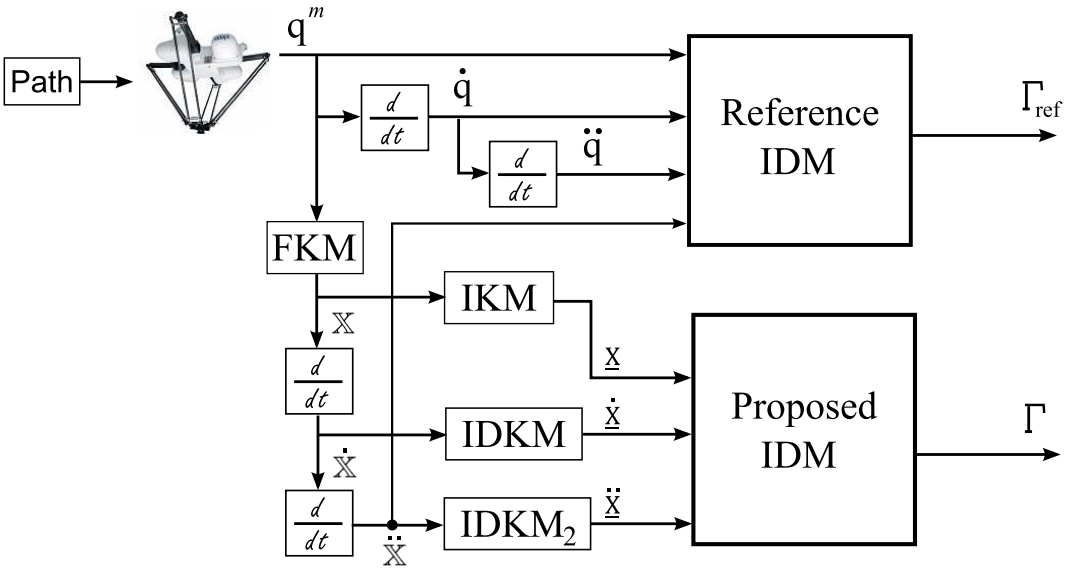


Figure 4.18 – Flow chart for the comparison of the outputs of the reference IDM and the proposed IDM with the measurements obtained during a motion of the Quattro robot. FKM is the forward differential kinematic model. $IDKM$ is the inverse differential kinematic model which relates the end-effector pose velocity to the velocity of leg orientation vectors. $IDKM_2$ is the second order inverse differential kinematic model which relates the end-effector pose acceleration to the acceleration of leg orientation vectors.

The comparison is evaluated using the normalized root mean squares (NRMSE) metric:

$$NRMSE = \frac{RMSE}{\max(\tau) - \min(\tau)} \quad where \quad RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\tau_{ref} - \tau)_i^2}{n}} \tag{4.18}$$

where τ_{ref} is a reference model output torque; τ is a proposed model output torque; and n is the number of outputs; $\max(\tau)$ and $\min(\tau)$ are the maximum and minimum torque values of the proposed model outputs. In comparison of models, the maximum difference rate of motor torques is calculated less than 2%. This difference confirms that our proposed dynamic modeling is accurate enough to be used in control. This difference also does not mean that our proposed IDM is worse than the reference IDM, since the reference IDM is a simplified model.

Figure 4.19 depicts the superimposed output torques of the reference (red dashed line) and the proposed (black solid line) inverse dynamic models of the Quattro parallel robot during the motion of the test square trajectory shown in Figure 4.7. In Figure 4.19, one can observe that the output curves fit each other quite well. Figure 4.20 shows the errors between the reference IDM output torques ($\mathbf{\Gamma}_{ref}$) and proposed IDM output torques ($\mathbf{\Gamma}$) versus time for the given test square trajectory.

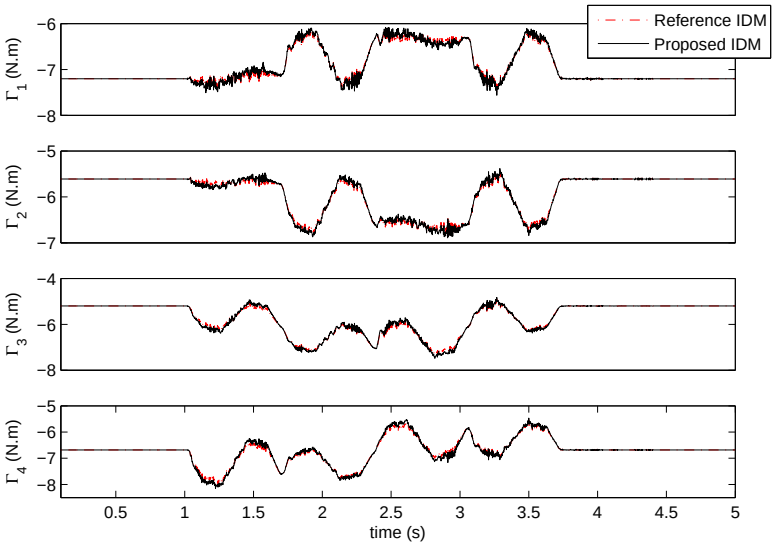


Figure 4.19 – Superimposed motor torques for the square test trajectory. Red dashed line is the output torque ($\mathbf{\Gamma}_{ref}$) of the reference inverse dynamic model, and black solid line is the output ($\mathbf{\Gamma}$) of our proposed inverse dynamic model. The maximum difference between these two models is found to be less than 2% using the normalized root mean squares of the output.

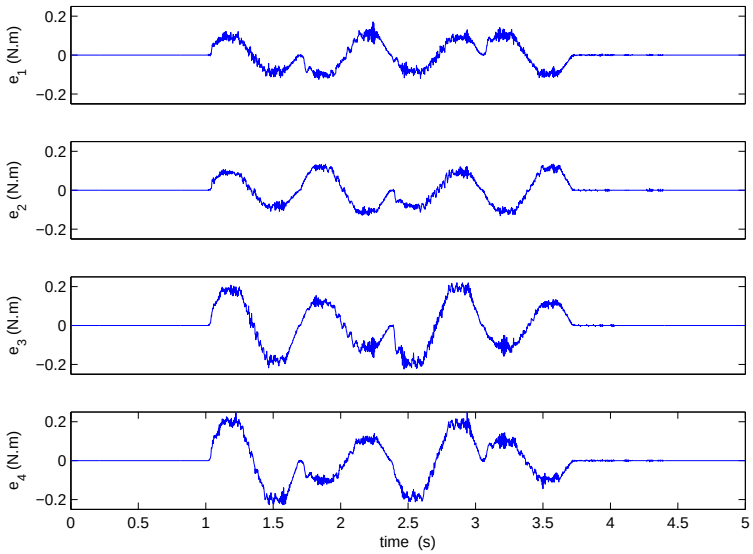


Figure 4.20 – The errors between the reference IDM output torques ($\mathbf{\Gamma}_{ref}$) and proposed IDM output torques ($\mathbf{\Gamma}$) for Figure 4.19.

4.2.2 Inverse Dynamic Model with High-Speed Dynamic State Observer

The proposed inverse dynamic model of the Quattro robot is integrated with the high-speed dynamic state estimation algorithm, and then compared with the reference inverse dynamic model [NKC+08] once more. This time, the unit orientation vectors ($\underline{\mathbf{x}}$) of the legs and their velocities ($\dot{\underline{\mathbf{x}}}$) for the proposed IDM are delivered by the high-speed dynamic state observer which uses the sequentially grabbed leg contours. The input accelerations can be either directly received from the desired reference trajectory or computed from the measured joint positions ($\mathbf{q}^m$). Figure 4.21 shows how comparison is made between the reference and proposed inverse dynamic models using the measurements during a motion of the Quattro parallel robot.

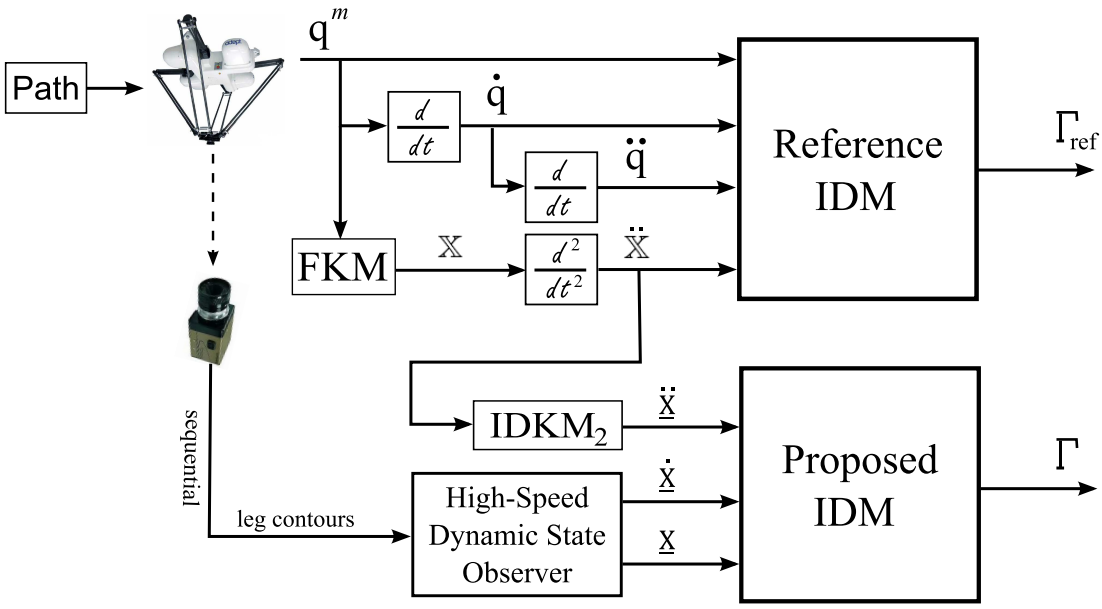


Figure 4.21 – Flow chart for the comparison of the outputs of the reference IDM and proposed IDM integrated with the high-speed state estimation algorithm. The accelerations are obtained from the measured articular positions of the Quattro robot. FKM is the forward differential kinematic model. $IDKM_2$ is the second order inverse differential kinematic model which relates the end-effector pose acceleration to the acceleration of leg orientation vectors.

In comparison of models, the difference rate of motor torques is calculated as 5% using the normalized root mean squares (NRMSE) metric. This difference implies that the proposed IDM with the integrated high-speed dynamic state observer is precise enough to be used in a dynamic control. Figure 4.22 depicts the superimposed output torques of the reference (red dashed line) and the proposed (black solid line) inverse dynamic models of the Quattro parallel robot during the motion of the test square trajectory shown in Figure 4.7. Figure 4.23 shows the errors between the output torques ($\mathbf{\Gamma}_{ref}$) of the reference IDM and the output torques ($\mathbf{\Gamma}$) of the proposed IDM versus time for the given test square trajectory.

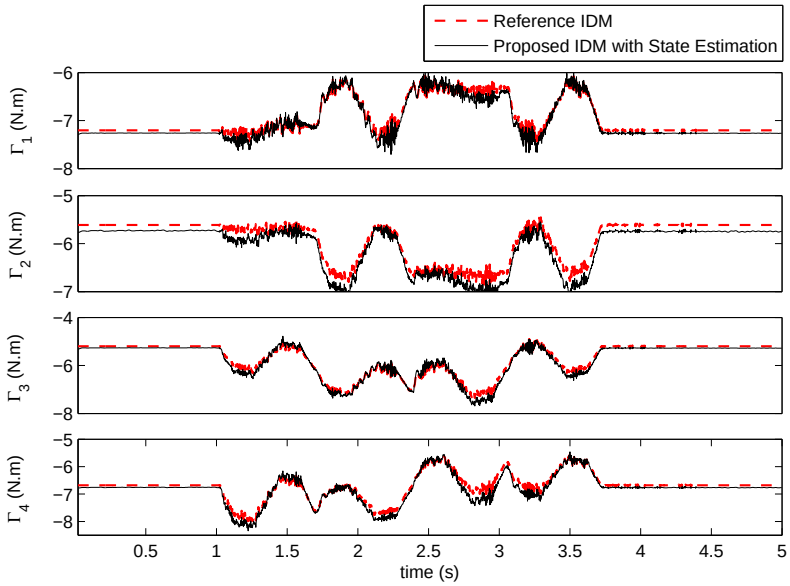


Figure 4.22 – Superimposed motor torques for the square test trajectory. Red dashed line is the output of the reference inverse dynamic model, and black solid line is the output of our proposed inverse dynamic model integrated with the high-speed dynamic state estimation algorithm. The difference between these two models is calculated as 5% using the normalized root mean squares of the output torques.

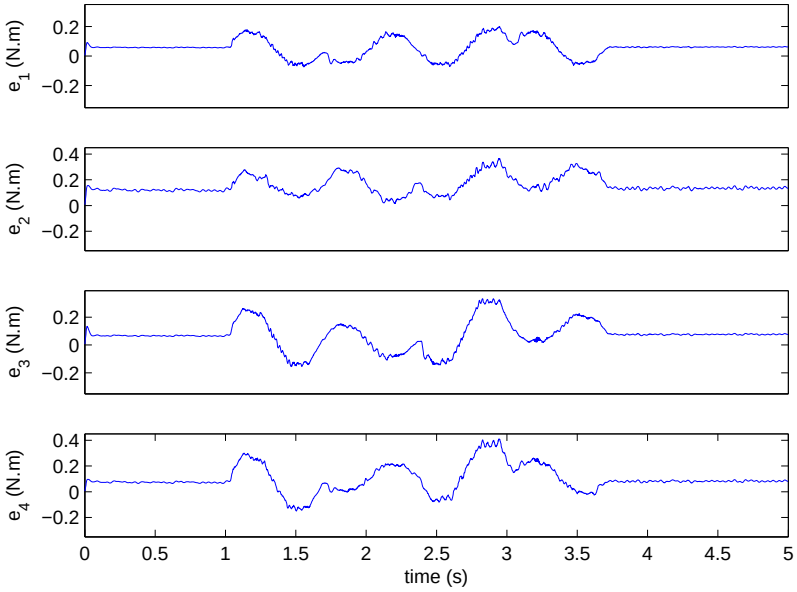


Figure 4.23 – The errors between the output torques ($\mathbf{\Gamma}_{ref}$) of the reference IDM and the output torques ($\mathbf{\Gamma}$) of the proposed IDM integrated with the high-speed dynamic state observer. The errors of Figure 4.22.

4.3 Conclusions

In this Chapter, we first performed the extrinsic calibration of the camera with respect to the base frame of the Quattro parallel robot by means of the leg edges. This extrinsic calibration was one of the initial steps of the high-speed dynamic state tracking algorithm. Secondly, we showed the feasibility of the high-speed dynamic state tracking algorithm in an off-line manner on the Quattro parallel robot. Then we discussed the source of the errors of the tracking results and proposed some ways to improve the speed and the accuracy of the algorithm.

Afterwards, we validated the correctness of the proposed linear inverse dynamic model which is based on the leg orientations of the Quattro parallel robot comparing to a reference inverse dynamic model. As we stated earlier, this proposed inverse dynamic model is linear on the condition that the leg orientations of the parallel robot and their velocities are provided. Therefore, we integrated the previously validated high-speed dynamic state observer to the inverse dynamic model so that this observer can feed the inverse dynamic model and can keep it linear. The integrated system of the current high-speed dynamic state observer and the proposed inverse dynamic model produced quite accurate results which allow to use this integrated system in the construction of a high-speed vision-based dynamic control framework.

In order to perform a vision-based dynamic control, we should first identify the dynamic parameters of the Quattro robot which is another quite challenging problem of parallel robots.

Chapter 5

Conclusions and Perspectives

5.1 Conclusions

This thesis accomplished 3 of the 4 remaining objectives of the *Integrated dynamic MICMAC*. This means that we have almost finished drawing the big picture of the vision-based control of parallel robots based on their leg observations. Speaking more precisely in the light of the MICMAC project, in this thesis we achieved the following goals:

- Firstly, we pushed control-oriented modeling of parallel robots further from the kinematic level to the dynamic level based on the observations of their legs. Furthermore, we improved modeling from joint-based representation to body-based representation. That is to say, we exploit the motion of concrete lines (i.e., orientations of the kinematic elements) rather than the motion of abstract axes (i.e., joint axes). This brings more physical and geometrical insight to the modeling of mechanisms. This proposed modeling scheme is applicable to wide range of parallel robots. In addition, the presented modeling scheme is linear since it uses the geometry of lines.
- Secondly, we presented a novel method which estimates the dynamic state (i.e., position and velocity) of a parallel robot using high-speed vision ($@\,500Hz$). In order to perform fast estimations, we sequentially observed small portions of the slim legs of the parallel robot by a technologically available sequential acquisition concept. This dynamic state estimator is a non-linear observer based on a virtual visual servoing scheme. Moreover, we proposed another dynamic state observer which is linear, although not yet feasible with off-the-shelf cameras.
- Thirdly, we proposed a versatile computed-torque control scheme based on the observations of the legs of parallel robots. This versatile control scheme allows one to perform a task in different control spaces so that one can examine the performances and then choose the best control space for the given task.

These new theoretic approaches are validated with the first promising simulation and experimental results which encourage us to explore further the proposed research path of parallel robots based on their leg observations in the future. The last unaccomplished goal of the integrated dynamic MICMAC project is as follows:

– Identification of the dynamic parameters of a parallel robot from its leg observations. Who is next to solve it?

The aforementioned novel schemes are contributions to the integrated dynamic MICMAC, and now we demonstrate once more the new updated state of the MICMAC art in Figure 5.1.

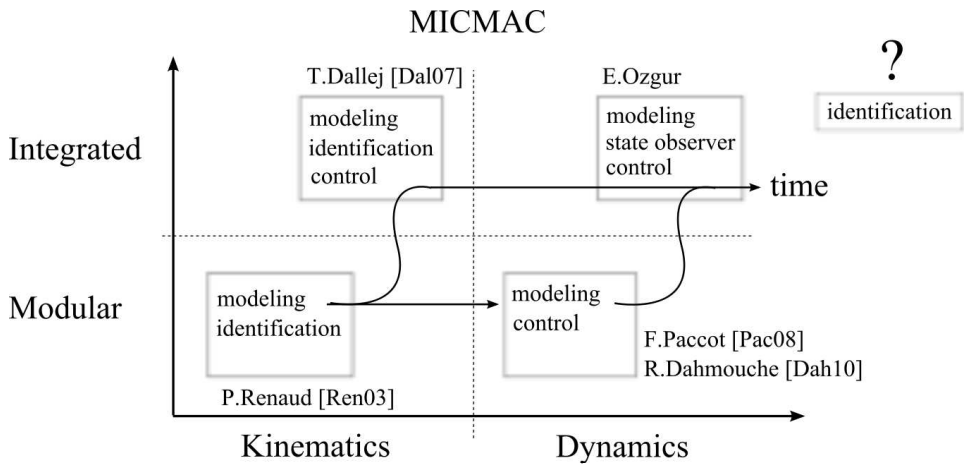


Figure 5.1 – New state of the MICMAC art.

Finally, we would like to point out that the message of this thesis is to say that: « *observing the legs of parallel robots is an interesting option, and it has the potential of making modeling and control of parallel robots simpler and more accurate by means of lines* ». Of course, these presented novel methods need to be investigated further to be competitive with the other methods.

5.2 Perspectives

There are a lot of things to improve in the drawn big picture of the vision-based control of parallel robots based on their leg observations. In this part, we remark the important points of this picture and we propose some future perspectives so as to make this picture more colorful.

5.2.1 Control-Oriented Linear Dynamic Modeling

The linearity, simplicity and accuracy properties of the proposed modeling scheme requires that the orientation vectors of the legs of a parallel robot can be measured precisely at each sampling instance. Therefore, the modeling scheme needs relevant sensing techniques so as to be feasible.

Moreover, this control-oriented linear dynamic modeling has the following main points to be improved:

– (i) the proposed modeling scheme is built upon the rigid body assumption and dynamic control of parallel robots interests with high speed motions. That is to say, in high speed motions it is possible that the vibrations can appear on the flexible kinematic elements

due to the accelerated masses of the parallel robot, and unfortunately the current proposed modeling scheme does not compensate for the flexibility. The source of the flexibility in a parallel robot can come from the construction materials of the kinematic elements and from the backlashes and clearances of the passive joints. Therefore, the proposed modeling scheme should be improved to take into account the dynamics of the robot flexibility so that the possible vibrations can be suppressed in the control.
- (ii) in order to increase the accuracy in the proposed modeling scheme and consequently in the control scheme, the dynamic parameters should be identified based on this modeling concept. This is an essential step before the control level. That is to say, the dynamic constraint equations of this proposed modeling scheme should be first rewritten in a linear form in terms of the dynamic parameters, and then should be solved with exciting trajectories.

5.2.2 Vision-Based High-Speed Dynamic State Observer

The proposed vision-based observer has the following main points to be improved:
- (i) the calibration errors of the extrinsic parameters of the camera are directly pronounced on the estimated states, because it is the main reference frame with respect to which everything is defined;
- (ii) if the observational distance of a cylindrical leg in the scene is quite long compared to its radius, then the depth estimation of the leg along the optical axis of the camera becomes very ill conditioned;
- (iii) good observability of the cylindrical legs by a camera at high speed is quite difficult because of the challenging lighting problem of the scene;
- (iv) existence of occlusion puts off the state estimation one sub-image acquisition time later, and it seems difficult to foresee an occlusion in the sub-image before the sub-image is grabbed;
- (v) it is not possible to perform a dynamic control, for the time being, more than $500\,Hz$ with the proposed vision-based system. This is because of the physical limits of the sensing environment rather than the theoretical limits, such as the relatively long exposure time of the CCD/CMOS sensor;

One solution to increase the accuracy and the robustness of our high-speed dynamic state observer can be the use of a camera which can grab multiple sub-images at a time, which is technologically possible today [URLP04]. The scenario is as follows:
- we can first simultaneously observe small portions of all the legs (e.g., for the Quattro robot the 4 lower-legs at the same time) rather than one by one, and consequently we can form the spatiotemporal reference input signal for the single iteration virtual visual servoing by stacking at least the last two sequential simultaneous measurements of the legs. This should increase the accuracy since there is more information for the present time and since fewer steps (i.e., 1 or 2 steps) are sufficient to evolve back in time with the approximated motion models than before. This will certainly increase the robustness against the occlusions, since it is possible to calculate the dynamic state as long as one sub-image per leg exists and at least one of the sub-images is grabbed at a different time instant than the others. In the case of detected occlusions on the certain legs, we

can foresee them for the next step and we can securely locate the sub-images on the unobstructed region of the legs. Moreover, the speed of this tracking algorithm will not be much worse than our current setup.

Although all these points are improved, this does not mean that this vision-based observer can ensure the best control performance. Maybe the optimum in practical sense is to fuse sensors in order to have a faster, more accurate and more robust observer. Thus, one can imagine simply fusing vision and motor encoders to define a new state observer, and then one can propose, for example, a joint-space ($\mathbf{q}$) plus a body-orientation space ($\underline{\mathbf{x}}$) computed-torque control scheme (JBS-CTC), which might yield better results.

