ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

COMPLIANCE CONTROL OF COLLABORATING ROBOTS

M.Sc. THESIS

Mertcan KAYA

Department of Mechanical Engineering

System Dynamics and Control Engineering Programme

JUNE 2019

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

COMPLIANCE CONTROL OF COLLABORATING ROBOTS

M.Sc. THESIS

Mertcan KAYA
(503151611)

Department of Mechanical Engineering

System Dynamics and Control Engineering Programme

Thesis Advisor: Assoc. Prof. Dr. Zeki Yağız BAYRAKTAROĞLU

JUNE 2019

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

İŞBİRLİKÇİ ROBOTLARIN UYUM KONTROLÜ

YÜKSEK LİSANS TEZİ

Mertcan KAYA
(503151611)

Makina Mühendisliği Anabilim Dalı

Sistem Dinamiği ve Kontrol Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Zeki Yağız BAYRAKTAROĞLU

HAZİRAN 2019

Mertcan KAYA, a M.Sc. student of ITU Graduate School of Science Engineering and Technology, student ID 503151611, successfully defended the thesis entitled "COMPLIANCE CONTROL OF COLLABORATING ROBOTS", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assoc. Prof. Dr. Zeki Yağız BAYRAKTAROĞLU** Istanbul Technical University	.....
Jury Members :	**Prof. Dr. Ata MUĞAN** Istanbul Technical University	.....
	Assoc. Prof. Dr. Cüneyt YILMAZ Yıldız Technical University	.....

Date of Submission : **3 May 2019**
Date of Defense : **11 June 2019**

FOREWORD

I would like to thank my advisor Assoc. Prof. Dr. Zeki Yağız Bayraktaroğlu for his guidance throughout my study. I would also like to thank İTÜ Mechatronics Education and Research Center and its staff for allowing me to use the robot manipulators for this study.

June 2019 Mertcan KAYA

TABLE OF CONTENTS

Page

FOREWORD ..... **vii**
TABLE OF CONTENTS ..... **ix**
ABBREVIATIONS ..... **xi**
SYMBOLS ..... **xiii**
LIST OF TABLES ..... **xv**
LIST OF FIGURES ..... **xvii**
SUMMARY ..... **xix**
ÖZET ..... **xxi**
1. INTRODUCTION **1**
1.1 Purpose of Thesis 2
1.2 Literature Review 2
1.3 System Description 3
1.3.1 Hardware 3
1.3.1.1 Robot manipulators 4
1.3.1.2 End-effector 4
1.3.1.3 F/T transducer and controller 5
1.3.1.4 Robot controller 6
1.3.1.5 User input/output devices 7
1.3.2 Software 8
1.3.2.1 Controller operating system and interfaces 8
1.3.2.2 Robot algorithm development and analyzing environments 9
2. MATHEMATICAL MODELING **11**
2.1 Coordinate Frames and Representations 11
2.1.1 Joint space and task space coordinates 11
2.1.1.1 Position and orientation representations 12
2.1.1.2 Screw representations of velocities and forces 15
2.1.2 Virtual displacement and principle of virtual work 17
2.1.3 Transformation matrices 17
2.1.4 Geometric representation 19
2.2 Kinematics 21
2.2.1 Forward kinematics 21
2.2.2 Inverse kinematics 24
2.2.3 Forward differential kinematics 25
2.2.4 Inverse differential kinematics 28
2.3 Dynamics 29
2.3.1 Inverse dynamics 29
2.3.1.1 Euler-Lagrange method 31

2.3.1.2 Newton-Euler method 32
2.3.1.3 Joint friction and balancing forces 36
2.3.1.4 End-effector and F/T transducer dynamics 36
2.3.2 Forward dynamics 38
3. CONTROL SYSTEM DESIGN 41
3.1 Trajectory Generation 41
3.1.1 Joint space trajectories 42
3.1.2 Task space trajectories 42
3.2 Controller Design 44
3.2.1 Motion control 46
3.2.2 Force control 48
3.2.3 Compliance control 49
3.2.3.1 Hybrid position/force control 49
3.2.3.2 Parallel position/force control 52
4. EXPERIMENTATION 57
4.1 Experimental Setup 57
4.2 Task Description 58
4.3 Implementations 59
4.3.1 Task 1: Compliance control of a single robot 60
4.3.1.1 Case 1: Hybrid position/force control 61
4.3.1.2 Case 2: Parallel position/force control 62
4.3.1.3 Discussion of the task 1 64
4.3.2 Task 2: Compliance control of collaborating robots 65
4.3.2.1 Case 1: Hybrid position/force control 65
4.3.2.2 Case 2: Parallel position/force control 66
4.3.2.3 Discussion of the task 2 70
5. CONCLUSIONS 73
REFERENCES 75
APPENDICES 79
APPENDIX A 81
APPENDIX B 85
APPENDIX C 87
APPENDIX D 93
CURRICULUM VITAE 97

ABBREVIATIONS

3-D	**:** Three-dimensional
API	**:** Application Program Interface
CoM	**:** Center of Mass
D-H	**:** Denavit-Hartenberg
DoF	**:** Degree of Freedom
E-L	**:** Euler-Lagrange
EoM	**:** Equation of motion
F/T	**:** Force/Torque
FCL	**:** Force control loop
IDE	**:** Integrated Development Environment
I/O	**:** Input/Output
LLI	**:** Low-level Interface
N-E	**:** Newton-Euler
OS	**:** Operating System
PC	**:** Personal computer
PCLJ	**:** Position control loop in joint-space
PCLT	**:** Position control loop in task-space
PID	**:** Proportional–Integral–Derivative
TCP/IP	**:** Transmission Control Protocol/Internet Protocol
USB	**:** Universal Serial Bus

SYMBOLS

$\boldsymbol{q}$	: Generalized coordinate vector
$\mathbf{x}$	: Position and orientation in special Euclidean group
$\boldsymbol{x}$	: Parameterized position and orientation vector
$\boldsymbol{v}$	: Linear velocity vector
$\boldsymbol{\omega}$	: Angular velocity vector
$\mathbf{v}$	: Stack of linear and angular velocity vectors (twist)
$\boldsymbol{f}$	: Force vector
$\boldsymbol{n}$	: Moment vector
$\mathbf{f}$	: Stack of force and moment vectors (wrench)
$\mathbf{E}$	: Mapping matrix
W	: Work
$\mathbf{J}_A$	: Analytic (task) Jacobian
$\mathbf{J}$	: Geometric (basic) Jacobian
$\mathbf{R}$	: Rotation matrix
$\boldsymbol{p}$	: Position vector
$\mathbf{T}$	: Homogeneous transformation matrix
$\mathbf{X}$	: Spatial transformation matrix
$\mathbf{S}$	: Skew-symmetric matrix
$\mathbf{1}$	: Identity (unit) matrix
$\mathbf{0}$	: Matrix or vector of zeros
f	: Forward kinematics function