5.2.3 We do need VISION and DYNAMIC CONTROL

Vision is a contactless optical sensor and dynamic control deals with forces. These properties make vision-based dynamic control the only solution for certain applications where the other sensing techniques (e.g., motor encoders) and control methods (e.g., kinematic control) are not adequate or not possible at all. For instance:

Micro space In micro space, the dimensions of robots and objects decrease to micro scales, and micro space tasks require adequate sensors which should have better resolution (i.e., nanoscale) than this micro space. Furthermore, these micro space tasks usually need global scene information. The construction of such sensors which can meet these demands is quite challenging and yet to be achieved. At this point, vision seems to be the only available proper sensing technique. Afterwards, it is interesting to note that the dominant (e.g., gravity) and the subordinate (e.g., surface) forces of macro space contrarily constitute the subordinate and dominant forces of micro space. Furthermore, in micro space, the surface forces (e.g., Van der Waals) are a lot stronger than the gravity forces, such that the manipulated micro-scale object can fly from one surface to another easily and stick to it because of adhesion forces. In such a space where the dynamic effects are extreme, dynamic control is inevitable. For example, one of the most well known problems in micro space is that it is difficult to release an object which is being held by the gripper to a desired position due to adhesive surface forces. The simple solution might be to arrive this desired position with a certain acceleration and suddenly stop the gripper such that the final inertia force of the object can defeat the surface adhesive forces and release itself.

Macro space In macro space, one of interesting application is the control of cable driven giant parallel robots. The main objective of such giant parallel robots is both to carry heavy objects and to do long-distance moves. Therefore, they have long cables, and unfortunately the forces of gravity along such long cables can deform the straightness of cables and can lengthen them due to their elasticity. Consequently, the rigidity assumption is not valid anymore. In addition, these effects will probably be exaggerated by the presence of a heavy payload. Therefore, the use of motor encoders as a sensor in such a giant robot can give only the local information, and furthermore it is not very feasible to place any sensor somewhere on the cable in order to have more information about the geometry since the cables are rolled back and forward on spools. Thus, the current situation seems poor to

provide a precise control of the moving platform because of all these deformations caused by the dynamic effects and insufficient feedback about the robot. Once again, vision seems one of the best solutions to observe the geometry of such giant robots since it is a contactless distance sensor and of course the dynamic control is essential too in order to compensate undesired dynamic effects.

5.2.4 From MICMAC to RODAP

Why must parallel robots be blind? Why not save the skillful ugly parallel robots by dressing them up? So, why should the outcome of MICMAC art not be the initiation of a new art: MICMAC oriented RObot Design and AdaPtation (MICMAC-RODAP). The objectives of this new art can be envisioned as follows:

- *Robots with MICMAC eyes*: to change vision from being an exteroceptive sensor to a proprioceptive sensor on parallel robots, and to make this proprioceptive vision sensor exploit the MICMAC algorithms on an embedded system of the robot;
- *Robots with MICMAC legs*: to design parallel robots with slim, cylindrical and observable legs so that they are consistent with the line geometry and vision;
- *Robots with MICMAC costumes*: to dress up existing geometrically and visually non compatible parallel robots with costumes in order to adapt them to the MICMAC art. For instance, imagine axial cylindrical shells that just envelope the physical boundaries of non-uniform links of a parallel robot.

Thus, one can imagine a RODAP development framework in order either to produce a MICMAC robot or to adapt a current parallel robot to the MICMAC concept. This framework can be built upon the following 6 modules which interact between each other:

1. **CAD module**: Synthesis of the geometry of the kinematic elements of the robot.
2. **Blender module**: Treatment of the lighting conditions of the robot legs.
3. **Adams & Simulink module**: Analysis of the dynamics and control of the robot.
4. **Coding module**: Development of a MICMAC library in advanced programming languages where the MICMAC theories are efficiently implemented (i.e., real time and security considerations).
5. **Sensing module**: Development of fast smart cameras and adaptation of other intelligent sensing technologies for the MICMAC concept.
6. **Optimization module**: Governs the interactions between the modules in order to satisfy the desired criteria for a MICMAC robot.

Appendix A

Modeling: Applied to Parallel Robots

A.1 The Gough-Stewart Robot

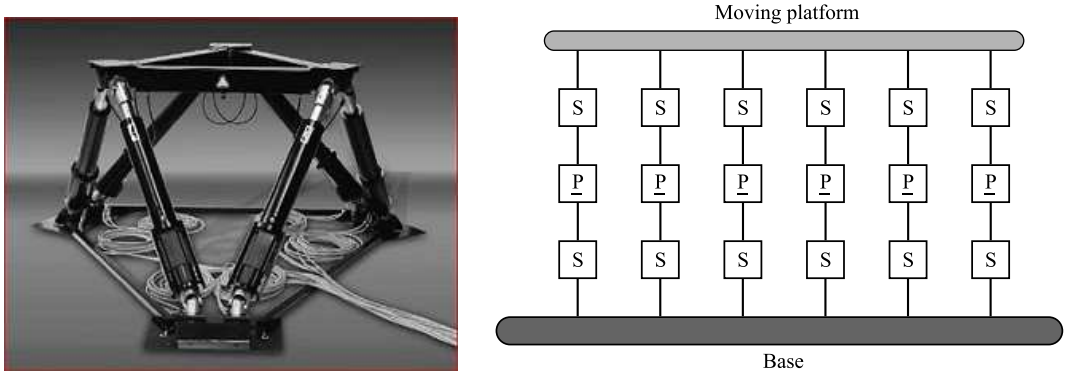


Figure A.1 – The Gough-Stewart parallel robot (*left*) and its graphical layout (*right*).

A.1.1 Geometry and Notation

The Gough-Stewart parallel robot incorporates 6 identical kinematic legs. Figure A.1 shows the Gough-Stewart parallel robot and its graphical layout. The mechanism's fixed base, $\{\mathbf{A}_1, \ldots, \mathbf{A}_6\}$, holds the moving-platform with these kinematic legs. The moving-platform is a single rigid body, $\{\mathbf{B}_1, \ldots, \mathbf{B}_6\}$. Each kinematic leg is a telescopic system consisting of a single kinematic element $[\mathbf{A}_i\mathbf{B}_i]$ with an embedded prismatic actuator. A kinematic leg of the Gough-Stewart is symbolically noted as $S-\underline{P}-S$ where S and $\underline{P}$ stand for a passive spherical joint and an actuated prismatic joint, respectively. These actuators give 6 degrees of freedom to the moving-platform by pushings and pullings: 3 translational movements in x, y, z axes (lateral, longitudinal and vertical), and 3 rotational movements (pitch, roll and yaw). Figure A.2 depicts the geometrical notation of the Gough-Stewart robot. In modeling, the following notation is used:

– $i = 1, 2, 3, 4, 5, 6$ denotes the kinematic legs.

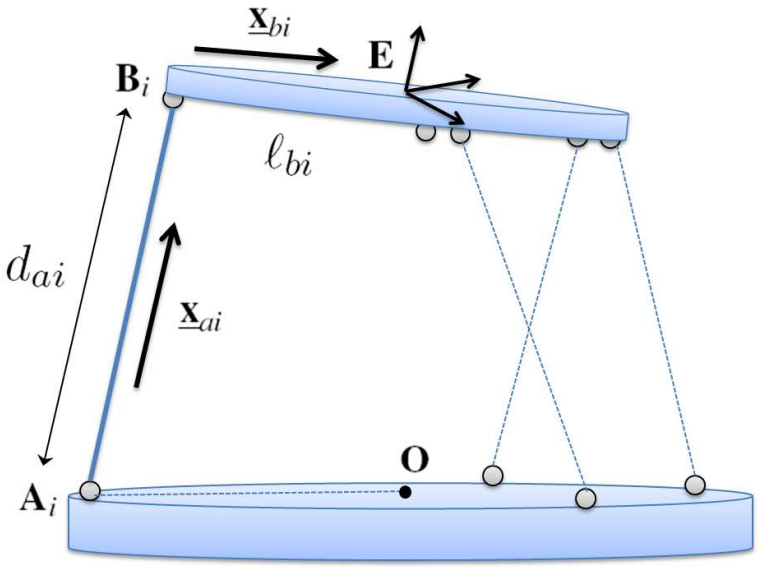


Figure A.2 – The notation of the Gough-Stewart parallel robot.

- $j = \{a, b\}$ is the literal representation of the kinematic elements.
- $\xi_{geo} = \{\mathbf{O}, \mathbf{A}_i, \ell_{bi}, \alpha_i, \beta_i\}$ are the geometric parameters (constant lengths and points).
- $\xi_{dyn} = \{m_{ji}, \mathcal{I}_{ji}, f_{v_i}, f_{c_i}\}$ are the dynamic parameters (weights, inertias and frictions).
- $\mathcal{F}_o = (\mathbf{O}, \underline{\mathbf{x}}_o, \underline{\mathbf{y}}_o, \underline{\mathbf{z}}_o)$, $\mathcal{F}_e = (\mathbf{E}, \underline{\mathbf{x}}_e, \underline{\mathbf{y}}_e, \underline{\mathbf{z}}_e)$, $\mathcal{F}_{ai} = (\mathbf{A}_i, \underline{\mathbf{x}}_{ai}, \underline{\mathbf{y}}_{ai}, \underline{\mathbf{z}}_{ai})$ and $\mathcal{F}_{bi} = (\mathbf{B}_i, \underline{\mathbf{x}}_{bi}, \underline{\mathbf{y}}_{bi}, \underline{\mathbf{z}}_{bi})$ denote respectively the base, the end-effector, the i^{th} kinematic leg and the moving-platform's i^{th} virtual kinematic element frames.
- d_{ai} is the dynamic length of the i^{th} kinematic leg.
- The end-effector pose is composed of the origin $\mathbf{E}$, of the x-axis unit vector $\underline{\mathbf{x}}_e$ and of the rotational angle θ_e around the x-axis $\underline{\mathbf{x}}_e$ of the end-effector frame. The end-effector pose velocity is then $\dot{\mathbf{E}}$, $\dot{\underline{\mathbf{x}}}_e$ and $\dot{\theta}_e$:

$$\mathbb{X} \triangleq \begin{bmatrix} \mathbf{E} \\ \underline{\mathbf{x}}_e \\ \theta_e \end{bmatrix}, \qquad \dot{\mathbb{X}} \triangleq \begin{bmatrix} \dot{\mathbf{E}} \\ \dot{\underline{\mathbf{x}}}_e \\ \dot{\theta}_e \end{bmatrix} \in \Re^{7\times 1} \tag{A.1}$$

A.1.2 State Variables

Kinematic Element Types

The Gough-Stewart parallel robot has 2 different kinds of kinematic elements:

Kinematic legs $[\mathbf{A}_i\mathbf{B}_i]$ Each of these kinematic elements has 3 dof rotational extrinsic mobility due to (S)pherical joint and as well as 1 dof translational intrinsic mobility (along its direction) due to the (P)rismatic joint. Spherical joints located at the tips of the kinematic element allow for a self-rotation which does not have any effect on the moving-platform

configuration or on the end-effector frame. It is thus a *telescopic type* kinematic element: $\{\underline{\mathbf{x}}_{ai},\, d_{ai}\}$.

Moving-platform $[\mathbf{B}_i\mathbf{E}]$ Each of these virtual kinematic elements of the moving-platform has 3 dof rotational extrinsic mobility due to (S)pherical joint and as well as 1 dof self-rotational intrinsic mobility (around its direction) due to the pushings and pullings of the other kinematic legs attached to the moving-platform. It is thus an *spindle type* kinematic element: $\{\underline{\mathbf{x}}_{bi},\, \theta_{bi}\}$.

The $\{\underline{\mathbf{x}}_{ai},\, d_{ai},\, \underline{\mathbf{x}}_{ai},\, \theta_{bi}\}$ are the new variables that redefine the state of the mechanism.

A.1.3 Kinematics

Mass Centers

Assuming that all the kinematic elements are homogenous and symmetric, the mass center positions are written as follows:

$$\mathbf{S}_{ai} \;=\; \mathbf{A}_i \;+\; \frac{1}{2}\, d_{ai}\, \underline{\mathbf{x}}_{ai}\,, \quad \mathbf{S}_{bi} \;=\; \mathbf{A}_i \;+\; d_{ai}\, \underline{\mathbf{x}}_{ai} \;+\; \ell_{bi}\, \underline{\mathbf{x}}_{bi} \tag{A.2}$$

Velocities

The translational and rotational velocities of the kinematic elements are computed as below: (i) translational velocities,

$$\dot{\mathbf{S}}_{ai} \;=\; \frac{1}{2}\, \left(\, \dot{d}_{ai}\, \underline{\mathbf{x}}_{ai} \;+\; d_{ai}\, \dot{\underline{\mathbf{x}}}_{ai} \right)\,, \quad \dot{\mathbf{S}}_{bi} \;=\; \left(\, \dot{d}_{ai}\, \underline{\mathbf{x}}_{ai} \;+\; d_{ai}\, \dot{\underline{\mathbf{x}}}_{ai} \right) \;+\; \ell_{bi}\, \dot{\underline{\mathbf{x}}}_{bi} \tag{A.3}$$

(ii) rotational velocities,

$$\boldsymbol{\omega}_{ai} = \underline{\mathbf{x}}_{ai} \;\times\; \dot{\underline{\mathbf{x}}}_{ai} \;+\; \dot{\theta}_{ai}\, \underline{\mathbf{x}}_{ai} \quad \Rightarrow \quad \boldsymbol{\omega}_{ai} \triangleq \underline{\mathbf{x}}_{ai} \;\times\; \dot{\underline{\mathbf{x}}}_{ai} \tag{A.4}$$

The rotation θ_{ai} does not change the posture of the moving-platform.

$$\boldsymbol{\omega}_{bi} = \underline{\mathbf{x}}_{bi} \;\times\; \dot{\underline{\mathbf{x}}}_{bi} \;+\; \dot{\theta}_{bi}\, \underline{\mathbf{x}}_{bi} \tag{A.5}$$

Accelerations

The translational and rotational accelerations of the kinematic elements are computed as below: (i) translational accelerations,

$$\ddot{\mathbf{S}}_{ai} \;=\; \frac{1}{2}\, \left(\, \ddot{d}_{ai}\, \underline{\mathbf{x}}_{ai} \;+\; 2\, \dot{d}_{ai}\, \dot{\underline{\mathbf{x}}}_{ai} \;+\; d_{ai}\, \ddot{\underline{\mathbf{x}}}_{ai} \right) \tag{A.6}$$

$$\ddot{\mathbf{S}}_{bi} \;=\; \left(\, \ddot{d}_{ai}\, \underline{\mathbf{x}}_{ai} \;+\; 2\, \dot{d}_{ai}\, \dot{\underline{\mathbf{x}}}_{ai} \;+\; d_{ai}\, \ddot{\underline{\mathbf{x}}}_{ai} \right) \;+\; \ell_{bi}\, \ddot{\underline{\mathbf{x}}}_{bi} \tag{A.7}$$

(ii) rotational accelerations,

$$\dot{\boldsymbol{\omega}}_{ai} \triangleq \underline{\mathbf{x}}_{ai} \;\times\; \ddot{\underline{\mathbf{x}}}_{ai} \tag{A.8}$$

$$\dot{\boldsymbol{\omega}}_{bi} = \underline{\mathbf{x}}_{bi} \;\times\; \ddot{\underline{\mathbf{x}}}_{bi} \;+\; \ddot{\theta}_{bi}\, \underline{\mathbf{x}}_{bi} \;+\; \dot{\theta}_{bi}\, \dot{\underline{\mathbf{x}}}_{bi} \tag{A.9}$$

A.1.4 Kinematic Constraints

Assuming that the end-effector frame is located at the mass center position of the moving platform, the closed-loop constraint equation for each of the kinematic legs can be written as follow:

$$\overrightarrow{\mathbf{OE}} - \ell_{bi}\,\underline{\mathbf{x}}_{bi} - d_{ai}\,\underline{\mathbf{x}}_{ai} - \overrightarrow{\mathbf{OA}_i} = \mathbf{0} \tag{A.10}$$

where $\mathbf{O}$ and $\mathbf{A}_i$ are constants. Afterwards, one can differentiate the last closed-loop constraint equation with respect to time in order to obtain the motion constraint equation, which yields:

$$\dot{\mathbf{E}} - \ell_{bi}\,\dot{\underline{\mathbf{x}}}_{bi} - \dot{d}_{ai}\,\underline{\mathbf{x}}_{ai} - d_{ai}\,\dot{\underline{\mathbf{x}}}_{ai} = \mathbf{0} \tag{A.11}$$

The motion constraints for the attachment points $\mathbf{B}_i$ of the moving-platform can be derived from (A.11) using the constant parameters α_i and β_i as below:

$$\underline{\mathbf{x}}_{bi} = \alpha_i\,\underline{\mathbf{x}}_e + \beta_i\,\underline{\mathbf{y}}_e \tag{A.12}$$

$$\dot{\underline{\mathbf{x}}}_{bi} = \alpha_i\,\dot{\underline{\mathbf{x}}}_e + \beta_i\,\dot{\underline{\mathbf{y}}}_e \tag{A.13}$$

$$\dot{\underline{\mathbf{y}}}_e = \boldsymbol{\omega}_e \times \underline{\mathbf{y}}_e = (\,\underline{\mathbf{x}}_e \times \dot{\underline{\mathbf{x}}}_e + \dot{\theta}_e\,\underline{\mathbf{x}}_e\,) \times \underline{\mathbf{y}}_e \tag{A.14}$$

$$\dot{\underline{\mathbf{y}}}_e = \underline{\mathbf{z}}_e \times \dot{\underline{\mathbf{x}}}_e + \dot{\theta}_e\,\underline{\mathbf{z}}_e \tag{A.15}$$

$$\dot{\underline{\mathbf{x}}}_{bi} = \alpha_i\,\dot{\underline{\mathbf{x}}}_e + \beta_i\,\underline{\mathbf{z}}_e \times \dot{\underline{\mathbf{x}}}_e + \beta_i\,\underline{\mathbf{z}}_e\,\dot{\theta}_e \tag{A.16}$$

$$\dot{\mathbf{E}} - \ell_{bi}\,\dot{\underline{\mathbf{x}}}_{bi} = \dot{\mathbf{B}}_i \tag{A.17}$$

$$\dot{\mathbf{B}}_i = L_{Bi}\,\dot{\mathbb{X}}, \qquad L_{Bi} = \begin{bmatrix} I_3 & -\ell_{bi}\,(\,\alpha_i\,I_3 + \beta_i\,[\underline{\mathbf{z}}_e]_\times\,) & -\ell_{bi}\,\beta_i\,\underline{\mathbf{z}}_e \end{bmatrix} \in \Re^{3\times 7} \tag{A.18}$$

where L_{Bi} is the relation between the Cartesian velocity of the terminal point of the i^{th} kinematic leg and the end-effector pose velocity $\dot{\mathbb{X}}$.

Constraints on the Active Telescopic Kinematic Legs

Constraints on the variations of the lengths of the telescopic kinematic elements are computed as follows:

$$\dot{d}_{ai}\,\underline{\mathbf{x}}_{ai} + d_{ai}\,\dot{\underline{\mathbf{x}}}_{ai} = L_{Bi}\,\dot{\mathbb{X}} \tag{A.19}$$

$$\dot{d}_{ai} = M_{d_{ai}}\,\dot{\mathbb{X}}, \qquad M_{d_{ai}} = \begin{bmatrix} \underline{\mathbf{x}}_{ai}^T\,L_{Bi} \end{bmatrix} \in \Re^{1\times 7} \tag{A.20}$$

Constraints on the orientations of the telescopic kinematic elements are computed as below:

$$d_{ai}\,\dot{\underline{\mathbf{x}}}_{ai} + \underline{\mathbf{x}}_{ai}\,M_{d_{ai}}\,\dot{\mathbb{X}} = L_{Bi}\,\dot{\mathbb{X}} \tag{A.21}$$

$$\dot{\underline{\mathbf{x}}}_{ai} = M_{ai}\,\dot{\mathbb{X}}, \qquad M_{ai} = \left[\frac{1}{d_{ai}}\,(\,L_{Bi} - \underline{\mathbf{x}}_{ai}\,M_{d_{ai}}\,)\right] \in \Re^{3\times 7} \tag{A.22}$$

Constraints on the Passive Moving-Platform Virtual Kinematic Elements

Constraints on the orientations of the virtual axis type kinematic elements of the moving-platform are computed as follows:

$$\dot{\underline{\mathbf{x}}}_{bi} \,=\, M_{bi}\,\dot{\mathbb{X}}\,, \qquad M_{bi} \,=\, \left[\; \mathbf{0}_{3\times 3} \quad (\,\alpha_i\, I_3 \,+\, \beta_i\,[\underline{\mathbf{z}}_e]_\times\,) \quad \beta_i\,\underline{\mathbf{z}}_e \;\right] \in \Re^{3\times 7} \tag{A.23}$$

Constraints on the self-rotations of the virtual axis type kinematic elements of the moving-platform are computed as below:

$$\boldsymbol{\omega}_{bi} \,=\, \boldsymbol{\omega}_e\,, \qquad i = 1,\,\ldots,6 \tag{A.24}$$

$$\boldsymbol{\omega}_e \,=\, M_{\omega_e}\,\dot{\mathbb{X}}\,, \qquad M_{\omega_e} \,=\, \left[\; \mathbf{0}_{3\times 3} \quad [\,\underline{\mathbf{x}}_e\,]_\times \quad \underline{\mathbf{x}}_e \;\right] \in \Re^{3\times 7} \tag{A.25}$$

$$\dot{\theta}_{bi} \,=\, \underline{\mathbf{x}}_{bi}^T\,\boldsymbol{\omega}_{bi} \,=\, \underline{\mathbf{x}}_{bi}^T\,(\underline{\mathbf{x}}_{bi} \,\times\, \dot{\underline{\mathbf{x}}}_{bi} \,+\, \dot{\theta}_{bi}\,\underline{\mathbf{x}}_{bi}) \tag{A.26}$$

$$\dot{\theta}_{bi} \,=\, M_{\theta_{bi}}\,\dot{\mathbb{X}}\,, \qquad M_{\theta_{bi}} \,=\, \left[\,\underline{\mathbf{x}}_{bi}^T\,M_{\omega_e}\,\right] \in \Re^{1\times 7} \tag{A.27}$$

A.1.5 Kinematic Coordinates

Equation (A.28) and Table A.1 show the motion basis and the kinematic coordinates of the mechanism, respectively.

$$u_{d_{ai}} \,=\, \dot{d}_{ai}\,, \qquad \mathbf{u}_{x_{ai}} \,=\, \dot{\underline{\mathbf{x}}}_{ai}\,, \qquad \mathbf{u}_{x_{bi}} \,=\, \dot{\underline{\mathbf{x}}}_{bi}\,, \qquad u_{\theta_{bi}} \,=\, \dot{\theta}_{bi}\,, \qquad i \,=\, 1,\ldots,6 \tag{A.28}$$

Table A.1 – The (transposed) kinematic coordinates of the Gough-Stewart robot, i=1,...,6.

	$\partial\,\dot{\mathbf{S}}_{\,ai}$	$\partial\,\boldsymbol{\omega}_{\,ai}$	$\partial\,\dot{\mathbf{S}}_{\,bi}$	$\partial\,\boldsymbol{\omega}_{\,bi}$
$\partial\,\dot{d}_{\,ai}$	$\frac{1}{2}\,\underline{\mathbf{x}}_{ai}^T$	$\mathbf{0}_{3\times 1}$	$\underline{\mathbf{x}}_{ai}^T$	$\mathbf{0}_{3\times 1}$
$\partial\,\dot{\underline{\mathbf{x}}}_{ai}$	$\frac{1}{2}\,d_{ai}\,\mathbf{I}_3$	$[\,\underline{\mathbf{x}}_{ai}\,]_\times^T$	$d_{ai}\,\mathbf{I}_3$	$\mathbf{0}_{3\times 3}$
$\partial\,\dot{\underline{\mathbf{x}}}_{bi}$	$\mathbf{0}_{3\times 3}$	$\mathbf{0}_{3\times 3}$	$\ell_{bi}\,\mathbf{I}_3$	$[\,\underline{\mathbf{x}}_{bi}\,]_\times^T$
$\partial\,\dot{\theta}_{\,bi}$	$\mathbf{0}_{3\times 1}$	$\mathbf{0}_{3\times 1}$	$\mathbf{0}_{3\times 1}$	$\underline{\mathbf{x}}_{bi}^T$

A.1.6 Dynamic Coordinates

Listing the Active and Reactive Forces

The active forces are as follows:

– *Actuator and Gravity Forces:*

$$\mathbf{f}_{d_{ai}} \,=\, f_{d_{ai}}\,\underline{\mathbf{x}}_{ai}\,, \qquad \mathbf{f}_{\mathbf{g}(ai)} \,=\, m_{ai}\,\mathbf{g}\,, \qquad \mathbf{f}_{\mathbf{g}(bi)} \,=\, m_{bi}\,\mathbf{g} \tag{A.29}$$

where $\mathbf{f}_{d_{ji}}$ and $\mathbf{f}_{\mathbf{g}(ji)}$ are the actuator and gravity forces, respectively.

The reactive forces are as follows:

– *Actuator Inertial Forces:* The inertial forces of the linear actuators are as below:

$$\mathbf{f}^{*}_{d_{ai}} \;=\; -\,m_{d_{ai}}\,\ddot{d}_{ai}\,\underline{\mathbf{x}}_{ai} \tag{A.30}$$

where $m_{d_{ai}}$ is the mass moved inside the i^{th} kinematic leg by the linear actuator.

– *Kinematic Element Body Inertial Forces and Torques:* The inertial forces and torques of the kinematic elements are follows:

$$\mathbf{f}^{*}_{ai} \;=\; -\,m_{ai}\,\ddot{\mathbf{S}}_{ai}\,, \qquad \boldsymbol{\tau}^{*}_{ai} \;=\; -\,\mathcal{I}^{T}_{ai}\,\dot{\boldsymbol{\omega}}_{ai} \;-\; \boldsymbol{\omega}_{ai} \;\times\; (\mathcal{I}^{T}_{ai}\,\boldsymbol{\omega}_{ai}) \tag{A.31}$$

$$\mathbf{f}^{*}_{bi} \;=\; -\,m_{bi}\,\ddot{\mathbf{S}}_{bi}\,, \qquad \boldsymbol{\tau}^{*}_{bi} \;=\; -\,\mathcal{I}^{T}_{bi}\,\dot{\boldsymbol{\omega}}_{bi} \;-\; \boldsymbol{\omega}_{bi} \;\times\; (\mathcal{I}^{T}_{bi}\,\boldsymbol{\omega}_{bi}) \tag{A.32}$$

– *Active Joint Frictional Forces:* The frictional forces of the intrinsic active joints are as follows:

$$\bar{\mathbf{f}}_{d_{ai}} \;=\; -\,(\;\bar{f}_{v(d_{ai})}\,\dot{d}_{ai} \;+\; \bar{f}_{c(d_{ai})}\,sign(\;\dot{d}_{ai}\;)\;)\,\underline{\mathbf{x}}_{ai} \tag{A.33}$$

where $\bar{f}_{v(d_{ai})}$ and $\bar{f}_{c(d_{ai})}$ are the viscous and Coulomb friction coefficients of the linear actuators.

– *Passive Joint Frictional Torques:* The frictional torques of the extrinsic passive joints are as follows:

$$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{ai}} \;=\; -\,\bar{\tau}_{v(\underline{\mathbf{x}}_{ai})}\,\boldsymbol{\omega}_{ai} \;-\; \bar{\tau}_{c(\underline{\mathbf{x}}_{ai})}\,sign(\;\boldsymbol{\omega}^{T}_{ai}\,\underline{\mathbf{z}}_{ai}\;)\,\underline{\mathbf{z}}_{ai} \tag{A.34}$$

$$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{bi}} \;=\; -\,\bar{\tau}_{v(\underline{\mathbf{x}}_{bi})}\,(\;\boldsymbol{\omega}_{bi} \;-\; \boldsymbol{\omega}_{ai}\;) \;-\; \bar{\tau}_{c(\underline{\mathbf{x}}_{bi})}\,sign(\;(\;\boldsymbol{\omega}_{bi} \;-\; \boldsymbol{\omega}_{ai}\;)^{T}\,\underline{\mathbf{z}}_{bi}\;)\,\underline{\mathbf{z}}_{bi} \tag{A.35}$$

where $\bar{\tau}_{v(\underline{\mathbf{x}}_{ji})}$ and $\bar{\tau}_{c(\underline{\mathbf{x}}_{ji})}$ are the viscous and Coulomb friction coefficients of the passive rotary joints.

Table (A.2) tabulates all these local forces and torques of the Gough parallel robot.

Table A.2 – The local forces and torques of the Gough parallel robot, i=1, . . . ,6.

	Active		*Friction*		*Inertia**	
	Actuator	*Gravity*	*Actuator*	*Passive Joint*	*Actuator*	*Element*
$Forces\,(ai)$	$f_{d_{ai}}\,\underline{\mathbf{x}}_{ai}$	$\mathbf{f}_{\mathbf{g}(ai)}$	$\bar{\mathbf{f}}_{d_{ai}}$	$\mathbf{0}$	$\mathbf{f}^{*}_{d_{ai}}$	$\mathbf{f}^{*}_{ai}$
$Torques\,(ai)$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{ai}}$	$\mathbf{0}$	$\boldsymbol{\tau}^{*}_{ai}$
$Forces\,(bi)$	$\mathbf{0}$	$\mathbf{f}_{\mathbf{g}(bi)}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{f}^{*}_{bi}$
$Torques\,(bi)$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{bi}}$	$\mathbf{0}$	$\boldsymbol{\tau}^{*}_{bi}$

Computing Dynamic Coordinates

So as to eliminate the non-contributing forces, the dynamic coordinates are computed through the matrix-wise multiplication of the Tables A.1 (kinematic coordinates) and A.2 (sum of the local forces and torques).

$$\begin{bmatrix} \mathbb{F}_{d_{ai}} \\ \mathbb{F}_{\underline{x}_{ai}} \\ \mathbb{F}_{\underline{x}_{bi}} \\ \mathbb{F}_{\theta_{bi}} \end{bmatrix} \;=\; \begin{bmatrix} Kinematic \\ Coordinates \\ Table\ A.1 \end{bmatrix}_{(4\times 4)} \begin{bmatrix} Sum\ of \\ Forces \\ Torques \\ Table\ A.2 \end{bmatrix}_{(4\times 1)} \tag{A.36}$$

which can be explicitly written as follows:

$$\begin{bmatrix} \mathbb{F}_{d_{ai}} \\ \mathbb{F}_{\underline{x}_{ai}} \\ \mathbb{F}_{\underline{x}_{bi}} \\ \mathbb{F}_{\theta_{bi}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\,\underline{\mathbf{x}}_{ai}^T & \mathbf{0}_{3\times1} & \underline{\mathbf{x}}_{ai}^T & \mathbf{0}_{3\times1} \\ \frac{1}{2}\,d_{ai}\,\mathbf{I}_3 & [\,\underline{\mathbf{x}}_{ai}\,]_\times^T & d_{ai}\,\mathbf{I}_3 & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \ell_{bi}\,\mathbf{I}_3 & [\,\underline{\mathbf{x}}_{bi}\,]_\times^T \\ \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \underline{\mathbf{x}}_{bi}^T \end{bmatrix} \begin{bmatrix} f_{d_{ai}}\,\underline{\mathbf{x}}_{ai} + \widetilde{\mathbf{f}}_{ai} \\ \bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{ai}} + \boldsymbol{\tau}_{ai}^* \\ \mathbf{f}_{\mathbf{g}(bi)} + \mathbf{f}_{bi}^* \\ \bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{bi}} + \boldsymbol{\tau}_{bi}^* \end{bmatrix} \tag{A.37}$$

where

$$\widetilde{\mathbf{f}}_{ai} = \mathbf{f}_{\mathbf{g}(ai)} + \bar{\mathbf{f}}_{d_{ai}} + \mathbf{f}_{d_{ai}}^* + \mathbf{f}_{ai}^* \tag{A.38}$$

A.1.7 Dynamic Constraints

Exploiting (2.110), the dynamic constraints of the Gough-Stewart robot are written as follows:

$$\mathbf{0}_{6\times1} = M_{d_a}^T\,\mathbb{F}_{d_a} + M_a^T\,\mathbb{F}_a + M_b^T\,\mathbb{F}_b + M_{\theta_b}^T\,\mathbb{F}_{\theta_b} \tag{A.39}$$

where $\mathbb{F}_{d_a} \in \Re^{6\times1}$, $\mathbb{F}_a \in \Re^{18\times1}$, $\mathbb{F}_b \in \Re^{18\times1}$ and $\mathbb{F}_{\theta_b} \in \Re^{6\times1}$ are the stacked vectors of the dynamic coordinates:

$$\mathbb{F}_{d_a} = \begin{bmatrix} \mathbb{F}_{d_{a1}} \\ \vdots \\ \mathbb{F}_{d_{a6}} \end{bmatrix}, \quad \mathbb{F}_a = \begin{bmatrix} \mathbb{F}_{\underline{x}_{a1}} \\ \vdots \\ \mathbb{F}_{\underline{x}_{a4}} \end{bmatrix}, \quad \mathbb{F}_b = \begin{bmatrix} \mathbb{F}_{\underline{x}_{b1}} \\ \vdots \\ \mathbb{F}_{\underline{x}_{b4}} \end{bmatrix}, \quad \mathbb{F}_\theta = \begin{bmatrix} \mathbb{F}_{\theta_{b1}} \\ \vdots \\ \mathbb{F}_{\theta_{b6}} \end{bmatrix} \tag{A.40}$$

and where $M_{d_a} \in \Re^{6\times7}$, $M_a \in \Re^{18\times7}$, $M_b \in \Re^{18\times7}$ and $M_{\theta_b} \in \Re^{6\times7}$ are the stacked matrices of the inverse differential kinematic models $M_{d_{ai}} \in \Re^{1\times7}$, $M_{ai} \in \Re^{3\times7}$, $M_{bi} \in \Re^{3\times7}$ and $M_{\theta_{bi}} \in \Re^{1\times7}$, respectively.

A.2 The Delta Robot

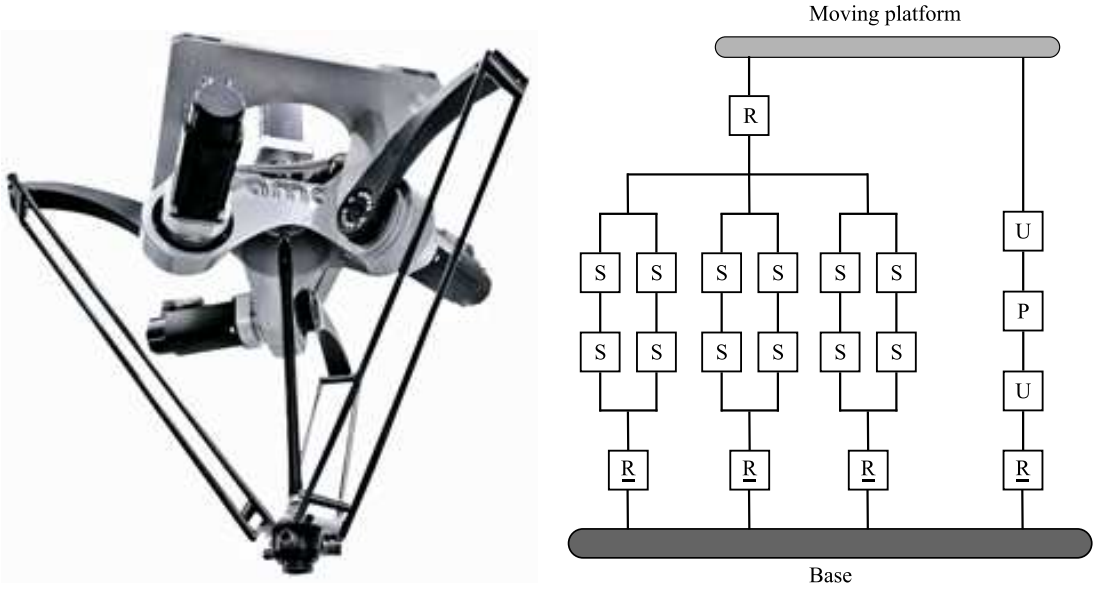


Figure A.3 – The Delta parallel robot (*left*) and its graphical layout (*right*).

A.2.1 Geometry and Notation

The Delta robot consists of 4 kinematic legs interconnecting the fixed base with the moving-platform. The moving-platform is a single rigid triangular body $\{\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3\}$. Figure A.3 shows the Delta robot and its graphical layout. All the kinematic legs are actuated from the base by revolute motors located at $\{\mathbf{P}_1, \ldots, \mathbf{P}_4\}$. The Delta robot has 4 degrees of freedom: 3 translational and 1 rotational. The 3 of the kinematic legs are identical and each of these identical kinematic legs has two consecutive kinematic elements (an upper-leg $[\mathbf{P}_i\mathbf{A}_i]$ and a lower-leg $[\mathbf{A}_i\mathbf{B}_i]$) linked with each other at $\mathbf{A}_i$. Each lower-leg consists of two slim and cylindrical shaped rods fitted with ball-joints (($\mathbf{A}_{i1}, \mathbf{A}_{i2}$) and ($\mathbf{B}_{i1}, \mathbf{B}_{i2}$)), forming a parallelogram. A kinematic leg of the Delta is symbolically noted as $\underline{R} - (S - S)_2$ (this also equals to $\underline{R} - U - U$) where $\underline{R}$ and S stand for an actuated revolute joint and a passive spherical joint, respectively. The parallelograms of the identical kinematic legs restrict the movement of the moving-platform to pure translations in x, y and z axes. From the base, a fourth non-identical kinematic leg $[\mathbf{P}_4\mathbf{E}]$ extends to the middle of the moving-platform to give the end-effector a fourth, rotational degree of freedom around the z-axis $\underline{\mathbf{z}}_e$ of the end-effector frame. Figure A.4 depicts the geometric notation of the kinematic legs and the moving-platform. In modeling, the following notation is used:

- $i = 1, 2, 3$ denotes the identical kinematic legs.
- $j = \{p, a, b\}$ represents literally the kinematic elements in identical kinematic legs.
- $\xi_{geo} = \{\mathbf{O}, \mathbf{P}_i, \ell_{pi}, \ell_{ai}, \ell_{bi}\}$ are the geometric parameters (constant lengths and points).
- $\xi_{dyn} = \{m_{ji}, \mathcal{I}_{ji}, f_{v_i}, f_{c_i}\}$ are the dynamic parameters (weights, inertias and frictions).

– $\mathcal{F}_o = (\mathbf{O}, \underline{\mathbf{x}}_o, \underline{\mathbf{y}}_o, \underline{\mathbf{z}}_o)$, $\mathcal{F}_e = (\mathbf{E}, \underline{\mathbf{x}}_e, \underline{\mathbf{y}}_e, \underline{\mathbf{z}}_e)$, $\mathcal{F}_{pi} = (\mathbf{P}_i, \underline{\mathbf{x}}_{pi}, \underline{\mathbf{y}}_{pi}, \underline{\mathbf{z}}_{pi})$, $\mathcal{F}_{ai} = (\mathbf{A}_i, \underline{\mathbf{x}}_{ai}, \underline{\mathbf{y}}_{ai}, \underline{\mathbf{z}}_{ai})$, $\mathcal{F}_{bi} = (\mathbf{B}_i, \underline{\mathbf{x}}_{bi}, \underline{\mathbf{y}}_{bi}, \underline{\mathbf{z}}_{bi})$ and $\mathcal{F}_4 = (\mathbf{P}_4, \underline{\mathbf{x}}_4, \underline{\mathbf{y}}_4, \underline{\mathbf{z}}_4)$ denote respectively the base, the end-effector, the i^{th} upper-leg, the i^{th} lower-leg, the moving-platform's i^{th} virtual kinematic element and the 4^{th} kinematic leg frames.
– q_i is the articulated position of the i^{th} upper-leg leg.
– d_4 and θ_4 are the length and the rotational angle around itself of the 4^{th} non-identical kinematic leg.
– The end-effector pose is composed of the origin $\mathbf{E}$ of the end-effector frame and rotational angle θ_4 of the 4^{th} non-identical kinematic leg. The end-effector pose velocity is then $\dot{\mathbf{E}}$ and $\dot{\theta}_4$:

$$\mathbb{X} \triangleq \begin{bmatrix} \mathbf{E} \\ \theta_4 \end{bmatrix}, \qquad \dot{\mathbb{X}} \triangleq \begin{bmatrix} \dot{\mathbf{E}} \\ \dot{\theta}_4 \end{bmatrix} \in \Re^{4\times 1} \tag{A.41}$$

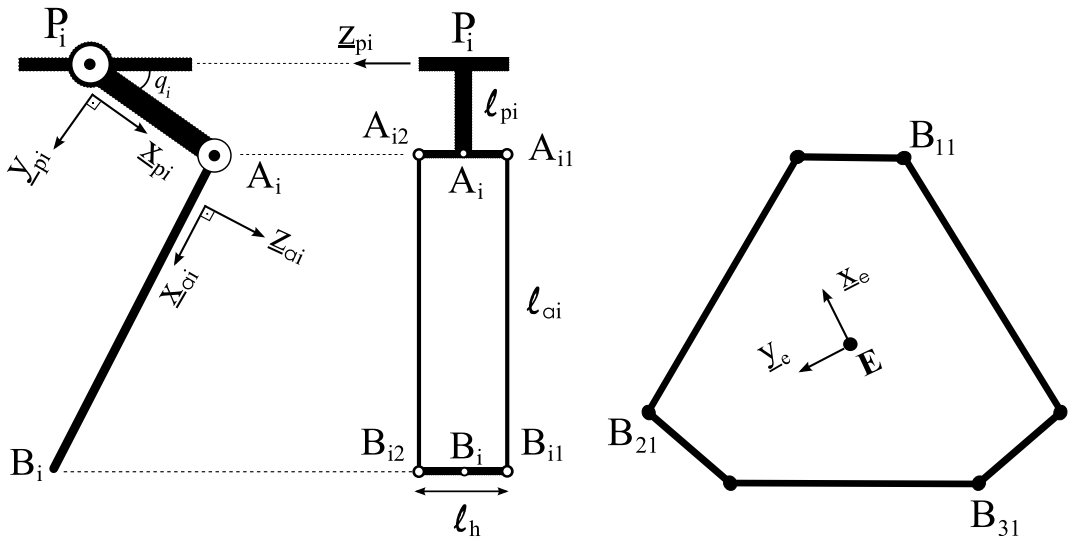


Figure A.4 – Side and front views of a kinematic leg with its variables and parameters (*left*). The plan of the moving-platform with its variables and parameters (*right*).

A.2.2 State Variables

Kinematic Element Types

We have got the following basic parts in the Delta: upper-legs, lower-legs, the fourth non-identical kinematic leg and the moving platform. The configurations of those parts are defined by the new variable set of a kinematic element rather than the non-linear joint coordinates:

Upper-legs $[\mathbf{P}_i\mathbf{A}_i]$ An upper-leg rotates around a fixed axis. It has 1 dof rotational extrinsic mobility due to a (R)evolute joint. It is a *bar type* kinematic element: $\underline{\mathbf{x}}_{pi}$.

Lower-legs $[\mathbf{A}_i\mathbf{B}_i]$ A lower-leg (a parallelogram) rotates around a moving axis. It has 2 dof rotational extrinsic mobility due to the existing (S)pherical joints. It is thus a *bar type* kinematic element: $\underline{\mathbf{x}}_{ai}$.

Fourth-leg $[\mathbf{P}_4\mathbf{E}]$ The fourth non-identical kinematic leg has 3 dof rotational extrinsic mobility and as well as 2 dof intrinsic mobility (1 for translation along its direction and 1 for self-rotation around its direction). It is a *screw type* kinematic element: $\{\underline{\mathbf{x}}_4,\, d_4,\, \theta_4\}$.

Moving-platform $[\mathbf{B}_i\mathbf{E}]$ A virtual kinematic element of the rigid moving-platform only translates in space. It is a *bar type* kinematic element: $\underline{\mathbf{x}}_{bi}$ (moving vector with a fixed direction).

The $\{\underline{\mathbf{x}}_{pi},\, \underline{\mathbf{x}}_{ai},\, \underline{\mathbf{x}}_4,\, d_4,\, \theta_4\}$ are now the new variables that redefine the state of the mechanism.