LIST OF TABLES

Page

Table 2.1 : D-H table for the RX160 series robot arms. 20
Table 2.2 : Parameters of the identified model of $d_e(f_e^z)$. 23
Table 4.1 : PID gains in hybrid position/force control scheme. 60
Table 4.2 : Controller parameters in parallel position/force control scheme. 60
Table 4.3 : Control parameters for different impedances. 69
Table 4.4 : Performances of the parallel position/force control in terms of position and force tracking errors. 69
Table A.1 : Work envelope of RX160 family. 84
Table A.2 : Amplitude, speed and resolution of RX160 family. 84
Table B.1 : Delta calibrations. 85
Table B.2 : Delta physical properties. 86

LIST OF FIGURES

Page

Figure 1.1 : Stäubli RX160L and RX160 model robot manipulators [29] [30]. .. 4
Figure 1.2 : End-effector. 5
Figure 1.3 : ATI Delta transducer [31]. 6
Figure 1.4 : ATI F/T controller [32]. 6
Figure 1.5 : CS8C robot controller [33]. 7
Figure 1.6 : Manual control pendant [33]. 8
Figure 2.1 : Rotation on XYZ Euler angles. 14
Figure 2.2 : Elements of twist and wrench shown on a rigid link. 16
Figure 2.3 : Implementation of D-H convention on bodies and joints. 20
Figure 2.4 : Placement of coordinate frames on the robot joints, base and end-effector (for $\boldsymbol{q} = [0, 0, \pi/2, 0, \pi/4, 0]^T$). 20
Figure 2.5 : Model of the F/T transducer and the end-effector length $d_e(f_e^z)$. 23
Figure 2.6 : 6-DoF robot arm with a spherical wrist. 24
Figure 2.7 : Velocity and acceleration vectors for link *i*. 33
Figure 2.8 : Force and moment vectors for link *i*. 34
Figure 2.9 : Coordinate placement of sensor and end-effector tip/CoM. 37
Figure 3.1 : Trajectory generator and controller. 42
Figure 3.2 : Inverse kinematics in block diagram. 43
Figure 3.3 : PID controller. 45
Figure 3.4 : Computed torque method. 46
Figure 3.5 : Motion control in joint space. 47
Figure 3.6 : Motion control in task space. 47
Figure 3.7 : Explicit force control scheme. 48
Figure 3.8 : Hybrid position/force control in joint space. 50
Figure 3.9 : Hybrid position/force control in task space. 50
Figure 3.10 : Hybrid position/force control with computed torque. 51
Figure 3.11 : Basic impedance controller. 53
Figure 3.12 : Implemented impedance controller scheme. 54
Figure 3.13 : Parallel position/force control scheme. 55
Figure 4.1 : Location of robots and work-piece. 58
Figure 4.2 : Experimental setup. 59
Figure 4.3 : Position tracking error of the end-point on the work-piece (O-xz) plane. 61
Figure 4.4 : Force control along the (O-y) axis to the work-piece plane. 62
Figure 4.5 : Translational motion of end-effector along the (O-y) axis. 63
Figure 4.6 : Orientation error of the end-effector in terms of the Euler angles. ... 63

Figure 4.7 : Absolute mean error with standard deviations of both cases for task 1. 64
Figure 4.8 : Position tracking error of the end-point on the work-piece (O-xz) plane. 66
Figure 4.9 : Force control along the (O-y) axis to the work-piece plane. 67
Figure 4.10 : Translational motion of end-effector along the (O-y) axis. 67
Figure 4.11 : Orientation error of the end-effector in terms of the Euler angles. ... 68
Figure 4.12 : Absolute mean error with standard deviations of both cases for task 2. 68
Figure 4.13 : Position tracking error of the end-point on the work-piece (O-xz) plane, with different impedances. 70
Figure 4.14 : Force control along the normal (O-y) axis to the work-piece plane, with different impedances. 71
Figure 4.15 : Translational motion of end-effector along the normal (O-y) axis, with different impedances. 71
Figure A.1 : Describtion of links and joints of the RX160 family robots. 81
Figure A.2 : Dimensions of RX160. 82
Figure A.3 : Dimensions of RX160L. 82
Figure A.4 : Work envelope of RX160 family robots on xz-plane. 83
Figure A.5 : Work envelope of RX160 family robots on xy-plane. 83
Figure B.1 : Placement of sensor frame on the F/T transducer. 85

COMPLIANCE CONTROL OF COLLABORATING ROBOTS

SUMMARY

Robot manipulators are known for their productivity in industrial applications. In a highly restricted environment, robots can achieve given tasks as predicted. However some more complex tasks require dynamic environment with interactions between humans or other robots. While they were preferred to be used individually in isolated spaces, technological developments and new studies on robotics made it possible to use multiple robots in collaboration.

Industrial robots generally work at high speeds and produce high forces on contact. Being in the workspaces of these robots is a dangerous position because it is possible to be impacted by the robot arm while it is in a motion. In order to use robots interacting with humans and other robots, various collaboration approaches have been developed.

Collaboration of robots requires a high environmental sensibility and compliance in motion. Implemented force sensors are important for measuring interaction forces. Controlling interaction forces and robot motions individually or jointly is a key part of collaboration. Various types of force and motion control methods can be classified under compliance control. Two of the compliance control methods presented in this study are named as *hybrid position/force control* and *parallel position/force control*.

In this thesis, different compliance control methods are implemented practically for a single robot and two robots in collaboration and the experimental results are compared and discussed. The purpose of this study is presented with a literature review in the introduction chapter. The system is also described in this chapter in terms of hardware and software. The hardware such as robot manipulators, sensors, computational and input/output devices are given in details. The software used throughout the development and running of the robot algorithm is described under corresponding sections.

The mathematical modeling of the robots and their attachments such as end-effector and F/T transducer is formed for kinematics and dynamics. At first, the coordinate frames and different types of representation methods are described as a basis of formation for modeling. Orientation representations such as Euler angles are mentioned. Screws denoting general velocities and forces as twist and wrench are shown. Kinematic model of robots describes the motion in different spaces. The term Jacobian is introduced and derived for various uses. Dynamic model relates forces and torques acted on a body with its motion. Two methods for deriving dynamic models are described in addition with friction and balancing effects.

In the next chapter, the control system is designed for motion and force control separately and together under compliance control schemes. Hybrid position/force control and paralel position/force control schemes are shown as control law equations

and block diagrams. The basis of trajectory generation, which acts as reference for control schemes, is also shown in this chapter.