A.2.3 Kinematics

Mass Centers

Assuming that all the kinematic elements are homogenous and symmetric, the mass center positions are written as follows:

$$\mathbf{S}_{pi} \,=\, \mathbf{P}_i \,+\, \frac{1}{2}\,\ell_{pi}\,\underline{\mathbf{x}}_{pi}\,, \qquad \mathbf{S}_{ai} \,=\, \mathbf{P}_i \,+\, \ell_{pi}\,\underline{\mathbf{x}}_{pi} \,+\, \frac{1}{2}\,\ell_{ai}\,\underline{\mathbf{x}}_{ai}\,, \qquad i = 1,2,3 \tag{A.42}$$

$$\mathbf{S}_{bi} \,=\, \mathbf{P}_i \,+\, \ell_{pi}\,\underline{\mathbf{x}}_{pi} \,+\, \ell_{ai}\,\underline{\mathbf{x}}_{ai} \,+\, \ell_{bi}\,\underline{\mathbf{x}}_{bi}\,, \qquad i = 1,2,3 \tag{A.43}$$

The direction vectors ($\underline{\mathbf{x}}_{bi}$) of the virtual kinematic elements of the moving-platform have fixed orientations. The mass center position of the non-identical kinematic leg is written as below:

$$\mathbf{S}_4 \,=\, \mathbf{P}_4 \,+\, \frac{1}{2}\,d_4\,\underline{\mathbf{x}}_4 \tag{A.44}$$

Velocities

The translational and rotational velocities of kinematic elements are computed as follows:

$$\dot{\mathbf{S}}_{pi} \,=\, \frac{1}{2}\,\ell_{pi}\,\dot{\underline{\mathbf{x}}}_{pi}\,, \qquad \boldsymbol{\omega}_{pi} \triangleq \underline{\mathbf{x}}_{pi} \,\times\, \dot{\underline{\mathbf{x}}}_{pi} \tag{A.45}$$

$$\dot{\mathbf{S}}_{ai} \,=\, \ell_{pi}\,\dot{\underline{\mathbf{x}}}_{pi} \,+\, \frac{1}{2}\,\ell_{ai}\,\dot{\underline{\mathbf{x}}}_{ai}\,, \qquad \boldsymbol{\omega}_{ai} \triangleq \underline{\mathbf{x}}_{ai} \,\times\, \dot{\underline{\mathbf{x}}}_{ai} \tag{A.46}$$

$$\dot{\mathbf{S}}_{bi} \,=\, \ell_{pi}\,\dot{\underline{\mathbf{x}}}_{pi} \,+\, \ell_{ai}\,\dot{\underline{\mathbf{x}}}_{ai}\,, \qquad \boldsymbol{\omega}_{bi} = \mathbf{0} \tag{A.47}$$

$$\dot{\mathbf{S}}_4 \,=\, \frac{1}{2}\,\left(\,\dot{d}_4\,\underline{\mathbf{x}}_4 \,+\, d_4\,\dot{\underline{\mathbf{x}}}_4\,\right)\,, \qquad \boldsymbol{\omega}_4 = \underline{\mathbf{x}}_4 \,\times\, \dot{\underline{\mathbf{x}}}_4 \,+\, \dot{\theta}_4\,\underline{\mathbf{x}}_4 \tag{A.48}$$

Accelerations

The translational and rotational accelerations of kinematic elements are computed as below:

$$\ddot{\mathbf{S}}_{pi} \,=\, \frac{1}{2}\,\ell_{pi}\,\ddot{\underline{\mathbf{x}}}_{pi}\,, \qquad \dot{\boldsymbol{\omega}}_{pi} \triangleq \underline{\mathbf{x}}_{pi} \,\times\, \ddot{\underline{\mathbf{x}}}_{pi} \tag{A.49}$$

$$\ddot{\mathbf{S}}_{ai} \,=\, \ell_{pi}\,\ddot{\underline{\mathbf{x}}}_{pi} \,+\, \frac{1}{2}\,\ell_{ai}\,\ddot{\underline{\mathbf{x}}}_{ai}\,, \qquad \dot{\boldsymbol{\omega}}_{ai} \triangleq \underline{\mathbf{x}}_{ai} \,\times\, \ddot{\underline{\mathbf{x}}}_{ai} \tag{A.50}$$

$$\ddot{\mathbf{S}}_{bi} \,=\, \ell_{pi}\,\ddot{\underline{\mathbf{x}}}_{pi} \,+\, \ell_{ai}\,\ddot{\underline{\mathbf{x}}}_{ai}\,, \qquad \dot{\boldsymbol{\omega}}_{bi} = \mathbf{0} \tag{A.51}$$

$$\ddot{\mathbf{S}}_4 \,=\, \frac{1}{2}\,\left(\,\ddot{d}_4\,\underline{\mathbf{x}}_4 \,+\, 2\,\dot{d}_4\,\dot{\underline{\mathbf{x}}}_4 \,+\, d_4\,\ddot{\underline{\mathbf{x}}}_4\,\right)\,, \qquad \dot{\boldsymbol{\omega}}_4 = \underline{\mathbf{x}}_4 \,\times\, \ddot{\underline{\mathbf{x}}}_4 \,+\, \ddot{\theta}_4\,\underline{\mathbf{x}}_4 \,+\, \dot{\theta}_4\,\dot{\underline{\mathbf{x}}}_4 \tag{A.52}$$

A.2.4 Kinematic Constraints

Assuming that the end-effector frame is located at the mass center position of the moving platform, the closed-loop constraint equation for each of the identical kinematic legs can be written as follow:

$$\overrightarrow{\mathbf{OE}} - \ell_{bi}\,\underline{\mathbf{x}}_{bi} - \ell_{ai}\,\underline{\mathbf{x}}_{ai} - \ell_{pi}\,\underline{\mathbf{x}}_{pi} - \overrightarrow{\mathbf{OP}_i} = \mathbf{0}\,, \qquad i = 1,2,3 \tag{A.53}$$

where $\mathbf{O}$, $\mathbf{P}_i$ and $(\ell_{bi}\,\underline{\mathbf{x}}_{bi})$ are constants. Afterwards, one can differentiate the last closed-loop constraint equation with respect to time in order to obtain the motion constraint equation, which yields:

$$\dot{\mathbf{E}} - \ell_{ai}\,\dot{\underline{\mathbf{x}}}_{ai} - \ell_{pi}\,\dot{\underline{\mathbf{x}}}_{pi} = \mathbf{0} \tag{A.54}$$

Constraints on the Active Upper-Legs

Constraints on the orientations of the active upper-legs are computed as follows:

$$\underline{\mathbf{x}}_{ai}^T\,\dot{\mathbf{E}} - \ell_{pi}\,\underline{\mathbf{x}}_{ai}^T\,\dot{\underline{\mathbf{x}}}_{pi} = \mathbf{0} \tag{A.55}$$

$$\underline{\mathbf{y}}_{pi}\,\underline{\mathbf{x}}_{ai}^T\,\dot{\mathbf{E}} - \ell_{pi}\,\underline{\mathbf{y}}_{pi}\,\underline{\mathbf{x}}_{ai}^T\,\dot{\underline{\mathbf{x}}}_{pi} = \mathbf{0} \tag{A.56}$$

$$\underline{\mathbf{y}}_{pi}\,\underline{\mathbf{x}}_{ai}^T\,\dot{\mathbf{E}} - \ell_{pi}\,\underline{\mathbf{x}}_{ai}^T\,\underline{\mathbf{y}}_{pi}\,\dot{\underline{\mathbf{x}}}_{pi} = \mathbf{0} \tag{A.57}$$

$$\dot{\underline{\mathbf{x}}}_{pi} = D_{pi}\,\dot{\mathbf{E}}\,, \qquad D_{pi} = \left[\frac{\underline{\mathbf{y}}_{pi}\,\underline{\mathbf{x}}_{ai}^T}{\ell_{pi}\,\underline{\mathbf{x}}_{ai}^T\,\underline{\mathbf{y}}_{pi}}\right] \in \Re^{3\times 3} \tag{A.58}$$

$$\dot{\underline{\mathbf{x}}}_{pi} = M_{pi}\,\dot{\mathbb{X}}\,, \qquad M_{pi} = \left[\begin{array}{cc} D_{pi} & \mathbf{0}_{3\times 1} \end{array}\right] \in \Re^{3\times 4} \tag{A.59}$$

Constraints on the Passive Lower-Legs

Constraints on the orientations of the passive lower-legs are computed as follows:

$$\dot{\mathbf{E}} - \ell_{ai}\,\dot{\underline{\mathbf{x}}}_{ai} - \ell_{pi}\,D_{pi}\,\dot{\mathbf{E}} = \mathbf{0} \tag{A.60}$$

$$\dot{\underline{\mathbf{x}}}_{ai} = D_{ai}\,\dot{\mathbf{E}}\,, \qquad D_{ai} = \left[\frac{1}{\ell_{ai}}\,(\,I_3 - \ell_{pi}\,D_{pi}\,)\right] \in \Re^{3\times 3} \tag{A.61}$$

$$\dot{\underline{\mathbf{x}}}_{ai} = M_{ai}\,\dot{\mathbb{X}}\,, \qquad M_{ai} = \left[\begin{array}{cc} D_{ai} & \mathbf{0}_{3\times 1} \end{array}\right] \in \Re^{3\times 4} \tag{A.62}$$

Constraints on the Active Fourth Kinematic-Leg

Constraint on the variation of the length of the 4^{th} kinematic leg is computed as follows:

$$\overrightarrow{\mathbf{OE}} - d_4\,\underline{\mathbf{x}}_4 - \overrightarrow{\mathbf{OP}_4} = \mathbf{0} \tag{A.63}$$

$$\dot{\mathbf{E}} - \dot{d}_4\,\underline{\mathbf{x}}_4 - d_4\,\dot{\underline{\mathbf{x}}}_4 = \mathbf{0} \tag{A.64}$$

$$\underline{\mathbf{x}}_4^T\,\dot{\mathbf{E}} - \dot{d}_4 = 0 \tag{A.65}$$

$$\dot{d}_4 = \underline{\mathbf{x}}_4^T \dot{\mathbf{E}} \tag{A.66}$$

$$\dot{d}_4 \;=\; M_{d_4}\,\dot{\mathbb{X}}\,, \qquad M_{d_4} \;=\; \left[\begin{array}{cc} \underline{\mathbf{x}}_4^T & 0 \end{array}\right] \in \Re^{1\times 4} \tag{A.67}$$

Constraint on the orientation of the 4^{th} kinematic leg is computed as follows:

$$\dot{\mathbf{E}} \;-\; \underline{\mathbf{x}}_4\,\underline{\mathbf{x}}_4^T\,\dot{\mathbf{E}} \;-\; d_4\,\dot{\underline{\mathbf{x}}}_4 \;=\; \mathbf{0} \tag{A.68}$$

$$\dot{\underline{\mathbf{x}}}_4 \;=\; D_4\,\dot{\mathbf{E}}\,, \qquad D_4 \;=\; \left[\frac{1}{d_4}\,\left(I_3 \;-\; \underline{\mathbf{x}}_4\,\underline{\mathbf{x}}_4^T\right)\right] \in \Re^{3\times 3} \tag{A.69}$$

$$\dot{\underline{\mathbf{x}}}_4 \;=\; M_4\,\dot{\mathbb{X}}\,, \qquad M_4 \;=\; \left[\begin{array}{cc} D_4 & \mathbf{0}_{3\times 1} \end{array}\right] \in \Re^{3\times 4} \tag{A.70}$$

Since the rotation of the end-effector directly comes from the 4^{th} kinematic leg, then the self-rotation of the 4^{th} kinematic leg is computed as follows:

$$\dot{\theta}_4 \;=\; M_{\theta_4}\,\dot{\mathbb{X}}\,, \qquad M_{\theta_4} \;=\; \left[\begin{array}{cc} \mathbf{0}_{1\times 3} & 1 \end{array}\right] \in \Re^{1\times 4} \tag{A.71}$$

Constraints on the Active Joints

Constraints on the active joints coordinates are computed as follows:

$$\dot{\underline{\mathbf{x}}}_{pi} \;=\; \dot{q}_i\,\underline{\mathbf{z}}_{pi} \;\times\; \underline{\mathbf{x}}_{pi} \;=\; \dot{q}_i\,\underline{\mathbf{y}}_{pi} \tag{A.72}$$

$$\underline{\mathbf{y}}_{pi}^T\,\dot{\underline{\mathbf{x}}}_{pi} \;=\; \dot{q}_i \tag{A.73}$$

$$\dot{q}_i \;=\; M_{q_i}\,\dot{\mathbb{X}}\,, \qquad M_{q_i} \;=\; \left[\begin{array}{cc} \underline{\mathbf{y}}_{pi}^T\,D_{pi} & 0 \end{array}\right] \in \Re^{1\times 4} \tag{A.74}$$

A.2.5 Kinematic Coordinates

Equations (A.75), (A.76) and Tables (A.3), (A.4) show the motion basis and the kinematic coordinates of the mechanism, respectively.

$$\mathbf{u}_{i1} \;=\; \dot{\underline{\mathbf{x}}}_{pi}\,, \qquad \mathbf{u}_{i2} \;=\; \dot{\underline{\mathbf{x}}}_{ai}\,, \qquad i \;=\; 1,2,3 \tag{A.75}$$

$$\mathbf{u}_{x_4} \;=\; \dot{\underline{\mathbf{x}}}_4\,, \qquad u_{d_4} \;=\; \dot{d}_4\,, \qquad u_{\theta_4} \;=\; \dot{\theta}_4 \tag{A.76}$$

Table A.3 – The (transposed) kinematic coordinates of the Delta parallel robot, i=1,2,3.

	$\partial\,\dot{\mathbf{S}}_{pi}$	$\partial\,\boldsymbol{\omega}_{pi}$	$\partial\,\dot{\mathbf{S}}_{ai}$	$\partial\,\boldsymbol{\omega}_{ai}$	$\partial\,\dot{\mathbf{S}}_{bi}$	$\partial\,\boldsymbol{\omega}_{bi}$
$\partial\,\dot{\underline{\mathbf{x}}}_{pi}$	$\frac{1}{2}\,\ell_{pi}\,I_3$	$[\underline{\mathbf{x}}_{pi}]_\times^T$	$\ell_{pi}\,I_3$	$\mathbf{0}_{3\times 3}$	$\ell_{pi}\,I_3$	$\mathbf{0}_{3\times 3}$
$\partial\,\dot{\underline{\mathbf{x}}}_{ai}$	$\mathbf{0}_{3\times 3}$	$\mathbf{0}_{3\times 3}$	$\frac{1}{2}\,\ell_{ai}\,I_3$	$[\underline{\mathbf{x}}_{ai}]_\times^T$	$\ell_{ai}\,I_3$	$\mathbf{0}_{3\times 3}$

A.2.6 Dynamic Coordinates

Listing Active/Reactive Forces and Torques

Tables A.5 and A.6 tabulate the local forces and torques of the Delta parallel robot.

Table A.4 – The (transposed) kinematic coordinates of the 4^{th} leg of the Delta parallel robot.

	$\partial\,\dot{\mathbf{S}}_4$	$\partial\,\boldsymbol{\omega}_4$
$\partial\,\dot{\underline{\mathbf{x}}}_4$	$\frac{1}{2}\,d_4\,I_3$	$[\,\underline{\mathbf{x}}_4\,]^T_\times$
$\partial\,\dot{d}_4$	$\frac{1}{2}\,\underline{\mathbf{x}}_4^T$	$\mathbf{0}_{3\times1}$
$\partial\,\dot{\theta}_4$	$\mathbf{0}_{3\times1}$	$\underline{\mathbf{x}}_4^T$

Table A.5 – The local forces and torques of the Delta parallel robot, i=1,2,3.

	Active		*Friction*		*Inertia**	
	Actuator	*Gravity*	*Actuator*	*Passive Joint*	*Actuator*	*Element*
Forces (pi)	$\mathbf{0}$	$\mathbf{f}_{\mathbf{g}(pi)}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{f}^*_{pi}$
Torques (pi)	$\tau_{\underline{\mathbf{x}}_{pi}}\,\underline{\mathbf{z}}_{pi}$	$\mathbf{0}$	$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{pi}}$	$\mathbf{0}$	$\boldsymbol{\tau}^*_{\underline{\mathbf{x}}_{pi}}$	$\boldsymbol{\tau}^*_{pi}$
Forces (ai)	$\mathbf{0}$	$\mathbf{f}_{\mathbf{g}(ai)}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{f}^*_{ai}$
Torques (ai)	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{ai}}$	$\mathbf{0}$	$\boldsymbol{\tau}^*_{ai}$
Forces (bi)	$\mathbf{0}$	$\mathbf{f}_{\mathbf{g}(bi)}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{f}^*_{bi}$
Torques (bi)	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{bi}}$	$\mathbf{0}$	$\mathbf{0}$

Computing Dynamic Coordinates

So as to eliminate the non-contributing forces, the dynamic coordinates are computed through the matrix-wise multiplication of the Tables A.3 (transposed kinematic coordinates) and A.5 (sum of the local forces and torques).

$$\begin{bmatrix} \mathbb{F}_{\underline{x}_{pi}} \\ \mathbb{F}_{\underline{x}_{ai}} \end{bmatrix} = \begin{bmatrix} Kinematic \\ Coordinates \\ Table\ A.3 \end{bmatrix}_{(2\times6)} \begin{bmatrix} Sum\ of \\ Forces \\ Torques \\ Table\ A.5 \end{bmatrix}_{(6\times1)} \tag{A.77}$$

which can be explicitly written as follows:

$$\begin{bmatrix} \mathbb{F}_{\underline{x}_{pi}} \\ \mathbb{F}_{\underline{x}_{ai}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\,\ell_{pi}\,I_3 & [\,\underline{\mathbf{x}}_{pi}\,]^T_\times & \ell_{pi}\,I_3 & \mathbf{0}_{3\times3} & \ell_{pi}\,I_3 & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \frac{1}{2}\,\ell_{ai}\,I_3 & [\,\underline{\mathbf{x}}_{ai}\,]^T_\times & \ell_{ai}\,I_3 & \mathbf{0}_{3\times3} \end{bmatrix} \begin{bmatrix} \mathbf{f}_{\mathbf{g}(pi)} + \mathbf{f}^*_{pi} \\ \tau_{\underline{\mathbf{x}}_{pi}}\,\underline{\mathbf{z}}_{pi} + \widetilde{\boldsymbol{\tau}}_{pi} \\ \mathbf{f}_{\mathbf{g}(ai)} + \mathbf{f}^*_{ai} \\ \bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{ai}} + \boldsymbol{\tau}^*_{ai} \\ \mathbf{f}_{\mathbf{g}(bi)} + \mathbf{f}^*_{bi} \\ \bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{bi}} \end{bmatrix} \tag{A.78}$$

Table A.6 – The local forces and torques of the 4^{th} leg of the Delta parallel robot.

	$Active$		$Friction$		$Inertia^*$	
	$Actuator$	$Gravity$	$Actuator$	$Passive\,Joint$	$Actuator$	$Element$
$Forces\,(4)$	$\mathbf{0}$	$\mathbf{f}_{\mathbf{g}(4)}$	$\mathbf{0}$	$\bar{\mathbf{f}}_{d_4}$	$\mathbf{0}$	$\mathbf{f}_4^*$
$Torques\,(4)$	$\tau_{\theta_4}\ \underline{\mathbf{x}}_4$	$\mathbf{0}$	$\bar{\boldsymbol{\tau}}_{\theta_4}$	$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_4}$	$\boldsymbol{\tau}_{\theta_4}^*$	$\boldsymbol{\tau}_4^*$

where

$$\widetilde{\boldsymbol{\tau}}_{pi} \;=\; \bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{pi}} \;+\; \boldsymbol{\tau}_{\underline{\mathbf{x}}_{pi}}^* \;+\; \boldsymbol{\tau}_{pi}^* \tag{A.79}$$

The dynamic coordinates for the 4^{th} kinematic leg of the Delta are computed with the matrix-wise multiplication of the Tables A.4 and A.6:

$$\begin{bmatrix} \mathbb{F}_{\underline{x}_4} \\ \mathbb{F}_{d_4} \\ \mathbb{F}_{\theta_4} \end{bmatrix} \;=\; \begin{bmatrix} Kinematic \\ Coordinates \\ Table\ A.4 \end{bmatrix}_{(3\times 2)} \begin{bmatrix} Sum\ of \\ Forces \\ Torques \\ Table\ A.6 \end{bmatrix}_{(2\times 1)} \tag{A.80}$$

which can be explicitly written as follows:

$$\begin{bmatrix} \mathbb{F}_{\underline{x}_4} \\ \mathbb{F}_{d_4} \\ \mathbb{F}_{\theta_4} \end{bmatrix} \;=\; \begin{bmatrix} \frac{1}{2}\, d_4\, I_3 & [\,\underline{\mathbf{x}}_4\,]_\times^T \\ \\ \frac{1}{2}\, \underline{\mathbf{x}}_4^T & \mathbf{0}_{3\times 1} \\ \\ \mathbf{0}_{3\times 1} & \underline{\mathbf{x}}_4^T \end{bmatrix} \begin{bmatrix} \mathbf{f}_{\mathbf{g}(4)} \;+\; \bar{\mathbf{f}}_{d_4} \;+\; \mathbf{f}_4^* \\ \\ \tau_{\theta_4}\ \underline{\mathbf{x}}_4 \;+\; \bar{\boldsymbol{\tau}}_{\theta_4} \;+\; \bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_4} \;+\; \boldsymbol{\tau}_{\theta_4}^* \;+\; \boldsymbol{\tau}_4^* \end{bmatrix} \tag{A.81}$$

A.2.7 Dynamic Constraints

Exploiting (2.110), the dynamic constraints of the Delta robot are written as follows:

$$\mathbf{0}_{\,4\times 1} \;=\; \begin{bmatrix} M_p^T & M_{\theta_4}^T \end{bmatrix} \begin{bmatrix} \mathbb{F}_p \\ \mathbb{F}_{\theta_4} \end{bmatrix} \;+\; M_a^T\, \mathbb{F}_a \;+\; M_4^T\, \mathbb{F}_{\underline{x}_4} \;+\; M_{d_4}^T\, \mathbb{F}_{d_4} \tag{A.82}$$

where $\mathbb{F}_p \in \Re^{9\times 1}$ and $\mathbb{F}_a \in \Re^{9\times 1}$ are the stacked vectors of the dynamic coordinates:

$$\mathbb{F}_p \;=\; \begin{bmatrix} \mathbb{F}_{\underline{x}_{p1}} \\ \vdots \\ \mathbb{F}_{\underline{x}_{p3}} \end{bmatrix}, \qquad\qquad \mathbb{F}_a \;=\; \begin{bmatrix} \mathbb{F}_{\underline{x}_{a1}} \\ \vdots \\ \mathbb{F}_{\underline{x}_{a3}} \end{bmatrix} \tag{A.83}$$

and where $\mathbb{F}_{\underline{x}_4} \in \Re^{3\times 1}$, $\mathbb{F}_{d_4} \in \Re^{1\times 1}$ and $\mathbb{F}_{\theta_4} \in \Re^{1\times 1}$ are brought from (A.81). The $M_a \in \Re^{9\times 4}$ and $M_p \in \Re^{9\times 4}$ are the stacked matrices of the inverse differential kinematic models $M_{ai} \in \Re^{3\times 4}$ and $M_{pi} \in \Re^{3\times 4}$, respectively. The $M_4 \in \Re^{3\times 4}$, $M_{d_4} \in \Re^{1\times 4}$ and $M_{\theta_4} \in \Re^{1\times 4}$ are brought from (A.70), (A.67) and (A.71), respectively.

A.3 The 3RRR Robot

A.3.1 Geometry and Notation

The 3-RRR consists of 3 kinematic legs interconnecting a moving-platform to a fixed base. The moving-platform is a single triangular rigid body $\{\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3\}$. Figure A.5 shows the 3-RRR planar parallel robot and its graphical layout.

Figure A.5 – The 3-RRR planar parallel robot (*left*) and its graphical layout (*right*).

The 3-RRR is designed symmetrically, that is to say, the base and the moving platform are equilateral triangles as well as the kinematic legs are identical. Each kinematic leg has 2 consecutive kinematic elements (an upper-leg $[\mathbf{P}_i\mathbf{A}_i]$ and a lower-leg $[\mathbf{A}_i\mathbf{B}_i]$) linked with each other at $\mathbf{A}_i$. A kinematic leg of 3-RRR symbolically is noted as $\underline{R}-R-R$ where $\underline{R}$ and R stand for an actuated revolute joint and a passive revolute joint, respectively. The moving-platform has 3 degrees of freedom: 2 translational movements in x and y axes and 1 rotational movement around the z axis. All the kinematic legs are actuated from the base by revolute motors located at $\{\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3\}$. Figure A.6 depicts the geometrical notation of the mechanism. In modeling, the following notation is used:

- $i = 1, 2, 3$ denotes the kinematic legs.
- $j = \{p, a, b\}$ is the literal representation of the kinematic elements.
- $\xi_{geo} = \{\mathbf{O}, \mathbf{P}_i, \ell_{pi}, \ell_{ai}, \ell_{bi}, \alpha_i, \beta_i\}$ are geometric parameters (constant lengths, points).
- $\xi_{dyn} = \{m_{ji}, \mathcal{I}_{ji}, f_{v_i}, f_{c_i}\}$ are the dynamic parameters (weights, inertias and frictions).
- $\mathcal{F}_o = (\mathbf{O}, \underline{\mathbf{x}}_o, \underline{\mathbf{y}}_o, \underline{\mathbf{z}}_o)$, $\mathcal{F}_e = (\mathbf{E}, \underline{\mathbf{x}}_e, \underline{\mathbf{y}}_e, \underline{\mathbf{z}}_e)$, $\mathcal{F}_{pi} = (\mathbf{P}_i, \underline{\mathbf{x}}_{pi}, \underline{\mathbf{y}}_{pi}, \underline{\mathbf{z}}_{pi})$, $\mathcal{F}_{ai} = (\mathbf{A}_i, \underline{\mathbf{x}}_{ai}, \underline{\mathbf{y}}_{ai}, \underline{\mathbf{z}}_{ai})$ and $\mathcal{F}_{bi} = (\mathbf{B}_i, \underline{\mathbf{x}}_{bi}, \underline{\mathbf{y}}_{bi}, \underline{\mathbf{z}}_{bi})$ denote respectively the base, the end-effector, the i^{th} upper-leg, the i^{th} lower-leg, the moving-platform's i^{th} virtual kinematic element frames.
- q_i is the articulated position of the i^{th} upper-leg leg.
- The end-effector pose is composed of the origin $\mathbf{E}$ and x axis $\underline{\mathbf{x}}_e$ of the end-effector

frame. The end-effector pose velocity is then $\dot{\mathbf{E}}$ and $\dot{\underline{\mathbf{x}}}_e$:

$$\mathbb{X} \triangleq \begin{bmatrix} \mathbf{E} \\ \underline{\mathbf{x}}_e \end{bmatrix}, \qquad \dot{\mathbb{X}} \triangleq \begin{bmatrix} \dot{\mathbf{E}} \\ \dot{\underline{\mathbf{x}}}_e \end{bmatrix} \in \Re^{6\times 1} \tag{A.84}$$

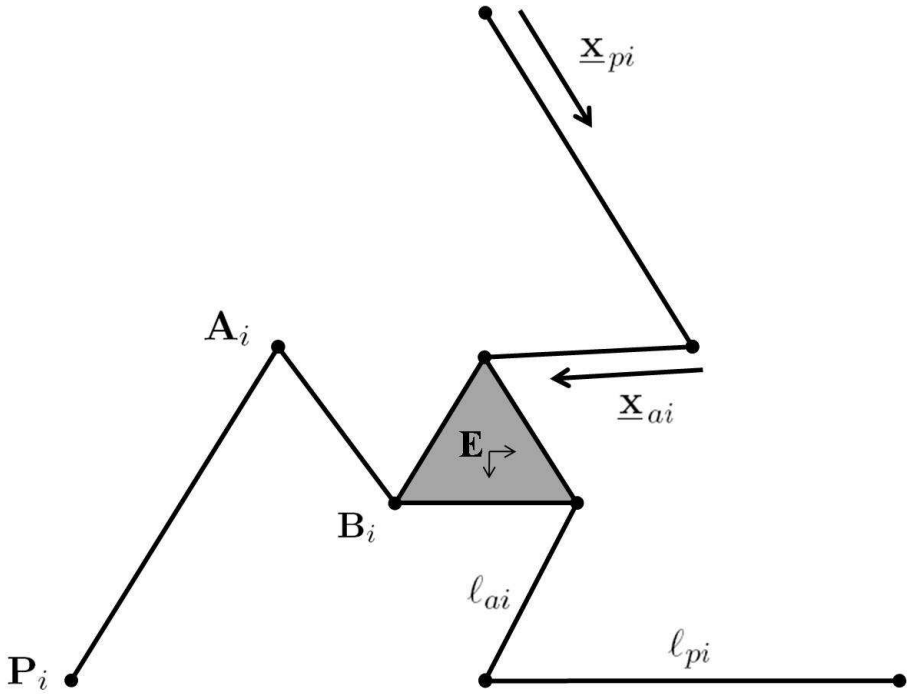


Figure A.6 – The notation of the 3-RRR planar parallel robot.

A.3.2 State Variables

Kinematic Element Types

We have got the following basic parts in the 3-RRR: upper-legs, lower-legs and the moving platform. The configurations of those parts are defined by the proposed new variable set of a kinematic element rather than the non-linear joint coordinates:

Upper-legs $[\mathbf{P}_i\mathbf{A}_i]$ An upper-leg rotates around a fixed axis. It has 1 dof rotational extrinsic mobility due to a (R)evolute joint. It is a *bar type* kinematic element: $\underline{\mathbf{x}}_{pi}$.

Lower-legs $[\mathbf{A}_i\mathbf{B}_i]$ A lower-leg rotates around a fixed axis. It has 1 dof rotational extrinsic mobility due to a (R)evolute joint. It is a *bar type* kinematic element: $\underline{\mathbf{x}}_{ai}$.

Moving-platform $[\mathbf{B}_i\mathbf{E}]$ A virtual kinematic element of the rigid moving-platform rotates around a fixed axis. It has 1 dof rotational extrinsic mobility due to a (R)evolute joint. It is a *bar type* kinematic element: $\underline{\mathbf{x}}_{bi}$.

The $\{\underline{\mathbf{x}}_{pi}, \underline{\mathbf{x}}_{ai}, \underline{\mathbf{x}}_{bi}\}$ are now the new variables that redefine the state of the mechanism.

A.3.3 Kinematics

Mass Centers

Assuming that all the kinematic elements are homogenous and symmetric, the mass center positions are written as follows:

$$\mathbf{S}_{pi} \;=\; \mathbf{P}_i \;+\; \frac{1}{2}\,\ell_{pi}\,\underline{\mathbf{x}}_{pi} \tag{A.85}$$

$$\mathbf{S}_{ai} \;=\; \mathbf{P}_i \;+\; \ell_{pi}\,\underline{\mathbf{x}}_{pi} \;+\; \frac{1}{2}\,\ell_{ai}\,\underline{\mathbf{x}}_{ai} \tag{A.86}$$

$$\mathbf{S}_{bi} \;=\; \mathbf{P}_i \;+\; \ell_{pi}\,\underline{\mathbf{x}}_{pi} \;+\; \ell_{ai}\,\underline{\mathbf{x}}_{ai} \;+\; \ell_{bi}\,\underline{\mathbf{x}}_{bi} \tag{A.87}$$

Velocities

The translational and rotational velocities of the kinematic elements are computed as below:

$$\dot{\mathbf{S}}_{pi} \;=\; \frac{1}{2}\,\ell_{pi}\,\dot{\underline{\mathbf{x}}}_{pi}\,, \qquad\qquad \boldsymbol{\omega}_{pi} \triangleq \underline{\mathbf{x}}_{pi} \times \dot{\underline{\mathbf{x}}}_{pi} \tag{A.88}$$

$$\dot{\mathbf{S}}_{ai} \;=\; \ell_{pi}\,\dot{\underline{\mathbf{x}}}_{pi} \;+\; \frac{1}{2}\,\ell_{ai}\,\dot{\underline{\mathbf{x}}}_{ai}\,, \qquad\qquad \boldsymbol{\omega}_{ai} \triangleq \underline{\mathbf{x}}_{ai} \times \dot{\underline{\mathbf{x}}}_{ai} \tag{A.89}$$

$$\dot{\mathbf{S}}_{bi} \;=\; \ell_{pi}\,\dot{\underline{\mathbf{x}}}_{pi} \;+\; \ell_{ai}\,\dot{\underline{\mathbf{x}}}_{ai} \;+\; \ell_{bi}\,\dot{\underline{\mathbf{x}}}_{bi}\,, \qquad \boldsymbol{\omega}_{bi} \triangleq \underline{\mathbf{x}}_{bi} \times \dot{\underline{\mathbf{x}}}_{bi} \tag{A.90}$$

Accelerations

The translational and rotational accelerations of the kinematic elements are computed as follows:

$$\ddot{\mathbf{S}}_{pi} \;=\; \frac{1}{2}\,\ell_{pi}\,\ddot{\underline{\mathbf{x}}}_{pi}\,, \qquad\qquad \dot{\boldsymbol{\omega}}_{pi} \triangleq \underline{\mathbf{x}}_{pi} \times \ddot{\underline{\mathbf{x}}}_{pi} \tag{A.91}$$

$$\ddot{\mathbf{S}}_{ai} \;=\; \ell_{pi}\,\ddot{\underline{\mathbf{x}}}_{pi} \;+\; \frac{1}{2}\,\ell_{ai}\,\ddot{\underline{\mathbf{x}}}_{ai}\,, \qquad\qquad \dot{\boldsymbol{\omega}}_{ai} \triangleq \underline{\mathbf{x}}_{ai} \times \ddot{\underline{\mathbf{x}}}_{ai} \tag{A.92}$$

$$\ddot{\mathbf{S}}_{bi} \;=\; \ell_{pi}\,\ddot{\underline{\mathbf{x}}}_{pi} \;+\; \ell_{ai}\,\ddot{\underline{\mathbf{x}}}_{ai} \;+\; \ell_{bi}\,\ddot{\underline{\mathbf{x}}}_{bi}\,, \qquad \dot{\boldsymbol{\omega}}_{bi} \triangleq \underline{\mathbf{x}}_{bi} \times \ddot{\underline{\mathbf{x}}}_{bi} \tag{A.93}$$

A.3.4 Kinematic Constraints

Assuming that the end-effector frame is located at the mass center position of the moving platform, the closed-loop constraint equation for each of the kinematic legs can be written as follow:

$$\overrightarrow{\mathbf{OE}} \;-\; \ell_{bi}\,\underline{\mathbf{x}}_{bi} \;-\; \ell_{ai}\,\underline{\mathbf{x}}_{ai} \;-\; \ell_{pi}\,\underline{\mathbf{x}}_{pi} \;-\; \overrightarrow{\mathbf{OP}_i} \;=\; \mathbf{0}\,, \qquad i = 1,2,3 \tag{A.94}$$

where $\mathbf{O}$ and $\mathbf{P}_i$ are constants. Afterwards, one can differentiate the last closed-loop constraint equation with respect to time in order to obtain the motion constraint equation, which yields:

$$\dot{\mathbf{E}} \;-\; \ell_{bi}\,\dot{\underline{\mathbf{x}}}_{bi} \;-\; \ell_{ai}\,\dot{\underline{\mathbf{x}}}_{ai} \;-\; \ell_{pi}\,\dot{\underline{\mathbf{x}}}_{pi} \;=\; \mathbf{0} \tag{A.95}$$

The motion constraints for the attachment points $\mathbf{B}_i$ of the moving-platform can be derived from (A.95) as below:

$$\underline{\mathbf{x}}_{bi} \;=\; \alpha_i\,\underline{\mathbf{x}}_e \;+\; \beta_i\,\underline{\mathbf{y}}_e \tag{A.96}$$

$$\dot{\underline{\mathbf{x}}}_{bi} \;=\; \alpha_i\,\dot{\underline{\mathbf{x}}}_e \;+\; \beta_i\,\dot{\underline{\mathbf{y}}}_e \tag{A.97}$$

$$\dot{\underline{\mathbf{y}}}_e \;\triangleq\; \boldsymbol{\omega}_e \times \underline{\mathbf{y}}_e \;\triangleq\; (\,\underline{\mathbf{x}}_e \times \dot{\underline{\mathbf{x}}}_e\,) \times \underline{\mathbf{y}}_e \;\triangleq\; \underline{\mathbf{z}}_e \times \dot{\underline{\mathbf{x}}}_e \tag{A.98}$$

$$\dot{\underline{\mathbf{x}}}_{bi} \;=\; \alpha_i\,\dot{\underline{\mathbf{x}}}_e \;+\; \beta_i\,\underline{\mathbf{z}}_e \times \dot{\underline{\mathbf{x}}}_e \tag{A.99}$$

$$\dot{\mathbf{E}} \;-\; \ell_{bi}\,\dot{\underline{\mathbf{x}}}_{bi} \;=\; \dot{\mathbf{B}}_i \tag{A.100}$$

$$\dot{\mathbf{B}}_i \;=\; L_{Bi}\,\dot{\mathbb{X}}, \qquad L_{Bi} \;=\; \left[\; I_3 \quad -\ell_{bi}\,(\,\alpha_i\,I_3 \;+\; \beta_i\,[\underline{\mathbf{z}}_e]_\times\,)\;\right] \in \Re^{3\times 6} \tag{A.101}$$

where L_{Bi} is the relation between the Cartesian velocity of the terminal point of the i^{th} kinematic leg and the end-effector pose velocity $\dot{\mathbb{X}}$.

Constraints on the Active Upper-Legs

Constraints on the orientations of the active upper-legs are computed as follows:

$$\ell_{pi}\,\dot{\underline{\mathbf{x}}}_{pi} \;+\; \ell_{ai}\,\dot{\underline{\mathbf{x}}}_{ai} \;=\; L_{Bi}\,\dot{\mathbb{X}} \tag{A.102}$$

$$\ell_{pi}\,\underline{\mathbf{x}}_{ai}^T\,\dot{\underline{\mathbf{x}}}_{pi} \;=\; \underline{\mathbf{x}}_{ai}^T\,L_{Bi}\,\dot{\mathbb{X}} \tag{A.103}$$

$$\ell_{pi}\,\underline{\mathbf{y}}_{pi}\,\underline{\mathbf{x}}_{ai}^T\,\dot{\underline{\mathbf{x}}}_{pi} \;=\; \underline{\mathbf{y}}_{pi}\,\underline{\mathbf{x}}_{ai}^T\,L_{Bi}\,\dot{\mathbb{X}} \tag{A.104}$$

$$\ell_{pi}\,\underline{\mathbf{x}}_{ai}^T\,\underline{\mathbf{y}}_{pi}\,\dot{\underline{\mathbf{x}}}_{pi} \;=\; \underline{\mathbf{y}}_{pi}\,\underline{\mathbf{x}}_{ai}^T\,L_{Bi}\,\dot{\mathbb{X}} \tag{A.105}$$

$$\dot{\underline{\mathbf{x}}}_{pi} \;=\; M_{pi}\,\dot{\mathbb{X}}, \qquad M_{pi} \;=\; \left[\; \frac{\underline{\mathbf{y}}_{pi}\,\underline{\mathbf{x}}_{ai}^T}{\ell_{pi}\,\underline{\mathbf{x}}_{ai}^T\,\underline{\mathbf{y}}_{pi}}\;L_{Bi}\right] \in \Re^{3\times 6} \tag{A.106}$$

Constraints on the Passive Lower-Legs

Constraints on the orientations of the passive lower-legs are computed as follows:

$$\ell_{pi}\,M_{pi}\,\dot{\mathbb{X}} \;+\; \ell_{ai}\,\dot{\underline{\mathbf{x}}}_{ai} \;=\; L_{Bi}\,\dot{\mathbb{X}} \tag{A.107}$$

$$\dot{\underline{\mathbf{x}}}_{ai} \;=\; M_{ai}\,\dot{\mathbb{X}}, \qquad M_{ai} \;=\; \left[\; \frac{1}{\ell_{ai}}\;(\,L_{Bi} \;-\; \ell_{pi}\,M_{pi}\,)\right] \in \Re^{3\times 6} \tag{A.108}$$

Constraints on the Passive Moving-Platform Virtual Kinematic Elements

Constraints on the orientations of the virtual axis type kinematic elements of the moving-platform are computed as follows:

$$\dot{\underline{\mathbf{x}}}_{bi} \;=\; M_{bi}\,\dot{\mathbb{X}}, \qquad M_{bi} \;=\; \left[\; \mathbf{0}_3 \quad (\,\alpha_i\,I_3 \;+\; \beta_i\,[\underline{\mathbf{z}}_e]_\times\,)\;\right] \in \Re^{3\times 6} \tag{A.109}$$

Constraints on the Active Joints

Constraints on the active joint coordinates are computed as follows:

$$\dot{\underline{\mathbf{x}}}_{pi} \;=\; \dot{q}_i\, \underline{\mathbf{z}}_{pi} \,\times\, \underline{\mathbf{x}}_{pi} \;=\; \dot{q}_i\, \underline{\mathbf{y}}_{pi} \tag{A.110}$$

$$\underline{\mathbf{y}}_{pi}^{T}\; \dot{\underline{\mathbf{x}}}_{pi} \;=\; \dot{q}_i \tag{A.111}$$

$$\dot{q}_i \;=\; M_{q_i}\, \dot{\mathbb{X}}\,, \qquad M_{q_i} \;=\; \left[\, \underline{\mathbf{y}}_{pi}^{T}\, M_{pi} \right] \in \Re^{1\times 6} \tag{A.112}$$

A.3.5 Kinematic Coordinates

Equation (A.113) and Table (A.7) show the motion basis and the kinematic coordinates of the mechanism, respectively.

$$\mathbf{u}_{x_{pi}} \;=\; \dot{\underline{\mathbf{x}}}_{pi}\,, \qquad \mathbf{u}_{x_{ai}} \;=\; \dot{\underline{\mathbf{x}}}_{ai}\,, \qquad \mathbf{u}_{x_{bi}} \;=\; \dot{\underline{\mathbf{x}}}_{bi}\,, \qquad i \;=\; 1,2,3 \tag{A.113}$$

Table A.7 – The (transposed) kinematic coordinates of the 3-RRR parallel robot, i=1,2,3.

	$\partial\, \dot{\mathbf{S}}_{pi}$	$\partial\, \boldsymbol{\omega}_{pi}$	$\partial\, \dot{\mathbf{S}}_{ai}$	$\partial\, \boldsymbol{\omega}_{ai}$	$\partial\, \dot{\mathbf{S}}_{bi}$	$\partial\, \boldsymbol{\omega}_{bi}$
$\partial\, \dot{\underline{\mathbf{x}}}_{pi}$	$\frac{1}{2}\,\ell_{pi}\, I_3$	$[\,\underline{\mathbf{x}}_{pi}\,]_{\times}^{T}$	$\ell_{pi}\, I_3$	$\mathbf{0}_{3\times 3}$	$\ell_{pi}\, I_3$	$\mathbf{0}_{3\times 3}$
$\partial\, \dot{\underline{\mathbf{x}}}_{ai}$	$\mathbf{0}_{3\times 3}$	$\mathbf{0}_{3\times 3}$	$\frac{1}{2}\,\ell_{ai}\, I_3$	$[\,\underline{\mathbf{x}}_{ai}\,]_{\times}^{T}$	$\ell_{ai}\, I_3$	$\mathbf{0}_{3\times 3}$
$\partial\, \dot{\underline{\mathbf{x}}}_{bi}$	$\mathbf{0}_{3\times 3}$	$\mathbf{0}_{3\times 3}$	$\mathbf{0}_{3\times 3}$	$\mathbf{0}_{3\times 3}$	$\ell_{bi}\, I_3$	$[\,\underline{\mathbf{x}}_{bi}\,]_{\times}^{T}$

A.3.6 Dynamic Coordinates

Local Active/Reactive Forces and Torques

Table A.8 tabulates the local forces and torques of the 3-RRR parallel robot.

Table A.8 – The local forces and torques of the 3-RRR parallel robot, i=1,2,3.

	Active		*Friction*		*Inertia**	
	Actuator	*Gravity*	*Actuator*	*Passive Joint*	*Actuator*	*Element*
$Forces\,(pi)$	$\mathbf{0}$	$\mathbf{f}_{\mathbf{g}(pi)}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{f}^{*}_{pi}$
$Torques\,(pi)$	$\tau_{\underline{\mathbf{x}}_{pi}}\, \underline{\mathbf{z}}_{pi}$	$\mathbf{0}$	$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{pi}}$	$\mathbf{0}$	$\boldsymbol{\tau}^{*}_{\underline{\mathbf{x}}_{pi}}$	$\boldsymbol{\tau}^{*}_{pi}$
$Forces\,(ai)$	$\mathbf{0}$	$\mathbf{f}_{\mathbf{g}(ai)}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{f}^{*}_{ai}$
$Torques\,(ai)$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{ai}}$	$\mathbf{0}$	$\boldsymbol{\tau}^{*}_{ai}$
$Forces\,(bi)$	$\mathbf{0}$	$\mathbf{f}_{\mathbf{g}(bi)}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{f}^{*}_{bi}$
$Torques\,(bi)$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{bi}}$	$\mathbf{0}$	$\boldsymbol{\tau}^{*}_{bi}$

Computing Dynamic Coordinates

So as to eliminate the non-contributing forces, the dynamic coordinates are computed through the matrix-wise multiplication of the Tables A.7 (transposed kinematic coordinates) and A.8 (sum of the local forces and torques).

$$\begin{bmatrix} \mathbb{F}_{\underline{x}_{pi}} \\ \mathbb{F}_{\underline{x}_{ai}} \\ \mathbb{F}_{\underline{x}_{bi}} \end{bmatrix} = \begin{bmatrix} Kinematic \\ Coordinates \\ Table\ A.7 \end{bmatrix}_{(3\times 6)} \begin{bmatrix} Sum\ of \\ Forces \\ Torques \\ Table\ A.8 \end{bmatrix}_{(6\times 1)} \tag{A.114}$$

which can be explicitly written as follows:

$$\begin{bmatrix} \mathbb{F}_{\underline{x}_{pi}} \\ \mathbb{F}_{\underline{x}_{ai}} \\ \mathbb{F}_{\underline{x}_{bi}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\,\ell_{pi}\,I_3 & [\underline{\mathbf{x}}_{pi}]_{\times}^{T} & \ell_{pi}\,I_3 & \mathbf{0}_{3\times 3} & \ell_{pi}\,I_3 & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \frac{1}{2}\,\ell_{ai}\,I_3 & [\underline{\mathbf{x}}_{ai}]_{\times}^{T} & \ell_{ai}\,I_3 & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \ell_{bi}\,I_3 & [\underline{\mathbf{x}}_{bi}]_{\times}^{T} \end{bmatrix} \begin{bmatrix} \mathbf{f}_{\mathbf{g}(pi)} + \mathbf{f}_{pi}^{*} \\ \tau_{\underline{\mathbf{x}}_{pi}}\,\underline{\mathbf{z}}_{pi} + \widetilde{\boldsymbol{\tau}}_{pi} \\ \mathbf{f}_{\mathbf{g}(ai)} + \mathbf{f}_{ai}^{*} \\ \bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{ai}} + \boldsymbol{\tau}_{ai}^{*} \\ \mathbf{f}_{\mathbf{g}(bi)} + \mathbf{f}_{bi}^{*} \\ \bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{bi}} + \boldsymbol{\tau}_{bi}^{*} \end{bmatrix} \tag{A.115}$$

where

$$\widetilde{\boldsymbol{\tau}}_{pi} = \bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{pi}} + \boldsymbol{\tau}_{\underline{\mathbf{x}}_{pi}}^{*} + \boldsymbol{\tau}_{pi}^{*} \tag{A.116}$$

A.3.7 Dynamic Constraints

Exploiting (2.110), the dynamic constraints of the 3-RRR robot are written as follows:

$$\mathbf{0}_{6\times 1} = M_p^T\,\mathbb{F}_p + M_a^T\,\mathbb{F}_a + M_b^T\,\mathbb{F}_b \tag{A.117}$$

where $\mathbb{F}_p \in \Re^{9\times 1}$, $\mathbb{F}_a \in \Re^{9\times 1}$ and $\mathbb{F}_b \in \Re^{9\times 1}$ are the stacked vectors of the dynamic coordinates:

$$\mathbb{F}_p = \begin{bmatrix} \mathbb{F}_{\underline{x}_{p1}} \\ \vdots \\ \mathbb{F}_{\underline{x}_{p3}} \end{bmatrix}, \quad \mathbb{F}_a = \begin{bmatrix} \mathbb{F}_{\underline{x}_{a1}} \\ \vdots \\ \mathbb{F}_{\underline{x}_{a3}} \end{bmatrix}, \quad \mathbb{F}_b = \begin{bmatrix} \mathbb{F}_{\underline{x}_{b1}} \\ \vdots \\ \mathbb{F}_{\underline{x}_{b3}} \end{bmatrix} \tag{A.118}$$

and where $M_p \in \Re^{9\times 6}$, $M_a \in \Re^{9\times 6}$ and $M_b \in \Re^{9\times 6}$ are the stacked matrices of the inverse differential kinematic models $M_{pi} \in \Re^{3\times 6}$, $M_{ai} \in \Re^{3\times 6}$ and $M_{bi} \in \Re^{3\times 6}$, respectively.