Experiments on the implementation of the two compliance control schemes are conducted for single robot and collaborating robots tasks. The task is described as a sequence of a free motion of one of the robot's end-effector switching to a compliance motion after achieving a contact with a work-piece. Both tasks are identical except the first one has a static environment and the second one has a dynamic environment due to motion of the second robot. The experimental results for the two compliance control methods in both tasks are presented as numerical values and graphics in this chapter. Moreover, the parallel position/force control scheme is repeated with different control parameters in order to investigate the changed dynamic behavior of the system.

In the conclusions chapter, experimental results are discussed and differences between these results are shown. The shortcomings of this study and its implementations are identified and possible improvements are suggested for future studies.

İŞBİRLİKÇİ ROBOTLARIN UYUM KONTROLÜ

ÖZET

Robot manipülatörler, kullanıldıkları endüstriyel uygulamalarda verimliliği arttırması ile bilinirler. Genel olarak robotlar, kısıtları yüksek bir ortamda yüksek kesinlik ile tanımlanmış bir görevi öngörüldüğü biçimde yerine getirebilirler. Ancak robotlar için verilen bazı daha karmaşık görevler, iyi tanımlanmamış belirsiz ortamlarda ve buna ek olarak dinamik ortamların olduğu robotların insanlar ile ya da diğer robotlar ile etkileşiminin içerildiği durumlarda gerçekleştirilmeyi gerektirebilirler. Endüstriyel uygulamalarda son dönemlere kadar robotların tek başlarına yalıtılmış alanlarda kullanılması tercih edilmekteyken, robotik konusundaki teknolojik gelişmeler ve yeni çalışmalar, birden çok robotun işbirliği içinde kullanılmasına olanak sağlamıştır.

Endüstriyel robotlar, kullanıldıkları üretim süreçleri içerisinde genellikle yüksek hızda çalışırlar ve temas anında yüksek kuvvetler üretirler. Bu robotların çalışma alanları olası çarpışmalar nedeniyle tehlikelidirler. Robotların insanlar ile veya öteki robotlar ile etkileşime girebilmeleri için çeşitli işbirliği yaklaşımları geliştirilmiştir. Robotların işbirliği içerisinde çalışabilmesi için çevrelerinin yeteri kadar iyi bir şekilde algılanabilmesi ve hareketlerinin birbirleri ile uyumlu olması gerekmektedir. Robotların bilek eklemleri ile uç işlevcileri arasına eklenen kuvvet/tork algılayıcıları, robotların uç işlevcilerinin çevresi ile etkileşiminden kaynaklanan kuvveti ve torku ölçmek için kullanılmaktadırlar. Robotların işbirliği halinde çalışabilmeleri için etkileşim kuvvetlerinin ve uç işlevci hareketlerinin tek başına kontrol edilmesinin yetersiz kalacağı, her ikisinin de bir uyum içerisinde birlikte kontrol edilmesinin gerekli olduğu bilinmektedir. Kuvvet ve hareket kontrol yöntemlerinin bir çok farklı türü uyum kontrol methodları altında sınıflandırılabilir. Bu çalışmada sunulan ve uygulanan uyum kontrol yöntemlerinden ikisi literatürde *hibrit konum/kuvvet kontrolü* ve *paralel konum/kuvvet kontrolü* olarak adlandırılmıştır.

Bu çalışmada, önceki paragrafta belirtilen iki farklı uyum kontrol yöntemi tek bir robot manipülatörün önceden tanımlandırılmadığı statik bir çevrede ve aynı robot manipülatörün işbirliği içerisinde bir başka robot manipülatörün hareket etmesiyle değişen dinamik bir çevrede pratik uygulaması gerçekleştirilmiş ve deneysel sonuçları iki kontrol yöntemi arasında iki görev için de ayrı olarak karşılaştırılıp tartışılmıştır. Bu çalışmanın giriş bölümünde robotların tanımı yapılıp tarihteki yerlerinden kısaca bahsedilmiştir. Tezin amacı ve bu tez için kaynakça oluşturan çalışmaları içeren literatür incelemesi bu bölüm içerisinde sunulmuştur. Ayrıca bu bölümde sistem tanımlanması donanım ve yazılım olmak üzere iki bölümde yapılmıştır. Donanım altbölümünde sistemin tanımlanması karşılık geldikleri başlıkların altında robot manipülatörler, uç işlevci, kuvvet/tork algılayıcısı ve kontrolcüsü, robot kontrolcüsü ve son olarak da giriş/çıkış aygıtları olarak ayrıntılı bir biçimde yapılmıştır. Bu çalışmada robotun kontrol ve yörünge hesaplamalarının yapılması için oluşturulan

robot algoritmasının geliştirilmesi ve yürütülmesi boyunca kullanılan yazılımlar onlara karşılık gelen başlıkların altında tanımlanmıştır.

İkinci bölümde, robot manipülatörlerin kinematik ve dinamik matematiksel modelleri oluşturulmuştur. Öncelikle, matematiksel model oluşturmanın temelleri olarak koordinat çerçeveleri ve farklı türlerden temsil yöntemleri tanımlanmıştır. Robotun eklem uzayı ve görev uzayı koordinatları tanımlanmış, robotun uç işlevcisinin konumunun ve yönelimin temsil yöntemleri belirtilmiştir. Konum temsil yöntemleri için bu çalışmada kullanılan Kartezyen koordinatları tanımlanırken, diğer silindirik ve küresel koordinatlardan kısaca bahsedilmiştir. Yönelim temsil yöntemlerinden dönme matrisi ve minimal gösterim yöntemlerinden *angle-axis* ve birim kuaternion kısaca açıklanırken bu çalışmada kullanılan Euler açıları ile yönelim temsilin tanımlanması yapılıp, dönme matrisi ile dönüşümleri verilmiş, öteki minimal temsil yöntemlerine göre avantajlı ve dezavantajlı yönleri ortaya konulmuştur. Katı cisim dinamiğinde ortaya çıkan doğrusal/açısal hız ve kuvvet/tork vektörleri sırasıyla *twist* ve *wrench* olarak ifade edilen ve genel olarak bunun gibi 3 boyutlu uzayda bir eksen üzerinde öteleme ve dönme şeklinde gösterilebilen vektörleri *screw* olarak adlandırılan bir vektör çifti biçiminde ifade etmenin yöntemi gösterilmiştir. Bunun yanında dönme matrisinden *skew-symmetry* özelliği ile açısal hız vektörünün elde edilmesi ve doğrusal/açısal hızların başka gösterim yöntemleri ile eşlenmesi gösterilmiştir. Sonraki başlıkta kinematik ve dinamik modellemenin temellerini oluşturan sanal yerdeğiştirme ve sanal iş prensibi hakkında kısaca bahsedilmiştir. Ardından koordinat çevreveleri arasındaki dönüşümler için kullanılan homojen dönüşüm matrisleri ve Plücker koordinat sisteminde ifade edilen *screw* dönümüşümleri için kullanılan uzaysal dönüşüm matrisleri tanımlanmıştır. Robotların geometrik gösterimi için çoğunlukla kullanılan *Denavit-Hartenberg* yönteminin kuralları açık bir şekilde belirtilmiştir.

Gerekli matematiksel tanımlamalar yapıldıktan sonra robot kinematiği ve dinamiği modellenmeye hazır duruma getirilmiştir. Bu modellemelerde robotlara eklenen uç işlevci ve kuvvet/tork algılayıcısı da göz önünde bulundurulmuştur. Uç işlevcinin uygulanan kuvvete bağlı uzunluk değişimi deneysel olarak tanılanmış ve modellenmiştir. Robotların kinematik modeli, robotun hareketini farklı uzaylarda tanımlamaya yarar. Robotun eklem hareketlerini uç işlevcisinin hareketi olarak hesaplamak için ileri kinematik model, tersi için ise ters kinematik model kullanılmaktadır. Konum hesaplamaları yanı sıra, hız ve ivme hesaplamaları için birinci ve ikinci dereceden diferansiyel kinematik modellemeler de altbaşlıklarda tanımlanmıştır. Robotik uygulamalarda da kullanılan Jakobiyen terimi tanıtılmış, gövde üzerindeli herhangi bir noktaya göre türetilme yöntemleri gösterilmiş ve farklı kullanım alanları tanımlanmıştır. Jakobiyen matrisinin uç işlevciye göre geometrik ve analitik olarak iki farklı türü belirtilmiş ve birbirleriyle olan ilişkisi gösterilmiştir. Robotların dinamik modeli tüm gövdelere etki eden net kuvvet ve tork ile bu gövdelerin hareketleri arasındaki ilişkiyi ortaya koyar. Gövdelere uygulanan kuvvet/tork sonucunda ortaya çıkan gövde hareketi ters dinamik ile bu ilişkinin tersi ise ileri dinamik ile tanımlanmaktadır. Robot dinamiğini belirten hareket denklemi ve buradaki çeşitli dinamik etkiler tanımlanmıştır. Robotların dinamik modelinin türetilmesi için kullanılan *Euler-Lagrange* ve *Newton-Euler* yöntemlerinin temellerinden bahsedilmiş ve her eklem için sırayla kullanılan *Newton-Euler* denklemleri verilmiştir. Bunlara ek olarak eklem sürtünme ve yerçekimi dengeleme etkilerinin modellenmesinden de söz edilmiştir. Uç işlevcinin ağırlından kaynaklanan

etkinin kuvvet/tork algılayıcısının ölçüm değerlerinden çıkartılması için kullanılan yönetemler belirtilmiştir.

Bir sonraki bölümde, robotun hareket ve kuvvet kontrolü için kontrol sistemi tasarlanmıştır. İlk olarak kontrol sisteminde referans olarak kullanılacak yörüngeler tanımlanmıştır. Hareket yörüngeleri eklem uzayı ve görev uzayı olarak iki başlıkta anlatılmış, farklı kontol şemalarında kullanılma yöntemleri gösterilmiştir. Kontrolcü tasarımında hareket ve kuvvet kontrollerinin ayrı olarak ve uyum kontrol şemaları altında birlikte kullanımı gösterilmiştir. Hareket ve kuvvet konrol şemalarının ve uyum kontrol şemalarından hibrit konum/kuvvet kontrol ve paralel konum/kuvvet kontrol yöntemlerinin kontrol kanunu denklemleri ve blok diyagramları bu bölümde karşılık geldikleri başlıkların altında gösterilmiştir. Uyum kontrol şemalarında kontrol edilen eksenleri belirten uyum çerçevesi tanımlanmış, hibrit konum/kuvvet kontrol şemasında kullanılan uyum seçilim matrisi ve paralel konum/kuvvet kontrol şemasının temelini oluşturan empedans kontrol yöntemi gibi önemli konulara değinilmiştir. Uyum çerçevesinin belirlenmesi verilen görevin kısıtlamalarına göre seçilmesi gerekirken uyum seçilim matrisi bu çerçeve üzerindeki eksenlerin konum veya kuvvet kontrolünde olmasını belirler. Empedans kontrol yöntemi robotun uç işlevcisinin bir kütle-yay-damper sistemi gibi davranarak uç işlevcinin hareketi ile ona etkiyen kuvvet/tork arasında bir ilişki içerisinde davranmasını amaçlar. Paralel konum/kuvvet kontrol şeması ise empedans kontrolündeki dolaylı kuvvet kontrol yönteminin doğrudan kuvvet yöntemine çevirilmesi ile oluşmaktadır.

Bu çalışmada ele alınan iki uyum kontrol yönteminden hibrit konum/kuvvet kontolü ve paralel konum/kuvvet kontrolü, tek robot ve işbirlikçi robotlar ile uyum kontrolü görevlerinin pratik uygulanması için deneyler gerçekleştirilmiştir. Uygulanan iki deneysel görev de genel olarak serbest hareket eden robotlardan birinin uç-işlevcisinin katı bir cisime temas etmesi ardından kuvvet takibiyle birlikte uyumlu bir harekete geçmesi olarak tanımlanır. Bu iki deneysel görev de ilk görevin durağan bir çevrede gerçekleşmesi ve ikinci görevin ise ikinci bir robotun hareketinden kaynaklı dinamik bir çevrede gerçekleşmesi dışında özdeştir. Söz konusu iki uyum kontrol yöntemi için her iki görevin pratik uygulanmasından elde edilen deneysel sonuçlar, sayısal değerler ve grafikler verilerek bu bölümde sunulmuştur. Bunlara ek olarak, sistemin dinamik davranışının değişimleri, paralel konum/kuvvet kontrol yönteminin farklı kontrol parametreleri ile tekrarlanması yoluyla gözlemlenmiştir.

Sonuçlar bölümünde deneysel sonuçlar tartışılmış, değerlendirilmiş ve bu sonuçlar arasındaki farklar gösterilmiştir. Söz konusu iki uyum kontrol yönteminin hangi durumlarda daha yüksek konum ve kuvvet takibi performansları gösterdiği belirtilmiş ve bu anlamda çıkarımlar yapılmıştır. Daha sonra bu çalışmanın ve onun uygulamalarının eksiklikleri tanılanmış ve gelecek çalışmalar için olası geliştirmeler önerilmiştir.