A.4 The Orthoglide Robot

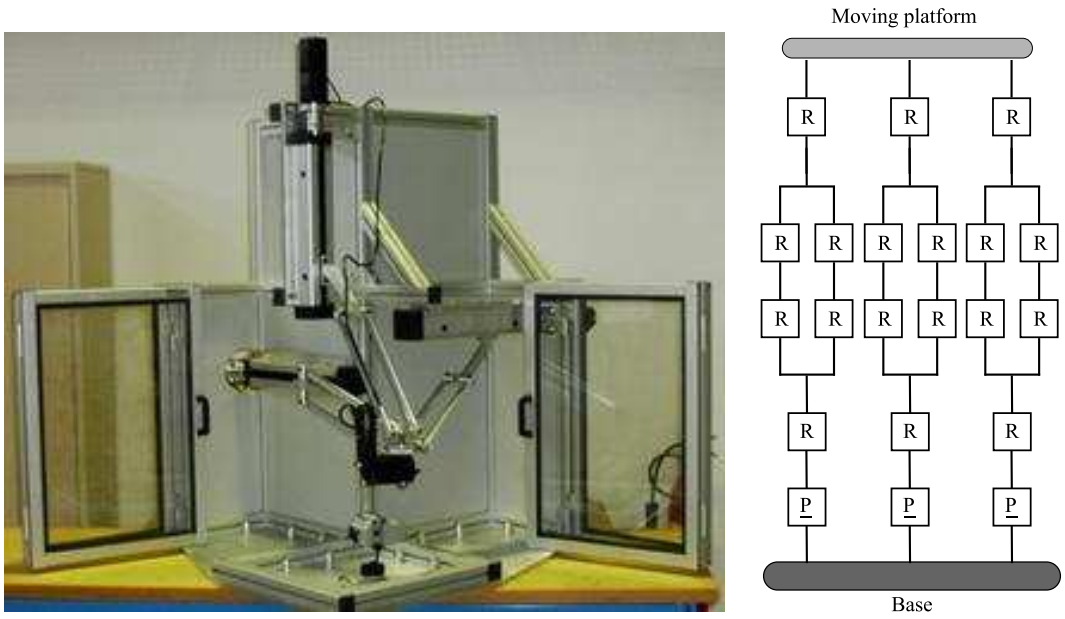


Figure A.7 – The Orthoglide parallel robot (*left*) and its graphical layout (*right*).

A.4.1 Geometry and Notation

The Orthoglide robot consists of 3 identical kinematic legs. Each kinematic leg has 3 consecutive kinematic elements $\{[\mathbf{P}_i\mathbf{A}_i], [\mathbf{A}_i\mathbf{B}_i], [\mathbf{B}_i\mathbf{C}_i]\}$. A kinematic leg of the Orthoglide is symbolically noted as $\underline{P} - R - Pa - R$ where $\underline{P}$, R and Pa denote an actuated prismatic joint, a revolute joint and a parallelogram joint, respectively. The kinematic legs interconnect the fixed base to the moving-platform. The moving-platform is a single rigid body $\{\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3\}$. It is attached to the parallelograms which restrict its motion only to translational movements in x, y and z axes. As a result, the moving-platform of the Orthoglide robot has 3 degrees of freedom. Figure A.7 shows the Orthoglide robot and its graphical layout. The kinematic legs are actuated from base by prismatic joints. The motion directions of these prismatic joints are oriented orthogonally to each other. Figure A.8 shows the geometric notation of the Orthoglide robot. In modeling, the following notation is used:

- $i = 1, 2, 3$ denotes the kinematic legs.
- $j = \{p, a, b, c\}$ is the literal representation of the kinematic elements.
- $\xi_{geo} = \{\mathbf{O}, \mathbf{P}_i, \ell_{ai}, \ell_{bi}, \ell_{ci}\}$ are the geometric parameters (constant lengths and points).
- $\xi_{dyn} = \{m_{ji}, \mathcal{I}_{ji}, f_{v_i}, f_{c_i}\}$ are the dynamic parameters (weights, inertias and frictions).
- $\mathcal{F}_o = (\mathbf{O}, \underline{\mathbf{x}}_o, \underline{\mathbf{y}}_o, \underline{\mathbf{z}}_o)$, $\mathcal{F}_e = (\mathbf{E}, \underline{\mathbf{x}}_e, \underline{\mathbf{y}}_e, \underline{\mathbf{z}}_e)$, $\mathcal{F}_{pi} = (\mathbf{P}_i, \underline{\mathbf{x}}_{pi}, \underline{\mathbf{y}}_{pi}, \underline{\mathbf{z}}_{pi})$, $\mathcal{F}_{ai} = (\mathbf{A}_i, \underline{\mathbf{x}}_{ai}, \underline{\mathbf{y}}_{ai}, \underline{\mathbf{z}}_{ai})$, $\mathcal{F}_{bi} = (\mathbf{B}_i, \underline{\mathbf{x}}_{bi}, \underline{\mathbf{y}}_{bi}, \underline{\mathbf{z}}_{bi})$ and $\mathcal{F}_{ci} = (\mathbf{C}_i, \underline{\mathbf{x}}_{ci}, \underline{\mathbf{y}}_{ci}, \underline{\mathbf{z}}_{ci})$ denote respectively the base frame, the end-effector frame, the 1^{st} kinematic element frame of the i^{th} kinematic leg, the 2^{nd} kinematic element frame of the i^{th} kinematic leg, the 3^{nd}

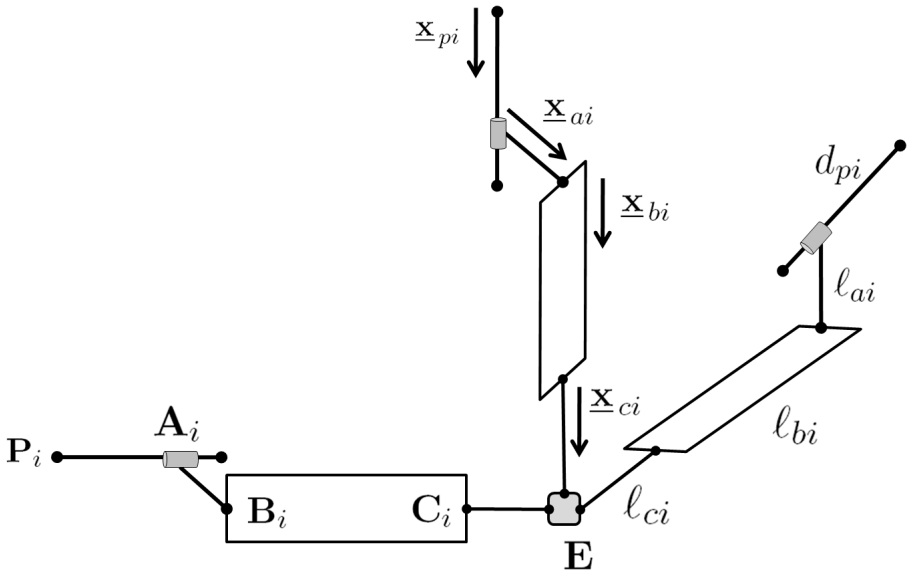


Figure A.8 – The notation of the Orthoglide parallel robot.

kinematic element frame of the i^{th} kinematic leg, and the moving-platform's i^{th} virtual kinematic element frame.

- d_{pi} is the prismatic joint coordinate of the i^{th} kinematic leg.
- The end-effector pose is the origin $\mathbf{E}$ of the end-effector frame:

$$\mathbb{X} \triangleq \begin{bmatrix} \mathbf{E} \end{bmatrix}, \qquad \dot{\mathbb{X}} \triangleq \begin{bmatrix} \dot{\mathbf{E}} \end{bmatrix} \in \Re^{3\times 1} \tag{A.119}$$

A.4.2 State Variables

Kinematic Element Types

The identical kinematic legs and the moving-platform of the Orthoglide robot have the following kinematic elements:

Kinematic Elements $[\mathbf{P}_i\mathbf{A}_i]$ Each of these kinematic elements has only 1 dof translational (along its direction) intrinsic mobility. It is thus a *telescopic type* kinematic element: $\{\underline{\mathbf{x}}_{pi},\, d_{pi}\}$. The $\underline{\mathbf{x}}_{pi}$ is a moving vector with a fixed direction.

Kinematic Elements $[\mathbf{A}_i\mathbf{B}_i]$ Each of these kinematic elements only translates in space due to prismatic joint of the previous kinematic element. It is a *bar type* kinematic element: $\underline{\mathbf{x}}_{ai}$ (moving vector with a fixed direction).

Kinematic Elements $[\mathbf{B}_i\mathbf{C}_i]$ Each of these kinematic elements (a parallelogram) has only 2 dof rotational extrinsic mobility due to existing (R)evolute joints. It is a *bar type* kinematic element: $\underline{\mathbf{x}}_{bi}$.

Moving-Platform $[\mathbf{C}_i\mathbf{E}]$ Each of these virtual kinematic elements of the moving-platform has only 3 dof translational extrinsic mobility due to (Pa)rallelogram pseudo-joint. It is thus a *bar type* kinematic element: $\underline{\mathbf{x}}_{ci}$ (moving vector with a fixed direction).

The $\{\, d_{pi},\, \underline{\mathbf{x}}_{bi} \,\}$ is the new variable set that redefine the state of the mechanism.

A.4.3 Kinematics

Mass Centers

Assuming that all the kinematic elements are homogenous and symmetric, the mass center positions are written as follows:

$$\mathbf{S}_{ai} \;=\; \mathbf{P}_i \;+\; d_{pi}\,\underline{\mathbf{x}}_{pi} \;+\; \frac{1}{2}\,\ell_{ai}\,\underline{\mathbf{x}}_{ai} \tag{A.120}$$

$$\mathbf{S}_{bi} \;=\; \mathbf{P}_i \;+\; d_{pi}\,\underline{\mathbf{x}}_{pi} \;+\; \ell_{ai}\,\underline{\mathbf{x}}_{ai} \;+\; \frac{1}{2}\,\ell_{bi}\,\underline{\mathbf{x}}_{bi} \tag{A.121}$$

$$\mathbf{S}_{ci} \;=\; \mathbf{P}_i \;+\; d_{pi}\,\underline{\mathbf{x}}_{pi} \;+\; \ell_{ai}\,\underline{\mathbf{x}}_{ai} \;+\; \ell_{bi}\,\underline{\mathbf{x}}_{bi} \;+\; \ell_{ci}\,\underline{\mathbf{x}}_{ci} \tag{A.122}$$

where $(\underline{\mathbf{x}}_{pi})$, $(\ell_{ai}\,\underline{\mathbf{x}}_{ai})$ and $(\ell_{ci}\,\underline{\mathbf{x}}_{ci})$ are constants.

Velocities

The translational and rotational velocities of the kinematic elements are computed as below:

$$\dot{\mathbf{S}}_{ai} \;=\; \dot{d}_{pi}\,\underline{\mathbf{x}}_{pi}\,, \qquad\qquad \boldsymbol{\omega}_{ai} = \mathbf{0} \tag{A.123}$$

$$\dot{\mathbf{S}}_{bi} \;=\; \dot{d}_{pi}\,\underline{\mathbf{x}}_{pi} \;+\; \frac{1}{2}\,\ell_{bi}\,\dot{\underline{\mathbf{x}}}_{bi}\,, \qquad\qquad \boldsymbol{\omega}_{bi} \triangleq \underline{\mathbf{x}}_{bi} \;\times\; \dot{\underline{\mathbf{x}}}_{bi} \tag{A.124}$$

$$\dot{\mathbf{S}}_{ci} \;=\; \dot{d}_{pi}\,\underline{\mathbf{x}}_{pi} \;+\; \ell_{bi}\,\dot{\underline{\mathbf{x}}}_{bi}\,, \qquad\qquad \boldsymbol{\omega}_{ci} = \mathbf{0} \tag{A.125}$$

Accelerations

The translational and rotational accelerations of the kinematic elements are computed as follows:

$$\ddot{\mathbf{S}}_{ai} \;=\; \ddot{d}_{pi}\,\underline{\mathbf{x}}_{pi}\,, \qquad\qquad \dot{\boldsymbol{\omega}}_{ai} = \mathbf{0} \tag{A.126}$$

$$\ddot{\mathbf{S}}_{bi} \;=\; \ddot{d}_{pi}\,\underline{\mathbf{x}}_{pi} \;+\; \frac{1}{2}\,\ell_{bi}\,\ddot{\underline{\mathbf{x}}}_{bi}\,, \qquad\qquad \dot{\boldsymbol{\omega}}_{bi} \triangleq \underline{\mathbf{x}}_{bi} \;\times\; \ddot{\underline{\mathbf{x}}}_{bi} \tag{A.127}$$

$$\ddot{\mathbf{S}}_{ci} \;=\; \ddot{d}_{pi}\,\underline{\mathbf{x}}_{pi} \;+\; \ell_{bi}\,\ddot{\underline{\mathbf{x}}}_{bi}\,, \qquad\qquad \dot{\boldsymbol{\omega}}_{ci} = \mathbf{0} \tag{A.128}$$

A.4.4 Kinematic Constraints

Assuming that the end-effector frame is located at the mass center position of the moving platform, the closed-loop constraint equation for each of the kinematic legs can be written as follow:

$$\overrightarrow{\mathbf{OE}} \;-\; \ell_{ci}\,\underline{\mathbf{x}}_{ci} \;-\; \ell_{bi}\,\underline{\mathbf{x}}_{bi} \;-\; \ell_{ai}\,\underline{\mathbf{x}}_{ai} \;-\; d_{pi}\,\underline{\mathbf{x}}_{pi} \;-\; \overrightarrow{\mathbf{OP}}_i \;=\; \mathbf{0}\,, \qquad i = 1,2,3 \tag{A.129}$$

where $\mathbf{O}$, $\mathbf{P}_i$, $(\underline{\mathbf{x}}_{pi})$, $(\ell_{ai}\,\underline{\mathbf{x}}_{ai})$ and $(\ell_{ci}\,\underline{\mathbf{x}}_{ci})$ are constants. Afterwards, one can differentiate the last closed-loop constraint equation with respect to time in order to obtain the motion constraint equation, which yields:

$$\dot{\mathbf{E}} \;-\; \ell_{bi}\,\dot{\underline{\mathbf{x}}}_{bi} \;-\; \dot{d}_{pi}\,\underline{\mathbf{x}}_{pi} \;=\; \mathbf{0} \tag{A.130}$$

Constraints on the Active $[\mathbf{P}_i\mathbf{A}_i]$ Telescopic Type Kinematic Elements

Constraints on the variations of the lengths of the active telescopic type kinematic elements are computed as follows:

$$\underline{\mathbf{x}}_{bi}^T\,\dot{\mathbb{X}} \,-\, \dot{d}_{pi}\,\underline{\mathbf{x}}_{bi}^T\,\underline{\mathbf{x}}_{pi} \,=\, \mathbf{0} \tag{A.131}$$

$$\dot{d}_{pi} \,=\, M_{d_{pi}}\,\dot{\mathbb{X}}\,, \qquad M_{d_{pi}} \,=\, \left[\begin{array}{c} \underline{\mathbf{x}}_{bi}^T \\ \hline \underline{\mathbf{x}}_{bi}^T\,\underline{\mathbf{x}}_{pi} \end{array}\right] \in \Re^{1\times 3} \tag{A.132}$$

Constraints on the Passive $[\mathbf{B}_i\mathbf{C}_i]$ Bar Type Kinematic Elements

Constraints on the orientations of the passive bar type kinematic elements are computed as follows:

$$\dot{\mathbb{X}} \,-\, \ell_{bi}\,\dot{\underline{\mathbf{x}}}_{bi} \,-\, \underline{\mathbf{x}}_{pi}\,M_{di}\,\dot{\mathbb{X}} \,=\, \mathbf{0} \tag{A.133}$$

$$\dot{\underline{\mathbf{x}}}_{bi} \,=\, M_{bi}\,\dot{\mathbb{X}}\,, \qquad M_{bi} \,=\, \left[\frac{1}{\ell_{bi}}\,\left(\,I_3 \,-\, \underline{\mathbf{x}}_{pi}\,M_{di}\,\right)\right] \in \Re^{3\times 3} \tag{A.134}$$

A.4.5 Kinematic Coordinates

Equation (A.135) and Table A.9 show the motion basis and the kinematic coordinates of the mechanism, respectively.

$$u_{d_{pi}} \,=\, \dot{d}_{pi}\,, \qquad \mathbf{u}_{x_{bi}} \,=\, \dot{\underline{\mathbf{x}}}_{bi}\,, \qquad i \,=\, 1,2,3 \tag{A.135}$$

Table A.9 – The (transposed) kinematic coordinates of the Orthoglide parallel robot, i=1,2,3.

	$\partial\,\dot{\mathbf{S}}_{ai}$	$\partial\,\boldsymbol{\omega}_{ai}$	$\partial\,\dot{\mathbf{S}}_{bi}$	$\partial\,\boldsymbol{\omega}_{bi}$	$\partial\,\dot{\mathbf{S}}_{ci}$	$\partial\,\boldsymbol{\omega}_{ci}$
$\partial\,\dot{d}_{pi}$	$\underline{\mathbf{x}}_{pi}^T$	$\mathbf{0}_{3\times1}$	$\underline{\mathbf{x}}_{pi}^T$	$\mathbf{0}_{3\times1}$	$\underline{\mathbf{x}}_{pi}^T$	$\mathbf{0}_{3\times1}$
$\partial\,\dot{\underline{\mathbf{x}}}_{bi}$	$\mathbf{0}_{3\times3}$	$\mathbf{0}_{3\times3}$	$\frac{1}{2}\,\ell_{bi}\,I_3$	$[\,\underline{\mathbf{x}}_{bi}\,]_\times^T$	$\ell_{bi}\,I_3$	$\mathbf{0}_{3\times3}$

A.4.6 Dynamic Coordinates

Listing Active and Reactive Forces

The active forces are as follows:

– *Actuator and Gravity Forces:*

$$\mathbf{f}_{d_{pi}} \,=\, f_{d_{pi}}\,\underline{\mathbf{x}}_{pi}\,, \qquad \mathbf{f}_{\mathbf{g}(ai)} \,=\, m_{ai}\,\mathbf{g}\,, \qquad \mathbf{f}_{\mathbf{g}(bi)} \,=\, m_{bi}\,\mathbf{g}\,, \qquad \mathbf{f}_{\mathbf{g}(ci)} \,=\, m_{ci}\,\mathbf{g} \tag{A.136}$$

where $\mathbf{f}_{d_{pi}}$ and $\mathbf{f}_{\mathbf{g}(ji)}$ are the actuator and gravity forces, respectively.

The reactive forces are as follows:

– *Actuator Inertial Forces:* The inertial forces of the linear actuators are as below:

$$\mathbf{f}^{*}_{d_{pi}} \;=\; -\,m_{ai}\,\ddot{d}_{pi}\,\underline{\mathbf{x}}_{pi} \tag{A.137}$$

where m_{ai} is the mass moved by the linear actuator. These inertial forces of the actuators correspond to the inertias of the ai kinematic elements, since these kinematic elements are rigidly attached to the linear actuators.

– *Kinematic Element Body Inertial Forces and Torques:* The inertial force $\mathbf{f}^{*}_{ai}$ of the ai^{th} kinematic element is compensated by the actuator inertia, and inertial torque $\boldsymbol{\tau}^{*}_{ai} = \mathbf{0}$ since the ai^{th} kinematic element does not have a rotational mobility. Then, the inertial forces and torques of the rest of the kinematic elements are written as follows:

$$\mathbf{f}^{*}_{bi} \;=\; -\,m_{bi}\,\ddot{\mathbf{S}}_{bi}\,, \qquad \boldsymbol{\tau}^{*}_{bi} \;=\; -\,\mathcal{I}^{T}_{bi}\,\dot{\boldsymbol{\omega}}_{bi} \;-\; \boldsymbol{\omega}_{bi} \;\times\; (\mathcal{I}^{T}_{bi}\,\boldsymbol{\omega}_{bi}) \tag{A.138}$$

$$\mathbf{f}^{*}_{ci} \;=\; -\,m_{bi}\,\ddot{\mathbf{S}}_{bi}\,, \qquad \boldsymbol{\tau}^{*}_{ci} \;=\; \mathbf{0} \tag{A.139}$$

– *Active Joint Frictional Forces:* The frictional forces of the linear actuators are as follows:

$$\bar{\mathbf{f}}_{d_{pi}} \;=\; -\,(\,\bar{f}_{v(d_{pi})}\,\dot{d}_{pi} \;+\; \bar{f}_{c(d_{pi})}\,sign(\,\dot{d}_{pi}\,)\,)\,\underline{\mathbf{x}}_{pi} \tag{A.140}$$

where $\bar{f}_{v(d_{pi})}$ and $\bar{f}_{c(d_{pi})}$ are the viscous and Coulomb friction coefficients of the linear actuators.

– *Passive Joint Frictional Torques:* The frictional torques of the passive joints are as follows:

$$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{bi}} \;=\; -\,\bar{\tau}_{v(\underline{\mathbf{x}}_{bi})}\,\boldsymbol{\omega}_{bi} \;-\; \bar{\tau}_{c(\underline{\mathbf{x}}_{bi})}\,sign(\,\boldsymbol{\omega}^{T}_{bi}\,\underline{\mathbf{z}}_{bi}\,)\,\underline{\mathbf{z}}_{bi} \tag{A.141}$$

$$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{ci}} \;=\; -\,\bar{\tau}_{v(\underline{\mathbf{x}}_{ci})}\,(\,\boldsymbol{\omega}_{ci} \;-\; \boldsymbol{\omega}_{bi}\,) \;-\; \bar{\tau}_{c(\underline{\mathbf{x}}_{ci})}\,sign(\,(\,\boldsymbol{\omega}_{ci} \;-\; \boldsymbol{\omega}_{bi}\,)^{T}\,\underline{\mathbf{z}}_{ci}\,)\,\underline{\mathbf{z}}_{ci} \tag{A.142}$$

where $\bar{\tau}_{v(\underline{\mathbf{x}}_{ji})}$ and $\bar{\tau}_{c(\underline{\mathbf{x}}_{ji})}$ are the viscous and Coulomb friction coefficients of the passive rotary joints.

Table A.10 tabulates all of the local forces and torques of the Orthoglide parallel robot.

Table A.10 – The local forces and torques of the Orthoglide parallel robot, i=1,2,3.