1. INTRODUCTION

Robots are generally defined as programmable machines or devices carrying out actions in various levels of autonomy. The importance of robots comes up with their ability of achieving tasks humans cannot do manually. Robots can work in the environments that are dangerous or improbable for humans. They can be programmed to perform series of actions without continuous human handling. The controllability of robots allows them to complete complex tasks with high precision. The actuation and transmission system can produce high speed motion with high power.

The term "robot" was firstly used in fiction to describe artificial human-like creatures along with other different names. The idea of robot is dated back to legends and mythologies of several ancient civilizations. However, modern concept of robot began to be used after Industrial Revolution. The introduction of mass production came with a need of automated tasks. Modern industrial robots started to be used after Digital Revolution with the advances on electronics and invention of digital computing.

Robot manipulators are used for industrial applications such as pick and place, welding, painting and machining applications. Most of these applications take place under well-defined conditions by individual robots. The applications include uncertainties that require compliance in force and motion. Different control methods are proposed in literature to solve this problem [1] [2]. Designing an appropriate controller including motion and force controls with a variety of implicit and explicit combinations is essential [3]. However, the stability and performance of a task under physical interactions are challenging issues [4]. In order to compensate for the system uncertainties, model based or soft computing based methods are widely used [5] [6]. For situations including unknown or time-varying parameters, adaptive methods were developed [7].

As the industrial applications become more complicated, a single robot manipulator may not be sufficient for fulfillment of some dexterous tasks. Multiple robots can

be used for achieving a single goal. Moreover, a human can assist a robot or be assisted by a robot for a specialized task [8]. These different agents are required to be coordinated in order to work in harmony. This coordination is accomplished through several collaboration approaches.

1.1 Purpose of Thesis

The purpose of this thesis can be summarized as implementing compliance control schemes for collaborating robots and comparing the results about force and motion tracking. There are two compliance control schemes proposed: *hybrid position/force control* and *parallel position/force control*. These two control schemes are tested in a fixed environment and a dynamically changing environment by a second robot manipulator. The performances of these control schemes are compared across both environments and between each other. This comparison gives a perspective about which control scheme is favorable under specific circumstances.

1.2 Literature Review

The compliance control schemes for robotics were first discussed during the 1980s, refinements and variations have been done through the 1990s. The hybrid position/force control scheme in joint-space was originated from Craig and Raibert in 1981 [9], and later corrected formulation about "kinematic instability" was presented by Fisher in 1992 [10]. The use of hybrid position/force control scheme in operational-space was proposed by Khatib in 1987 [11]. The implementation of the mechanical impedance as a indirect force control method was originated from Hogan in 1985 [12]. Various implementations of impedance control scheme in robotics were presented as a hybrid impedance control by Anderson and Spong in 1988 [13] and a parallel approach to force/position control by Chiaverini and Sciavicco in 1993 [14].

Implementations of hybrid position/force control for 3-DoF and 6-DoF industrial robots were investigated in studies such as [15] and [16] respectively. This control method is also studied in parallel robot applications [17] [18]. Implementations of impedance control methods for 6-DoF industrial robots under undefined environments were investigated in Ott et al. (2010) and Jung et al. (2004) [19] [20]. A hybrid-mode impedance control method is implemented for robot-based rehabilitation in [21] and

[22]. Some machine learning algorithms are integrated in the control schemes of robot manipulators [23].

In order to achieve collaboration of multiple robots, there are main approaches like master-slave, centralized and decentralized control. In centralized control approach, collaborating robots are controlled under a unified control scheme. In contrary, the robots are controlled under independent control schemes in decentralized control approach. The master-slave approach gives a hierarchical relationship between collaborating robots. According to this approach, the general motion of robots is determined by the robot that is considered as the master. In the meanwhile, other robots that comply with the motion of the master through force inputs are considered as the slaves. An implementation of master-slave approach in multiple collaborating robots is shown by Sharifi at al. [24].

Coordination of multiple robots in collaborating tasks requires a compliance of force and motion control together. The control schemes evaluating the force and motion together are implemented in [25] for 3-DoF and [26] for 6-DoF collaborating robots. Some compliance control methods were used for physical human-robot collaboration tasks, which can be seen in [27] and [28].

1.3 System Description

The system used in this study is categorized under hardware and software. The first section describes the hardware, which consists of mechanical and electronic parts of the system. The second section describes the software running on the computational devices. The software are classified into two categories according to their functionalities about the robot algorithm. These functions of the software are execution and development of the robot algorithm.

1.3.1 Hardware

The hardware of a robot manipulator consists of two main parts: a robot arm and a robot controller. A force/torque sensor composed of a transducer and a controller is introduced to sense the contact forces. A specialized end-effector is mounted on the robot arm to interact with the environment. An input/output device is used for operational purposes. Other than the computer as an I/O device, an identical or similar

Figure 1.1 : Stäubli RX160L and RX160 model robot manipulators [29] [30].

pair of each of these hardware are available for a robot-robot collaboration. These hardware are described with their technical properties in the following subsections.

1.3.1.1 Robot manipulators

In this study, two robot manipulators are used for collaborating tasks. Stäubli RX160 and RX160L models seen in Figure 1.1 are the industrial robot manipulators located opposed to each other in the work place. The RX160 series robots are 6-axis articulated arms used for industrial automation with high-precision. The joints of the robot arm are actuated by brushless motors. Every joint's motion is tracked by a resolver and parking breaks are included for locking the positions of each joint. First four joints are equipped with cycloidal transmission and the joint bearing support. Last two joints are coupled at wrist and driven via a worm and wheel gear. Sixth joint also includes a bevel gear as a transmission component. The robot arms have an integrated spring mechanism on the (upper) arm link that counterbalances the weight of the links after the shoulder. RX160 and RX160L models differ by their forearms. While RX160 model has a standard sized forearm, RX160L model has an extended forearm for increased reachability. The load capacities for RX160 series robot arms are given as 34 kg and 28 kg for maximum load capacities and 20 kg and 16 kg at nominal speed respectively. More technical details about RX160 family robot arms are provided in Appendix A.

1.3.1.2 End-effector

A robot manipulator is desired to interact with the environment with its end point, because it is the highest maneuver-capable part of a manipulator. Every different kind

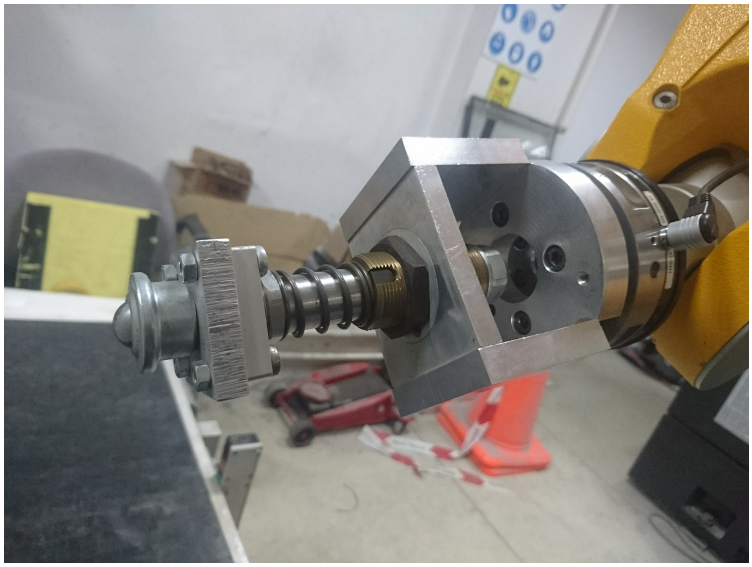

Figure 1.2 : End-effector.

of interactions with environment requires a unique hardware that can be mounted on the flange of the robot manipulator. In the setups that have a wrist F/T transducer, the end-effector is mounted on the F/T transducer. An appropriate end-effector is selected for the requirements of the given task, i.e. a gripper with mechanically moving parts for pick-and-place applications and a laser cutter for a cutting application. In practical implementations of this study, the interaction between the robot and environment is achieved through the end-effector making a contact with an object and moving while keeping the contact. Therefore, the end-effector used in this work, shown in Figure 1.2, consists of a spherical object and a spring mechanism for an easier interaction. While the end-effector is moving on the object, spherical object reduces the generated frictional forces by rolling in 3-DoF in its nest. The spring mechanism decreases the sudden changes of the applied forces on the end-effector via moving in 1-DoF.

1.3.1.3 F/T transducer and controller

Mechanical interactions with the end-effector of the robot produce forces and torque on the contact region. A wrist F/T transducer is mounted between the end-effector and the flange of the robot. The F/T transducer used in this study is an ATI Delta six-axis transducer shown in Figure 1.3. The F/T transducer uses semiconductor strain gauges to measure the forces and torque on six orthogonal axes. The ATI Delta transducer that is used in this study is calibrated according to SI-660-60. More technical details about ATI Delta F/T transducers are provided in Appendix B.

Figure 1.3 : ATI Delta transducer [31].

Figure 1.4 : ATI F/T controller [32].

An ATI F/T controller shown in Figure 1.4 is complementing the ATI F/T transducer. The transducer and controller are connected by an electrically shielded transducer cable. The F/T controller can be used for computations on strain gauge data obtained from the transducer, however in this study only analog output is used for transferring the strain gauge data. The analog data is converted from analog to digital on a WAGO terminal block and the digital data is sent through ethernet cable to ethernet port of the robot controller by Modbus TCP/IP protocol.

1.3.1.4 Robot controller

The robot controller for Staubli RX160 series robots is a CS8C controller model from CS8 controller series from the same manufacturer. CS8C robot controller is shown in Figure 1.5. A robot controller is a device that controls the power supply of the robot with processing the input data by running a control algorithm. CS8C controller has digital power amplifiers for each axis of the robot and a processor that controls those amplifiers.

Figure 1.5 : CS8C robot controller [33].

Safety stop channels are presented in the controller for robot and for a cell environment if it is defined. For connectivity, two serial ports and two ethernet ports are available on the controller device. One of the serial ports is used for data transfer with a computer that is used by human operator. An ethernet port is used for Modbus TCP/IP protocol communication with the F/T controller.

1.3.1.5 User input/output devices

Human operators use I/O devices to generate tasks for robot manipulators. These devices provide the connectivity between the human operator and a robot controller via taking inputs from the user and displaying feedback data from the controller.

For an industrial robot manipulator, a default user input device is a manual control pendant. Manual control pendant shown in Figure 1.6 is a handheld device that is used for generating input for the controller of the robot remotely. Pendants have emergency switches for safety of the user while the device is handling. The pendant that we use operates with a high-level interface and has a display for visual output and buttons for the user input. Even when the user uses other input devices, the emergency stop button of the pendant is strongly recommended to be ready to use.

The other user input device is a computer, which is used as the input device for this work. The desktop PC that is used in this work has components as a display for visual output and a keyboard for typing input. The communication between computer and the controller is achieved via serial ports.

Figure 1.6 : Manual control pendant [33].

1.3.2 Software

The software used in this work consists of operating systems, integrated development environments, compilers, serial console, programming interfaces, programming languages and numerical computing environments. These software are classified under execution and development of the robot algorithm. The robot algorithm developed for this study includes trajectory generation and control algorithms.

1.3.2.1 Controller operating system and interfaces

The robot controller CS8C runs a real-time operating system called VxWorks®. The version 5.5.1 of this OS is used in this study. Real-rime operating systems are used for applications that require computations completed on a strict time interval. Controlling of the robots have to be done with instantaneous feedback and input values, therefore real-time operating systems are commonly used in robotics. The default sampling frequency of the real-time algorithm is set to 250 Hz. CS8C robot controller supports two interfaces to run commands from two different programming languages. One of the interfaces is working with a high-level programming language named VAL3. It is a specialized scripting language for simplifying sophisticated tasks for human operators. Provided manual control pendants are using VAL3 programming on an easy to interactive environment.

The other interface is called LLI, which stands for low-level interface. It is an application program interface (API) for C/C++ programming language which is provided by the manufacturer. LLI provides functions and variables as input/output for a robot control application to power source and motor drives while checking the safety measurements. The robot algorithm written for LLI is interacted through a serial console from a computer.

1.3.2.2 Robot algorithm development and analyzing environments

The code is written in C by using LLI is developed in Microsoft® Visual Studio® IDE on Microsoft® Windows® OS. The versions of those software are specified as Visual Studio 2017 & 2019 and Windows 10 for this process. On Windows platform the code is compiled by Visual Studio's C/C++ compiler to test and simulate the robot program before using it on the robot's controller. To use the same code on the robot controller, which runs Wind River® VxWorks® OS, the code is compiled on Wind River® Tornado® IDE. The software versions for this process are given as Tornado 2.2 on Windows XP. The compiled code is transferred to robot controller via a USB flash drive.

The experimental data are written on the same USB flash drive as TXT files for every cycle of controller. The data on those files is analyzed and visualized on the MATLAB® R2019a numerical computing environment.