	Active		*Friction*		*Inertia**	
	Actuator	*Gravity*	*Actuator*	*Passive Joint*	*Actuator*	*Element*
$Forces(pi, ai)$	$f_{d_{pi}}\,\underline{\mathbf{x}}_{pi}$	$\mathbf{f}_{\mathbf{g}(ai)}$	$\bar{\mathbf{f}}_{d_{pi}}$	$\mathbf{0}$	$\mathbf{f}^{*}_{d_{pi}}$	$\mathbf{0}$
$Torques(pi, ai)$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
$Forces(bi)$	$\mathbf{0}$	$\mathbf{f}_{\mathbf{g}(bi)}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{f}^{*}_{bi}$
$Torques(bi)$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{bi}}$	$\mathbf{0}$	$\boldsymbol{\tau}^{*}_{bi}$
$Forces(ci)$	$\mathbf{0}$	$\mathbf{f}_{\mathbf{g}(ci)}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{f}^{*}_{ci}$
$Torques(ci)$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{ci}}$	$\mathbf{0}$	$\mathbf{0}$

Computing Dynamic Coordinates

So as to eliminate the non-contributing forces, the dynamic coordinates are computed through the matrix-wise multiplication of the Tables A.9 (transposed kinematic coordinates) and A.10 (sum of the local forces and torques).

$$\begin{bmatrix} \mathbb{F}_{d_{pi}} \\ \mathbb{F}_{\underline{x}_{bi}} \end{bmatrix} = \begin{bmatrix} Kinematic \\ Coordinates \\ Table\ A.9 \end{bmatrix}_{(2\times 6)} \begin{bmatrix} Sum\ of \\ Forces \\ Torques \\ Table\ A.10 \end{bmatrix}_{(6\times 1)} \tag{A.143}$$

which can be explicitly written as follows:

$$\begin{bmatrix} \mathbb{F}_{d_{pi}} \\ \mathbb{F}_{\underline{x}_{bi}} \end{bmatrix} = \begin{bmatrix} \underline{\mathbf{x}}_{pi}^T & \mathbf{0}_{3\times 1} & \underline{\mathbf{x}}_{pi}^T & \mathbf{0}_{3\times 1} & \underline{\mathbf{x}}_{pi}^T & \mathbf{0}_{3\times 1} \\ \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \frac{1}{2}\,\ell_{bi}\,I_3 & [\underline{\mathbf{x}}_{bi}]_{\times}^T & \ell_{bi}\,I_3 & \mathbf{0}_{3\times 3} \end{bmatrix} \begin{bmatrix} f_{d_{pi}}\,\underline{\mathbf{x}}_{pi} + \widetilde{\mathbf{f}}_{pi} \\ \mathbf{0} \\ \mathbf{f}_{\mathbf{g}(bi)} + \mathbf{f}_{bi}^* \\ \bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{bi}} + \boldsymbol{\tau}_{bi}^* \\ \mathbf{f}_{\mathbf{g}(ci)} + \mathbf{f}_{ci}^* \\ \bar{\boldsymbol{\tau}}_{\underline{\mathbf{x}}_{ci}} \end{bmatrix} \tag{A.144}$$

where

$$\widetilde{\mathbf{f}}_{pi} = \mathbf{f}_{\mathbf{g}(ai)} + \bar{\mathbf{f}}_{d_{pi}} + \mathbf{f}_{d_{pi}}^* \tag{A.145}$$

A.4.7 Dynamic Constraints

Exploiting (2.110), the dynamic constraints of the Orthoglide robot are written as follows:

$$\mathbf{0}_{3\times 1} = M_{d_p}^T\,\mathbb{F}_{d_p} + M_b^T\,\mathbb{F}_b \tag{A.146}$$

where $\mathbb{F}_{d_p} \in \Re^{3\times 1}$ and $\mathbb{F}_b \in \Re^{9\times 1}$ are the stacked vectors of the dynamic coordinates:

$$\mathbb{F}_{d_p} = \begin{bmatrix} \mathbb{F}_{d_{p1}} \\ \vdots \\ \mathbb{F}_{d_{p3}} \end{bmatrix}, \quad \mathbb{F}_b = \begin{bmatrix} \mathbb{F}_{\underline{x}_{b1}} \\ \vdots \\ \mathbb{F}_{\underline{x}_{b3}} \end{bmatrix} \tag{A.147}$$

and where $M_{d_p} \in \Re^{3\times 3}$ and $M_b \in \Re^{9\times 3}$ are the stacked matrices of the inverse differential kinematic models $M_{d_{pi}} \in \Re^{1\times 3}$ and $M_{bi} \in \Re^{3\times 3}$, respectively.

References

[AAALM06] O. Ait-Aider, N. Andreff, J.M. Lavest, and P. Martinet. Simultaneous object pose and velocity computation using a single view from a rolling shutter camera. *In Proceedings of the 9th Euro. Conf. on Computer Vision, (ECCV'06)*, pages 56–68, Graz, Austria, 2006.

[Aea04] H. Abdellatif and et al. Direct identification of dynamic parameters for parallel manipulators. *International Conference on Mechatronics and Robotics*, 2004.

[AEH02] N. Andreff, B. Espiau, and R. Horaud. Visual servoing from lines. *Int. Journal of Robotics Research*, pages 679–700, August 2002.

[AMM05] N. Andreff, A. Marchadier, and P. Martinet. Vision-based control of a Gough-Stewart parallel mechanism using legs observation. *IEEE International Conference on Robotics and Automation*, 2005.

[And99] N. Andreff. Asservissement visuel à partir de droites et auto-étalonnage pince-caméra. *Thèse, Institut National Polytechnique de Grenoble*, 1999.

[And06] N. Andreff. Des droites et des robots. modélisation, identification et commande référencées vision des machines complexes. *Habilitation à diriger des recherches, Université Blaise Pascal*, 2006.

[Ang97] J. Angeles. Fundamentals of robotic mechanical systems. *Springer-Verlag*, New York 1997.

[BA95] L. Baron and J. Angeles. The isotropic decoupling of the direct kinematics of parallel manipulators under sensor redundancy. *Intternational Conference on Robotics and Automations*, 1995.

[BAP98] L. Beji, A. Abichou, and M. Pascal. Tracking control of a parallel robot in the task space. *In International Conference on Robotics and Automation (ICRA'98)*, pages 2309–2314, Leuven, Belgium, May 1998.

[BHC00] E. Burdet, M. Honegger, and A. Codourey. Controllers with desired dynamics compensation and their implementation on a 6 dof parallel manipulator. *In International Conference on Intelligent Robots and Systems (IROS'00)*, pages 1–7, Takamatsu, Japon, October 2000.

[BK99a] S. Besnard and W. Khalil. Calibration of parallel robots using two inclinometers. *IEEE International Conferance on Robotics and Automation*, 1999.

[BK99b] S. Besnard and W. Khalil. Calibration of parallel robots using two inclinometers. *In IEEE International Conference on Robotics and Automation (ICRA'99)*, pages 1758–1763, Detroit, USA, Mai 1999.

[BK01] S. Besnard and W. Khalil. Identifiable parameters for parallel robots kinematic calibration. *IEEE International Conference on Robotics and Automation*, 2001.

[Bru99] H. Bruyninckx. Dualities between serial and parallel 321 manipulators. *In International Conference on Robotics and Automation (ICRA'99)*, pages 1532–1537, Detroit, Michigan, May 1999.

[BS98] H. Bruyninckx and J. De Schutter. Unified kinetostatics for serial, parallel and mobile robots. *Advances in Robotic Kinematics (ARK'98)*, pages 343–352, 1998.

[BTKL99] I. Bonev, J. Tyu, N-J. Kim, and S-K. Lee. A simple new closed-form solution of the direct kinematics of parallel manipulators using three linear extra sensors. *Intternational Conference on Robotics and Automations*, 1999.

[Can86] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, pages 679–698, 1986.

[CB07] P. Chalimbaud and F. Berry. Embedded active vision system based on FGPA achitecture. *In EURASIP Journal on Embedded Systems*, 2007.

[CH07] F. Chaumette and S. Hutchinson. Visual servo control, part II: Advanced approaches. *IEEE Robotics and Automation Magazine*, pages 109–118, March 2007.

[Cha30] M. Chasles. Note sur les propriétés générales du système de deux corps semblables entr'eux et placés d'une manière quelconque dans l'espace; et sur le déplacement fini ou infiniment petit d'un corps solide libre. *Bulletin des Sciences Mathématiques*, pages 321–326, 1830.

[CL96] T.F. Coleman and Y. Li. An interioir, trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization*, pages 418–445, 1996.

[Cla88] R. Clavel. Delta: a fast robot with parallel geometry. *18 th International Symposium on Industrial Robot*, pages 90–101, 1988.

[Cla89] R. Clavel. Une nouvelle structure de manipulateurs parallèles pour la robotique legére. *APII*, pages 501–519, 1989.

[Cla91] R. Clavel. Conception d'un robot parallèle rapide à 4 degrès de liberté. *Ph.D. Thesis, EPFL, Lousanne, Switzerland*, 1991.

[COB93] K. Cheok, J. Overhalt, and R. Beck. Exact methods for determining the kinematics of a stewart platform using additional displacement sensors. *Journal of Robotics System*, pages 689–707, 1993.

[Cor95] P.I. Corke. Dynamic issues in robot visual-servo systems. *Int. Symp. on Robotics Research, (ISSR'95)*, pages 488–498, Springer, 1995.

[CP04] Y.J. Chiu and M.H. Perng. Self-calibration of a general hexapod manipulator with enhanced precision in 5-dof motions. *Mechanisms and Machine Theory*, 39, 2004.

[CW98] D. Chablat and P. Wenger. Working modes and aspects in fullyparallel manipulators. *IEEE International Conference on Robotics and Automation*, 1998.

[CYH06] D. Cong, D. Yu, and J. Han. Kinematic calibration of parallel robots using CMM. *In Proc. 6th World Congress on Intelligent Control and Automation*, 2006.

[DACP06] D. Daney, N. Andreff, G. Chabert, and Y. Papegay. Interval method for calibration of parallel robots: vision-based experiments. *Mechanism and Machine Theory*, pages 929–944, 2006.

[Dah10] R. Dahmouche. Contribution à l'estimation de mouvement 3D et à la commande par vision rapide: Application aux robots parallèles. *Thèse de Doctorat, Université Blaise Pascal*, France, 2010.

[Dal07] T. Dallej. Contribution à un modèle générique pour l'asservissement visuel des robots parallèles par observation des éléments cinématiques. *Thèse de Doctorat, Université Blaise Pascal*, France, 2007.

[DAM07] T. Dallej, N. Andreff, and P. Martinet. Image-based visual servoing of the I4R parallel robot without proprioceptive sensors. *IEEE Int. Conf. on Robotics and Automation, (ICRA'07), Roma, Italy*, 2007.

[DAM11] T. Dallej, N. Andreff, and P. Martinet. Contribution to a generic model for visual servoing of parallel robots using legs observation. *Submitted to International Journal of Robotics Research, (IJRR'11)*, 2011.

[DAMM06] T. Dallej, N. Andreff, Y. Mezouar, and P. Martinet. 3D pose visual servoing relieves parallel robot control from joint sensing. *In International Conference on Intelligent Robots and Systems (IROS'06)*, pages 4291–4296, Beijing, China, October 2006.

[DAMM09] R. Dahmouche, N. Andreff, Y. Mezouar, and P. Martinet. 3D pose and velocity visual tracking based on sequential region of interest acquisition. *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, (IROS'09)*, St. Louis, USA, 2009.

[Dan99] D. Daney. Self calibration of Gough platform using leg mobility constraints. *In World congress on the theory of machine and mechanisms (IFToMM)*, pages 104–109, Oulu, Finlande, 1999.

[Dan03] D. Daney. Kinematic calibration of the Gough platform. *Robotica*, pages 677–690, 2003.

[DC99] B. Dasgupta and P. Choudhury. A general strategy based on the Newton-Euler approach for the dynamic formulation of parallels manipulators. *Mechanism and Machine Theory*, pages 801–824, 1999.

[DH55] J. Denavit and R.S. Hartenberg. A kinematic notation for lower pair mechanisms based on matrices. *Transactions of the ASME, Journal of Applied Mechanics*, pages 215–221, 1955.

[DH06] B. Denkena and C. Holz. Advanced position and force control concepts for the linear direct driven hexapod palida. *In Chemnitz Parallel Kinematics Seminar*, pages 359–378, Chemnitz, Germany, April 2006.

[Fau93] O. Faugeras. Three-dimensional computer vision - a geometric viewpoint. *Artificial intelligence. The MIT Press*, Cambridge, MA, ISBN 0-262-06158-9, 1993.

[FDM07] N. Farhat, M.A. Diaz, and V. Mata. Dynamic parameter identification of parallel robots considering physical feasibility and nonlinear friction models. *12th IFToMM World Congress*, 2007.

[Fea00] Roy Featherstone. On the limits to invariance in the twist/wrench and motor representations of motion and force vectors. *Proceedings of A Symposium Commemorating the Legacy, Works, and Life of Sir Robert Stawell Ball. Upon the 100th Anniversary of* A Treatise on the Theory of Screws, 2000.

[FLCS07] J. Falcou, J. Lapresté, T. Chateau, and J. Sérot. Nt2: A high-performance library for computer vision. 2007.

[GA88] C. Gosselin and J. Angeles. The optimum kinematic design of a planar three-degree-of-freedom parallel manipulator. *Journal of Mechanisms, Transmissions and Automation in Design*, pages 35–41, 1988.

[GA90] C. Gosselin and J. Angeles. Singularity analysis of closed-loop kinematic chains. *IEEE Transactions on Robotics and Automation*, pages 281–290, 1990.

[GCB96] G. Gogu, P. Coiffet, and A. Barraco. Représentation des déplacements des robots. *Hermes Sciences Publications, ISBN-13: 978-2866015725*, November 1996.

[GH94] C. Gosselin and J.F. Hamel. The agile eye: A high performance three-degree-of-freedom camera-orienting device. *In IEEE International Conference on Robotics and Automation*, pages 781–787, San Diego, May 1994.

[GK92] M. Gautier and W. Khalil. Exciting trajectories for the identification of base inertial parameters of robots. *International Journal of Robotics Reasearch*, pages 362–375, 1992.

[GLZ+02] F. Gao, W. Li, X. Zhao, Z. Jin, and H. Zhao. New kinematic structures for 2-, 3-, 4-, and 5-dof parallel manipulator designs. *Mechanism and machine theory*, pages 1395–1411, November 2002.

[GM03] J. Gangloff and M. De Mathelin. High-speed visual servoing of a 6 dof manipulator using multivariate predictive control. *Advanced Robotics. Special Issue: advanced 3D vision and its application to robotics*, pages 993–1021, 2003.

[Gog02] G. Gogu. Structural synthesis of parallel robotic manipulators with decoupled motions. *Internal Report ROBEA MAX-CNRS*, 2002.

[Gog04] G. Gogu. Fully-isotropic T3R1-type parallel manipulators. *On Advances In Robot Kinematics*, pages 265–272, Kluwer Academic Publishers, 2004.

[Gog08] G. Gogu. Structural synthesis of parallel robots - part1 : Methodology. *Springer*, 2008.

[GP01] M. Gautier and P. Poignet. Extended Kalman filtering and weighted least squares dynamic identification of robot. *Control Engineering Pratice*, pages 1361–1372, 2001.

[Gue03] S. Guegan. Contribution à la modélisation et l'identification dynamique des robots parallèles. *Thèse de Doctorat, Ecole Centrale de Nantes et Université de Nantes*, France, 2003.

[GW62] V. Gough and S. Whitehall. Universal tyre test machine. *In Proceedings 9th Int. Technical Congress F.I.S.I.T.A*, pages 117–137, London, May 1962.

[HBP+05] J. Hesselbach, C. Bier, I. Pietsch, N. Plitea, S. Büttgenbach, A. Wogersien, and J. Güttler. Passive-joint sensors for parallel robots. *Mechatronics*, pages 43–65, 2005.

[HBS00] M. Honegger, R. Brega, and G. Schweitzer. Application of a non-linear adaptative controller to a 6 dof parallel manipulator. *In IEEE International Conference on Robotics and Automation (ICRA'00)*, pages 1930–1935, San Francisco, USA, April 2000.

[HJ85] R.A. Horn and C.R. Johnson. Matrix analysis. *Cambridge University Press*, 1985.

[HL95] J.M. Hollerbach and D.M. Lokhorst. Closed-loop kinematic calibration of the Rsi 6-dof hand controller. *IEEE Transactions on Robotics and Automation*, pages 352–359, 1995.

[Hus94] M.L. Husty. An algorithm for solving the direct kinematics of the Gough-Stewart platforms. *Technical Report TR-CIMS-94-7, McGill University*, Montréal, Canada, 1994.

[Ion03] T.G. Ionescu. Terminology for mechanisms and machine science. *Mechanism and Machine Theory*, pages 597–901, 2003.

[KAS+98] Y. Koseki, T. Arai, K. Sugimoto, T. Takatuji, and M. Goto. Design and accuracy evaluation of high-speed and high precision parallel mechanism. *IEEE International Conferance on Robotics and Automation*, 1998.

[KD02] W. Khalil and E. Dombre. Modeling, identification and control of robots. *Hermes Penton Science*, London-Paris, 2002.

[KI04] W. Khalil and O. Ibrahim. General solution for the dynamic modeling of parallel robots. *IEEE Int. Conf. on Robotics and Automation, (ICRA'04)*, New Orleans, LA, 2004.

[KK86] W. Khalil and J.F. Kleinfinger. A new geometric notation for open and closed-loop robots. *In IEEE International Conference on Robotics and Automation*, pages 1174–1179, San Francisco, USA, April 1986.

[KL85] T.R. Kane and D.A. Levinson. Dynamics: Theory and applications. *McGraw Hill*, New York, 1985.

[Kru03] S. Krut. Contribution à l'etude des robots parallèles légers, 3t-1r et 3t-2r, à forts débattements angulaires. *Thèse de Doctorat, Université Montpellier II*, France, Novembre 2003.

[Lag87] Joseph Louis Lagrange. Mécanique analytique. 1787.

[Lam93] IMM Lammerts. Adaptive computed reference computed torque control of flexible manipulators. *Ph.D. Thesis, Technical University of Eindhoven*, 1993.

[lRd43] Jean le Rond d'Alembert. Traité de dynamique. 1743.

[LSCH03] S.H. Lee, J.B. Song, W.C. Choi, and D. Hong. Position control of a Stewart platform using inverse dynamics control with approximate dynamics. *Mechatronics*, pages 605–619, 2003.

[LWP80a] J.Y.S. Luh, M.W. Walker, and R.C.P. Paul. On-line computational scheme for mechanical manipulators. *Transaction of ASME, J. of Dynamic Systems, Measurement, and Control*, pages 69–76, 1980.

[LWP80b] J.Y.S. Luh, M.W. Walker, and R.C.P. Paul. Resolved acceleration control of mechanical manipulators. *IEEE Transactions on Automatic Control*, pages 468–474, 1980.

[MC02] E. Marchand and F. Chaumette. Virtual visual servoing: A framework for real-time augmented reality. *EUROGRAPHICS 2002 Conference Proceeding*, pages 289–298, Saarbrcken, Germany, 2002.

[MCKP02] F. Marquet, O. Company, S. Krut, and F. Pierrot. Enhancing parallel robots accuracy with redundant sensors. *Intternational Conference on Robotics and Automations*, 2002.

[MD85] M.G. Mohammed and J. Duffy. A direct analysis of instantaneous kinematics of fully parallel robot manipulators. *ASME Journal of Mechanism, Transmissions, and Automation in Design*, pages 226–229, 1985.

[Mer90] J.P. Merlet. An algorithm for the forward kinematics of general 6 dof parallel manipulator. *Research report, rr 1331*, INRIA, 1990.

[Mer00] J.P. Merlet. Parallel robots. *Kluwer Academic Publishers*, 2000.

[MH11] A. Muller and T. Hufnagel. A projection method for the elimination of contradicting control forces in redundantly actuated PKM. *IEEE International Conference on Robotics and Automation*, 2011.

[MK96] P. Mitiguy and T. Kane. Motion variables leading to efficient equations of motion. *Journal of Robotics Research*, pages 522–532, October 1996.

[MLS94] R.M. Murray, Z. Li, and S.S. Sastry. A mathematical introduction to robotic manipulation. *CRC Press*, 1994.

[MPC03] F. Marquet, F. Pierrot, and O. Company. A statistical approach for the computation of the forward kinematic model of redundantly actuated mechanisms. *IEEE International Conference on Intelligent Robots and Systems*, 2003.

[MTW03] G. Meng, L. Tiemin, and Y. Wensheng. Calibration method and experiment of stewart platform using a laser tracker. *IEEE International Conferance on Systems, Man and Cybernetics*, 2003.

[NBHW00] W.S. Newman, C.E. Birkhimer, R.J. Horning, and A.T. Wilkey. Calibration of a motoman p8 robot based on laser tracking. *In IEEE International Conference*

on Robotics and Automation (ICRA'00), pages 3597–3602, San Francisco, USA, April 2000.

[NCP12] G.S. Natal, A. Chemori, and F. Pierrot. Dual-space adaptive control of redundantly actuated parallel manipulators for extremely fast operations with load changes. *IEEE International Conference on Robotics and Automation*, 2012.

[NITM00] Y. Nakabo, M. Ishikawa, H. Toyoda, and S. Mizuno. 1ms column parallel vision system and its application of high speed target tracking. *IEEE Int. Conf. on Robotics and Automation, (ICRA'00)*, San Francisco, USA, 2000.

[NKC+08] V. Nabat, S. Krut, O. Company, P. Poignet, and F. Pierrot. On the design of a fast parallel robot based on its dynamic model. *Springer-Verlag Berlin Heidelberg, Experimental Robotics*, January 2008.

[NRC+05] V. Nabat, M.O. Rodrigues, O. Company, S. Kurt, and F. Pierrot. Par4: very high speed parallel robot for pick-and-place. *IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS'05)*, Alberta, Canada, 2005.

[OAM10] E. Ozgur, N. Andreff, and P. Martinet. Vector-based dynamic control of the Quattro parallel robot by means of leg orinetations. *IEEE Int. Conf. on Robotics and Automation, (ICRA'10)*, Alaska, USA, 2010.

[OBAM11] E. Ozgur, N. Bouton, N. Andreff, and P. Martinet. Dynamic control of the Quattro robot by the leg edges. *IEEE Int. Conf. on Robotics and Automation, (ICRA'11)*, Shanghai, China, 2011.

[OP01] M.M. Olsen and H.G. Peterson. A new method for estimating parameters of a dynamic robot model. *IEEE Transactions on Robotics and Automation*, pages 95–100, 2001.

[Pac08] F. Paccot. Contribution à la commande dynamique référencée capteur de robots parallèles. *Thèse de Doctorat, Université Blaise Pascal*, France, 2008.

[PAM09] F. Paccot, N. Andreff, and P. Martinet. A review on dynamic control of parallel kinematic machines: theory and experiments. *Int. Journal of Robotics Research*, pages 395–416, February 2009.

[Pau81] R.C.P. Paul. Robot manipulators: mathematics, programming and control. *MIT Press*, 1981.

[PCG99] V. Parenti-Castelli and R. Di Gregorio. Determination of the actual configuration of the general Stewart platform using only one additional sensor. *ASME Journal of Mechanical Design*, pages 21–25, 1999.

[Pie91] F. Pierrot. Robots pleinement parallèles légers: Conception modélisation et commande. *Thèse de Doctorat, Université Montpellier II*, France, Avril 1991.