2. MATHEMATICAL MODELING

Mathematical models are used to demonstrate and predict a physical system. In robotics, the system can be defined on task (operational) space and joint spaces in their corresponding coordinates. Kinematics are used for computing operational space and joint space positions of a robot manipulator from one another. Differential kinematics describes the time variations of geometric variables of a robot manipulator. Dynamic models are used for relating general forces acting on a robot and its motion. These mathematical models are used for the analysis and the control of the robot. These different types of mathematical models are presented in the following sections.

2.1 Coordinate Frames and Representations

A robot manipulator can be described as an open kinematic chain of links connected with joints. Each of these joints has 1-DoF prismatic or revolute movement capability. The sequence of joints forms the movement capability of the robot's end-effector. Base of the robot manipulator can be fixed or floating for different types of robots. The configuration of the robot can be given in joint space and task spaces with different coordinates.

2.1.1 Joint space and task space coordinates

On the joint space, the configuration of a n-axis robot is described by a generalized coordinate vector $\boldsymbol{q}$ in equation (2.1) with n number of independent variables for decoupled manipulators.

$$\boldsymbol{q} = \begin{pmatrix} q_1 \\ \vdots \\ q_n \end{pmatrix} \tag{2.1}$$

Each of these variables corresponds to a joint of a manipulator. A variable for i^{th} joint q_i is defined as a rotation angle θ_i for a revolute joint, and a linear displacement d_i for a

prismatic joint. Joint space coordinates are related to task-space with frames placed on each joint axis overlapping with those frame's $\hat{z}$-axes. The details of this relationship are given in the geometric representation section.

A coordinate frame i in a three-dimensional space has an origin O_i and a triad of mutually orthogonal vectors $(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{y}}_i, \hat{\boldsymbol{z}}_i)$. A vector in a frame i relative to another frame j is denoted as ${}^{j}\boldsymbol{r}_i$ in this study. The vectors and matrices relative to base or inertial frames are neglected in some cases for simplicity.

In the task space of a robot, pose of the end-effector is described in its position and orientation. Position and orientation are shown w.r.t. a reference frame, which is generally chosen as base or inertial frame on a single fixed base robot. On mobile or multiple robots the inertial frame is located on a fixed arbitrary point in space. A body in a three-dimensional Euclidean space is represented in a special Euclidean group $SE(3)$. That is a special combination of a position vector $\boldsymbol{p}_e \in \mathbb{R}^3$ and a rotation matrix $\mathbf{R}_e \in SO(3)$. This combination is shown in equation (2.2) and it is described in matrix form via the homogeneous transformation matrices in the later sections.

$$\mathbf{x}_e = (\boldsymbol{p}_e, \mathbf{R}_e) \in SE(3) \tag{2.2}$$

The end-effector pose can be represented with different parameterizations for both position and orientation. The pose is parameterized in minimally six coordinates or seven coordinates with one extra redundant parameter depending on the orientation representation. The parameterized pose vector for end-effector in base or inertial frame $\boldsymbol{x}_e$ is given in equation 2.3.

$$\boldsymbol{x}_e = \begin{pmatrix} \boldsymbol{x}_{e_P} \\ \boldsymbol{x}_{e_R} \end{pmatrix} \in \mathbb{R}^m, \quad m = 6|7 \tag{2.3}$$

2.1.1.1 Position and orientation representations

The position of the robot's end-effector relative to its base frame is shown as a vector $\boldsymbol{p}_e$ in three-dimensional Euclidean space. This position vector can be represented in Cartesian, cylindrical or spherical coordinates. Cartesian coordinates are more commonly used in robotics and written as follows:

$$\boldsymbol{x}_{e_P} = \boldsymbol{p}_e = \begin{pmatrix} {}^0p_e^x \\ {}^0p_e^y \\ {}^0p_e^z \end{pmatrix} \in \mathbb{R}^3 \tag{2.4}$$

Orientation of a robot's end-effector can be represented with more variety than its position. While position has different representations with three coordinates, the coordinate numbers for orientation representations can be different. In the homogeneous transformation matrix representation, rotation is represented by an orthogonal matrix in $SO(3)$ special orthogonal group in the equation (2.5), meaning that a rotation is represented by 9 parameters but these parameters are dependent on the orthogonality conditions. The rotation matrix of a frame i relative to a frame j is shown as dot product of their basis vectors as follows:

$${}^j\mathbf{R}_i = \begin{bmatrix} \hat{\boldsymbol{x}}_i \cdot \hat{\boldsymbol{x}}_j & \hat{\boldsymbol{y}}_i \cdot \hat{\boldsymbol{x}}_j & \hat{\boldsymbol{z}}_i \cdot \hat{\boldsymbol{x}}_j \\ \hat{\boldsymbol{x}}_i \cdot \hat{\boldsymbol{y}}_j & \hat{\boldsymbol{y}}_i \cdot \hat{\boldsymbol{y}}_j & \hat{\boldsymbol{z}}_i \cdot \hat{\boldsymbol{y}}_j \\ \hat{\boldsymbol{x}}_i \cdot \hat{\boldsymbol{z}}_j & \hat{\boldsymbol{y}}_i \cdot \hat{\boldsymbol{z}}_j & \hat{\boldsymbol{z}}_i \cdot \hat{\boldsymbol{z}}_j \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \in SO(3) \tag{2.5}$$

There are rotation representations such as Euler angle, angle-axis and unit quaternion for representing a rotation in more minimal parameters.

A rotation in 3D space is minimally represented with three independent parameters such as Euler angles. Three parameters of Euler angles are temporarily named as phi, theta and psi in this study. These parameters are replaced with selected Euler angle sets. In Euler angle representation of the orientation of the end-effector, a rotation can be shown as three consequent elementary rotations on a reference coordinate frame.

These three rotations can be defined in two sets as symmetric and asymmetric. If the first and third rotation occur on the same rotating frame, that set is called as symmetric (e.g. ZXZ, ZYZ). If all rotations occur on different rotating frames, that set is called as asymmetric (e.g. XYZ, ZYX).