[PK98] F.C. Park and J.W. Kim. Manipulability and singularity analysis of multiple robot systems: A geometric approach. *In International Conference on Robotics and Automation (ICRA'98)*, pages 1032–1036, Leuven, Belgium, May 1998.

[Plu65] J. Plucker. On a new geometry of space. *Philosophical Transactions of the Royal Society of London*, pages 725–791, 1865.

[PRV03] P. Poignet, N. Ramadani, and A. Vivas. Robust estimation of parallel robot dynamic parameters with interval analysis. *42nd IEEE Conference on Decision and Control*, 2003.

[RALD06] P. Renaud, N. Andreff, J.-M. Lavest, and M. Dhome. Simplifying the kinematic calibration of parallel mechanisms using vision-based metrology. *IEEE Transactions on Robotics*, pages 12–22, 2006.

[Ren03] P. Renaud. Apport de la vision pour l'identification géométrique de mécanismes parallèles. *Thèse de Doctorat, LaRAMA-LASMEA, Université Blaise Pascal*, France, 2003.

[RFS08] X. Ren, Z. Feng, and C. Su. Kinematic calibration of parallel robots using orientation constraint. *In Proceedings of the IEEE International Symposium on Industrial Electronics*, 2008.

[RGM08] L. Riberio, R. Guenther, and D. Martins. Screw-based relative Jacobian for manipulators cooperating in a task. *ABCM Symposium Series in Mechatronics*, pages 276–285, 2008.

[Ric03] I. Richardson. H.264 and mpeg-4 video compression: Video coding for next-generation multimedia. J. Wiley, 2003.

[RLN92] C. Reboulet, C. Lambert, and N. Nombrail. A parallel redundant manipulator: Speed-r-man and its control. *In 4th International Symposium on Robotics and Manufacturing, ISRAM*, pages 285–291, Santa-Fe, November 1992.

[RPR+06] A. Rauf, A. Pervez, J. Ryu, I. Complex, and P. Islamabad. Experimental results on kinematic calibration of parallel manipulators using a partial pose measurement device. *IEEE Transaction on Robotics*, pages 379–384, 2006.

[RR01a] A. Rauf and J. Ryu. Fully autonomous calibration of parallel manipulators by imposing position constraint. *IEEE International Conference on Robotics and Automation*, 2001.

[RR01b] J. Ryu and A. Rauf. A new method for fully autonomous calibration of parallel manipulators using a constraint link. *In Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 141–146, 2001.

[RVA+06] P. Renaud, A. Vivas, N. Andreff, P. Poignet, P. Martinet, F. Pierrot, and O. Company. Kinematic and dynamic identification of parallel mechanisms. *Control Engineering Practice*, pages 1099–1103, 2006.

[SGT+97] J. Swevers, C. Ganseman, B. Tuckel, J. De Schutter, and H. Van Brussel. Optimal robot excitation and identification. *IEEE Transactions on Robotics and Automation*, pages 730–740, 1997.

[SK52] J.G. Semple and G.T. Kneebone. Algebraic projective geometry. *Oxford Science Publication*, ISBN-13: 978-0198503637, 1952.

[SLE91] C. Samson, M. Leborgne, and B. Espiau. Robot control: The task function approach. *Oxford Engineering Series, 22, Oxford University Press, ISBN-13: 978-0198538059*, 1991.

[ST02] E. Staffetti and F. Thomas. Analytic formulation of the kinestatics of robot manipulators with arbitrary topology. *IEEE Int. Conf. on Robotics and Automation, (ICRA'02)*, 2002.

[Ste65] D. Stewart. A platform with six degrees of freedom. *In UK Institution of Mechanical Engineers Proceedings*, pages 371–386, London, 1965.

[TDH04] M. Terrier, A. Dugas, and J.Y. Hascoet. Qualification of parallel kinematics machines in high speed milling on free form. *International Journal of Machine tool and Manufacture*, pages 865–877, 2004.

[Tsa98] L.W. Tsai. The Jacobian analysis of a parallel maniuplator using the theory of reciprocal screws. *Thechnical Research Report*, 1998.

[Tsa99] L.W. Tsai. Robot analysis: The mechanics of serial and parallel manipulators. *John Wiley & Sons, Inc.*, 1999.

[TZR99] J. Tlusty, J. Ziegert, and S. Ridgeway. Fundamental comparison of the use of serial and parallel kinematics for machine tools. *Annals of the CIRP*, pages 351–356, 1999.

[URLP04] M. Ulrich, M. Ribi, P. Lang, and A. Pinz. A new high speed CMOS camera for real-time tracking application. *IEEE Int. Conf. on Robotics and Automation, (ICRA'04)*, New Orleans, 2004.

[Vin00] M. Vincze. Dynamics and system performance of visual servoing. *IEEE Int. Conf. on Robotics and Automation, (ICRA'00)*, San Francisco, USA, 2000.

[Viv04] A. Vivas. Contribution à l'identification et à la commande des robots parallèles. *Thèse de Doctorat, Université Montpellier II*, France, Novembre 2004.

[VPP03] A. Vivas, P. Poignet, and F. Pierrot. Predictive functional control for a parallel robot. *IEEE International Conference on Intelligent Robots and Systems (IROS'03)*, pages 2785–2790, October 2003.

[WC00] P. Wenger and D. Chablat. Kinematic analysis of a new parallel machine tool : the Orthoglide. *Advances in Robot Kinematics*, pages 305–314, 2000.

[WH91] K.J. Waldron and K.H. Hunt. Series-parallel dualities in actively coordinated mechanisms. *International Journal of Robotics Research*, pages 473–480, 1991.

[WHB96] L.W Wilson, C.W Hulls, and G.S Bell. Relative end-effector control using Cartesian position based visual servoing. *IEEE Transactions on Robotics and Automation*, pages 684–696, October, 1996.

[WLR86] D.E. Whitney, C.A. Lozinski, and J.M. Rourke. Industrial robot forward calibration method and results. *J. Dyn. Syst. Meas. Control*, 1986.

[WS06] Alon Wolf and Moshe Shoham. Screw theory for the synthesis of the geometry of a parallel robot for a given instantaneous task. *Mechanism and Machine Theory*, pages 656–670, 2006.

[YCYL02] G. Yang, I.M. Chen, S.H. Yeo, and W.K. Lim. Simultaneous base and tool calibration for self-calibrated parallel robots. *Robotica*, pages 367–374, 2002.

[YH05] D. Yu and J. Han. Kinematic calibration of parallel robots. *IEEE International Conference Mechatronics and Automation*, 2005.

[YOKN98] K. Yamane, M. Okada, N. Komine, and Y. Nakamura. Parallel dynamics computation and h1 acceleration control of parallel manipulators for acceleration display. *In International Conference on Robotics and Automation (ICRA'98)*, pages 2301–2308, Leuven, Belgium, May 1998.

[ZBG02] D. Zlatanov, I.A. Bonev, and C.M. Gosselin. Constraint singularities as $\mathcal{C}$-space singularities. *Advances in Robot Kinematics (ARK'02)*, pages 183–192, 2002.

List of Figures

1 A camera is observing the leg directions of a parallel robot. Leg orientation vectors unify modeling and control into a single linear control-oriented framework. . . . 2

1.1 A metaphor for a parallel robot concept: a hand holding a red apple with its fingertips. The palm of the hand forms the base platform. Fingers represent the kinematic chains. The red apple is either the moving platform or the moving platform with a large heavy load. . . . 5
1.2 The Stewart platform (left) and the Delta parallel robot (right). . . . 6
1.3 The Gough-Stewart platform (left) and the Delta parallel robot (right) joint-oriented graphical layouts. . . . 7
1.4 Robot model is assumed to be an integrator in a kinematic control. . . . 20
1.5 Sensor space kinematic control. . . . 22
1.6 Model-space kinematic control. . . . 23
1.7 Robot model is assumed to be a double integrator in a dynamic control. . . . 23
1.8 Joint Space CTC. . . . 25
1.9 Cartesian Space CTC. . . . 25
1.10 Identification. . . . 26
1.11 Identification. . . . 28
1.12 State of the MICMAC art with the unreached objectives (upper-right box). . . . 30
1.13 An Image . . . 31
1.14 3D Line . . . 33
1.15 Prisms . . . 34
1.16 Prism projection . . . 34
1.17 Cylindrical leg . . . 35

2.1 Kinematic Element . . . 47
2.2 Gough Moving Platform . . . 56
2.3 Gough Moving Platform . . . 56
2.4 Gough Moving Platform . . . 57
2.5 H4 Moving Platform . . . 58
2.6 fivebar . . . 59
2.7 The Quattro parallel robot with a base-mounted camera (*left*) and its joint-oriented graphical layout (*right*). . . . 71

2.8 Side and front views of a kinematic leg with its variables and parameters (*left*). The plan of the nacelle with its variables and parameters (*right*). 72
2.9 Body-oriented graphical layout of the Quattro parallel robot. 73
2.10 The mass centers $\mathbf{S}_{ji}$ (white squares) of the kinematic elements. All of the kinematic elements are assumed to be homogenous and symmetric. 74
2.11 Local forces and torques which act on one of the identical kinematic legs. Left figure shows the active forces and torques of a kinematic leg. Right figure shows the reactive inertial and frictional forces/torques which balance the active forces/torques. 79

3.1 View of the geometry of a cylindrical kinematic element from its 3D orientation direction (perpendicular to the paper plane). 86
3.2 Spatiotemporal set of the reference sub-images of the legs. 90
3.3 Sequentially grabbed sub-images of the legs during the motion of the Quattro robot. The lighter the color of the robot is, the more the motion is in the past. . 90
3.4 A posture error is defined using a single sub-image which is captured from a cylindrical leg of the real robot. The detected edge pixels of this sub-image allow us to form the reference green and red side rays which pass tangent to the surface of the cylindrical leg of the real robot. And the current feedback signals are the green and red edge vectors computed from the copy of the same cylindrical leg which belongs to the virtual robot. On the condition that the reference rays (i.e., contour points) are measured and the feedback edge vectors are constructed in the same camera frame, these reference rays and these edge vectors become perpendicular to each other only when the virtual leg is superimposed onto the real leg. That is to say, minimization of error (3.12) draws the posture of the virtual robot to the posture of the real robot. 91
3.5 Sequential k postures of the real robot (bottom left) and the virtual robot (top right) which imitates the real robot. We grab a sub-image per posture from a leg of the real robot and extract the edge contours. The green squares show in which posture, which leg, and which part of this leg is observed. The green arrow shows the motion path of the real robot, and below this green arrow we see the sub-images grabbed during this motion. When the last sub-image is grabbed, the virtual robot's motion is evolved with the latest estimated dynamic state $\{\mathbb{X}, \dot{\mathbb{X}}\}$ in order to approximate the k postures of the real robot. The motion of the virtual robot evolves on a straight line whereas the motion of the real robot can be along a curve as in the green arrow. The red straight arrow shows the motion path of the virtual robot. Then, for each virtual robot posture, we find the leg which corresponds to the leg observed on the real robot. The red lines on the virtual robot postures show the corresponding observed legs of the real robot. Below the red arrow, we see the 3D cylinders of these legs recovered from the postures of the virtual robot. Finally, the stacked complete error is formed with the extracted edge contours of these k sub-images and the visual edge vectors of these k 3D cylinders. 93

3.6 A full image of lower-legs with their sub-images from the base-mounted camera of the Quattro robot. These sub-images are consecutively grabbed at discrete time instants and given to the virtual visual servoing as a reference. In this image, the Quattro robot is static. 98
3.7 Single-iteration virtual visual servoing for fast dynamic state estimation of the Quattro robot. 100
3.8 Validation of the estimated state variables. 100
3.9 The scenario of the sub-image acquisition instants of the legs of the Quattro robot for its dynamic state estimation. Black squares represent instants of already grabbed sub-images. White squares are future instants of sub-image grabbing. t_c is the last acquisition instant of the camera. $\bar{t}_c$ is the last estimation instant. Black triangles represent instants of already made estimations for the dynamic state of the robot. White triangles are future estimation instants. T_c is a time period of successive acquisitions of the camera. 101
3.10 The evolution of the reference trajectory in time in the robot base frame. 102
3.11 Reference (blue dotted line) and estimated (red solid line) Cartesian space curves in the robot base frame (for the results of Table 3.4). 103
3.12 Superimposed reference (red dotted line) and estimated (black solid line) Cartesian velocities in the camera frame (for the results of Table 3.4). 104
3.13 Pose and pose velocity tracking errors versus time (for the results of Table 3.4). 105
3.14 Control space selection. 106
3.15 Linearized dynamics in the control space (left) and linearized dynamics in the state space (right). 108
3.16 Versatile computed-torque control scheme (V-CTC). 108
3.17 Body orientation space computed-torque control (BS-CTC). 109
3.18 The Quattro robot postures on the z-axis where lower-legs have the same 3D orientation vectors. 110
3.19 Leg-based Cartesian-space computed-torque control scheme (LCS-CTC). . . . 110
3.20 Edge-space computed-torque control scheme (ES-CTC). 112
3.21 Cartesian space reference trajectory expressed in the camera frame. 114
3.22 Versatile computed-torque control scheme integrated with the edge-based linear dynamic state observer. 118
3.23 End-effector computed-torque control scheme (EE-CTC). IDKM is the inverse differential kinematic model. IKM is the inverse kinematic model. 119
3.24 Reference (red) and performed (black) superimposed trajectories in ZY and ZX planes (top), motor torques (bottom) for the ES-CTC. 120
3.25 Reference (red) and performed (black) superimposed trajectories in ZY and ZX planes (top), motor torques (bottom) for the BS-CTC. 121
3.26 Reference (red) and performed (black) superimposed trajectories in ZY and ZX planes (top), and motor torques (bottom) for the LCS-CTC. 122

4.1 *Left picture*: The Quattro robot and its cell. The camera is mounted onto the base looking downwards to the legs. The back light platform is below the legs. *Right picture*: An image of the legs from the base-mounted camera. 126

4.2 The 16 edges (red lines) from 8 cylindrical legs of the Quattro robot for a given pose in calibration. . . . 128
4.3 Back-projection (white circles) of connection points $\mathbf{B}$ with linearly computed extrinsic pose parameters. . . . 129
4.4 Reference ROI^* upper-left corner positions on the image plane. Red, green, blue and black traces belong to the first, second, third and forth lower-legs, respectively. . . . 130
4.5 Cartesian space kinematic control with synchronous data acquisition. FKM and IDKM are the iterative forward kinematic model and the inverse differential kinematic model of the Quattro robot, respectively. . . . 131
4.6 The ground truth end-effector pose ($\mathbb{X}$), end-effector pose velocity ($\dot{\mathbb{X}}$) and end-effector pose acceleration ($\ddot{\mathbb{X}}$) generation. . . . 132
4.7 Performed square motion ($\mathbb{X}$) during kinematic control. *Left figure*: 3D Cartesian end-effector pose trajectory with starting and ending point (red circle). *Right figure*: Evolution of the Cartesian end-effector poses versus time. . . . 132
4.8 *Left figure*: Cartesian end-effector pose velocity ($\dot{\mathbb{X}}$) prints of the square motion. *Right figure*: Cartesian end-effector pose acceleration ($\ddot{\mathbb{X}}$) prints of the square motion. . . . 133
4.9 From left to right, sequentially grabbed (48×48 pixel^2) sub-images of the lower-legs 1, 2, 3 and 4, of the Quattro parallel robot. White dots are the detected edge pixels for the dynamic state estimation algorithm. In this figure, the sub-images are zoomed. . . . 133
4.10 High-speed dynamic state estimation scheme. . . . 133
4.11 Superimposed estimated ($\widehat{\mathbb{X}}$) and reference ($\mathbb{X}$) 3D trajectories expressed in the camera frame. Red solid line is the estimated trajectory and blue dashed line is the reference trajectory. . . . 134
4.12 Cartesian pose estimation errors ($\mathbf{e}_{\mathbb{X}}$) versus time for the Figure 4.11. . . . 135
4.13 Superimposed estimated ($\widehat{\dot{\mathbb{X}}}$) and reference ($\dot{\mathbb{X}}$) Cartesian pose velocities versus time for the Figure 4.11. . . . 136
4.14 Superimposed reference (ROI^*) and predicted ($\widehat{ROI}$) sub-image upper-left corner position trajectories on the image plane. Red dashed line is the reference and black solid line is the predicted. . . . 136
4.15 ROI prediction errors on each of the observed legs of the Quattro parallel robot. A prediction error for a ROI is computed as the Euclidean distance between the reference ROI position and the predicted ROI position. The maximum prediction error is calculated as 4.2 pixels through the whole tracking process. . . . 137
4.16 A cylinder whose radius (r) is relatively smaller than its observational distance (d_o) from the camera. A small (ϵ) noise on the measured thickness of the cylinder on the image creates ambiguity on the depth of the revolution axis of the cylinder with respect to the optical center $\mathbf{O}$ of the camera. The red line shows this ambiguity ($\tilde{z}$) on the depth. . . . 138

4.17 Lighting effects and visibility of a cylindrical leg. **O** is the optical center of the camera. *Left*: Front lighting from a point source which is assumed at the proximity of the **O**. In the case of a point-source front light, the visible part of the cylinder is the lighted region (bold arc). *Right*: Back lighting from a surface source. In the case of a surface-source back light, the visible part of the cylinder is the unlighted region (bold arc) and this region is now thinner than it is in front lighting. 139
4.18 Flow chart for the comparison of the outputs of the reference IDM and the proposed IDM with the measurements obtained during a motion of the Quattro robot. FKM is the forward differential kinematic model. $IDKM$ is the inverse differential kinematic model which relates the end-effector pose velocity to the velocity of leg orientation vectors. $IDKM_2$ is the second order inverse differential kinematic model which relates the end-effector pose acceleration to the acceleration of leg orientation vectors. 141
4.19 Superimposed motor torques for the square test trajectory. Red dashed line is the output torque ($\mathbf{\Gamma}_{ref}$) of the reference inverse dynamic model, and black solid line is the output ($\mathbf{\Gamma}$) of our proposed inverse dynamic model. The maximum difference between these two models is found to be less than 2% using the normalized root mean squares of the output. 142
4.20 The errors between the reference IDM output torques ($\mathbf{\Gamma}_{ref}$) and proposed IDM output torques ($\mathbf{\Gamma}$) for Figure 4.19. 142
4.21 Flow chart for the comparison of the outputs of the reference IDM and proposed IDM integrated with the high-speed state estimation algorithm. The accelerations are obtained from the measured articular positions of the Quattro robot. FKM is the forward differential kinematic model. $IDKM_2$ is the second order inverse differential kinematic model which relates the end-effector pose acceleration to the acceleration of leg orientation vectors. 143
4.22 Superimposed motor torques for the square test trajectory. Red dashed line is the output of the reference inverse dynamic model, and black solid line is the output of our proposed inverse dynamic model integrated with the high-speed dynamic state estimation algorithm. The difference between these two models is calculated as 5% using the normalized root mean squares of the output torques. 144
4.23 The errors between the output torques ($\mathbf{\Gamma}_{ref}$) of the reference IDM and the output torques ($\mathbf{\Gamma}$) of the proposed IDM integrated with the high-speed dynamic state observer. The errors of Figure 4.22. 144

5.1 New state of the MICMAC art. 148

A.1 The Gough-Stewart parallel robot (*left*) and its graphical layout (*right*). 153
A.2 The notation of the Gough-Stewart parallel robot. 154
A.3 The Delta parallel robot (*left*) and its graphical layout (*right*). 160
A.4 Side and front views of a kinematic leg with its variables and parameters (*left*). The plan of the moving-platform with its variables and parameters (*right*). . . . 161
A.5 The 3-RRR planar parallel robot (*left*) and its graphical layout (*right*). 167

A.6 The notation of the 3-RRR planar parallel robot. 168
A.7 The Orthoglide parallel robot (*left*) and its graphical layout (*right*). 173
A.8 The notation of the Orthoglide parallel robot. 174

Abstract

This thesis presents novel methods for modeling, tracking and control of parallel robots by means of lines. A parallel robot is composed of several closed-loop kinematic chains which cause a highly coupled-motion behavior. By treating the legs of a parallel robot as 3D lines and representing the geometry with a skeleton constructed from these 3D lines of the legs, the modeling, tracking and control of a parallel robot become geometrically and physically simpler and more intuitive.

The common key point for the simplicity and accuracy of all these methods is the precise observation of the 3D orientation vectors of the legs at high speed. This is because of parallel robots are designed for high speed applications. Thus, we first developed a body-based linear scheme both for kinematic and dynamic modeling of parallel robots. This body-based linear modeling scheme is so simple such that one can work out all the equations even for the most complex parallel robot by pen and paper. The simplicity and feasibility of this modeling scheme are conditioned on that the 3D leg direction vectors and their velocities are known. Therefore, secondly we proposed a high-speed vision based dynamic state observer which can provide these 3D leg direction vectors of a parallel robot and their velocities at each sampling time. We achieved this by sequentially observing small portions of the legs in order to form a spatio-temporal reference signal and then by minimizing the constraints written from the geometric shapes of the legs in a single-iteration virtual visual servoing scheme.

Afterwards, we constructed a versatile computed-torque control scheme which allows us to control the parallel robot for a given task in different control spaces. We defined this versatile control scheme so that we can analyse and then choose the best control space for better control of parallel robots for a given specific task.

These proposed novel methods are validated by the first promising simulation and experimental results. Obtained results encourage us to explore more the modeling, tracking and control of parallel robots by means of lines.

Résumé

Cette thèse présente des nouvelles approches de modélisation, de suivi visuel et de commande des robots parallèles en utilisant des droites. Un robot parallèle est composé de plusieurs chaînes cinématiques fermées. Par conséquent, un fort couplage de comportement apparait durant le mouvement du robot.

La géométrie (squelette) d'un robot parallèle peut être définie en considérant les jambes de ce robot comme des droites 3D. Nous avons montré qu'en observant ces droites 3D, la modélisation, le suivi visuel et la commande d'un robot parallèle deviennent plus simples et que sa représentation géométrique et physique est plus intuitive. Le point commun des méthodes proposées est l'observation des orientations 3D des jambes avec précision et à grandes vitesses. Cela permet de commander les robots parallèles de manière rapide avec une bonne précision.

Pour la modélisation cinématique et dynamique des robots parallèles, nous avons développé une représentation basée sur les éléments cinématiques qui constituent le robot. Cette représentation rend la modélisation simple et immédiate. Les modèles obtenus sont basés sur les mesures des orientations et des vitesses des éléments cinématiques.

Pour cela, nous avons proposé un observateur d'état dynamique à haute vitesse qui peut fournir les orientations et les vitesses des éléments cinématiques. La méthode proposée est basée sur l'observation séquentielle et par portion des contours de chaque jambe. Nous avons utilisé ces contours pour construire une consigne spatio-temporelle et des fonctions d'erreurs basées sur des contraintes géométriques. Ensuite, ces fonctions d'erreurs sont minimisées en une seule itération d'une tâche d'asservissement visuel virtuel.

Nous avons également proposé une commande dynamique pour contrôler un robot parallèle dans différents espaces de commande. Ceci nous a permis de mener des analyses pour identifier l'espace le plus adéquat pour réaliser une tâche spécifique.

Ces nouvelles approches sont validées en simulation et, partiellement, en expérimentation. Les résultats obtenus sont satisfaisants et ouvrent des perspectives dans le domaine de la modélisation, du suivi visuel et de la commande des robots parallèles basé sur l'observation des jambes.