XYZ Euler angles are selected as an example to demonstrate the following transformations. The three sequenced rotations in XYZ Euler angles are shown in Figure 2.1. The equivalent rotation matrix of a rotation parameterized by XYZ Euler angles is shown as,

$$\mathbf{R}_e = \mathbf{R}_X(\phi)\mathbf{R}_Y(\vartheta)\mathbf{R}_Z(\psi) = \begin{bmatrix} c\vartheta c\psi & -c\vartheta s\psi & s\vartheta \\ c\phi s\psi + s\phi s\vartheta c\psi & c\phi c\psi - s\phi s\vartheta s\psi & -c\vartheta s\phi \\ s\phi s\psi - c\phi s\vartheta c\psi & s\phi c\psi + c\phi s\vartheta s\psi & c\phi c\vartheta \end{bmatrix} \tag{2.6}$$

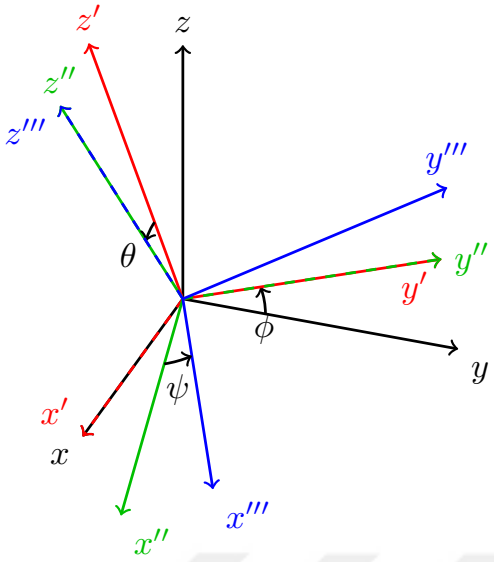

Figure 2.1 : Rotation on XYZ Euler angles.

where, for a given angle α, $\cos(\alpha)$ and $\sin(\alpha)$ are abbreviated as $c\alpha$ and $s\alpha$ respectively.

The rotation matrix is parameterized to XYZ Euler angle set is given in the following equation.

$$\boldsymbol{x}_{e_R,eulerXYZ} = \begin{pmatrix} \phi \\ \vartheta \\ \psi \end{pmatrix} = \begin{pmatrix} \text{atan2}(-r_{23}, r_{33}) \\ \text{atan2}(r_{13}, \sqrt{r_{11}^2 + r_{12}^2}) \\ \text{atan2}(-r_{12}, r_{11}) \end{pmatrix} \tag{2.7}$$

Without any redundant parameters, a singularity-free rotation on that space is not possible. An extra singularity named as gimbal lock is presented in Euler angle representation. This is caused by the loss of 1-DoF by first and third rotations that happen on the same axis on reference frame. To avoid the singularity caused by this representation, a different Euler angle set that is farther away from this singularity can be selected for that orientation [34].

Angle-axis and unit quaternion have one more parameter that eliminates that extra singularity but the singular configurations of robot still remain. While angle-axis and unit quaternion representations do not have the same singularity problem of Euler angles, they are relatively harder to implement and more computationally complex. Euler angle representation is preferred in this study due to its simplicity and orientational changes of the end-effector remain in a small range.

2.1.1.2 Screw representations of velocities and forces

Linear velocity of the end-effector is represented as vector $\boldsymbol{v}_e$ in task space. A linear velocity vector can be represented on Cartesian, cylindrical and spherical coordinates like in the case of a position vector. Linear mapping between coordinate frames can be achieved by a matrix $\mathbf{E}_P(\boldsymbol{x}_P)$. The time-derivative of position vector of the end-effector $\dot{\boldsymbol{x}}_{e_P}$ is linearly mapped as follows:

$$\boldsymbol{v}_e = \mathbf{E}_P(\boldsymbol{x}_{e_P})\dot{\boldsymbol{x}}_{e_P} \in \mathbb{R}^3 \tag{2.8}$$

For three-dimensional Cartesian coordinates, $\mathbf{E}_P(\boldsymbol{x}_{e_P})$ matrix equals to 3×3 identity matrix $\mathbf{1}_{3\times 3}$.

While angular displacement is not a proper vector in three-dimensional space, angular velocity can be described as a vector. Angular velocity vector has a duality relationship with an angular velocity tensor. Angular velocity tensor is a skew-symmetric matrix and can be derived in following equations.

$$\mathbf{S}(\boldsymbol{\omega}_e) = \dot{\mathbf{R}}\mathbf{R}^T = \begin{bmatrix} 0 & \omega_z & -\omega_y \\ -\omega_z & 0 & \omega_x \\ \omega_y & -\omega_x & 0 \end{bmatrix} \tag{2.9}$$

Time-derivative of rotation parameters are used for velocity representation on rotation parameterizations. A parameterized rotation vector is mapped to angular velocity vector as follows:

$$\boldsymbol{\omega}_e = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \mathbf{E}_R(\boldsymbol{x}_{e_R})\dot{\boldsymbol{x}}_{e_R} \tag{2.10}$$

The pairs of vectors such as linear and angular velocities or forces and torques can be described in pairs of vectors named *screws* according to the *screw theory*. It provides useful methods for mathematical computations in rigid body dynamics which is an essential part of robotics. A 6-dimensional representation of a pair of 3-dimensional vectors is named as a screw. The screw that is a pair of linear and angular velocities is called a twist. Likewise, the screw that is a pair of forces and moments is called a wrench. Twist $\mathbf{v}_A$ and wrench $\mathbf{f}_A$ of a point A about a fixed frame shown in Figure

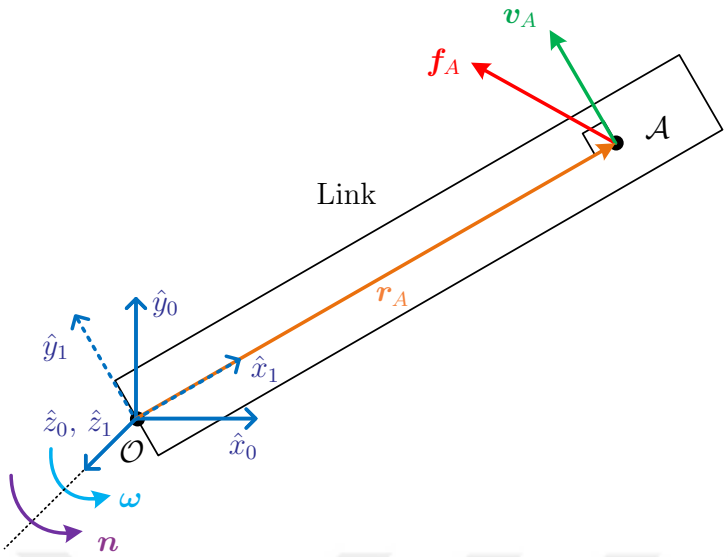

Figure 2.2 : Elements of twist and wrench shown on a rigid link.

2.2 are represented in Plücker axis and ray coordinate form of screws in the equations (2.11) and (2.12) respectively.

$$\mathbf{v}_A = \begin{pmatrix} \boldsymbol{v}_A \\ \boldsymbol{\omega} \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}_A \times \boldsymbol{\omega} \\ \boldsymbol{\omega} \end{pmatrix} \tag{2.11}$$

$$\mathbf{f}_A = \begin{pmatrix} \boldsymbol{f}_A \\ \boldsymbol{n} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_A \\ \boldsymbol{r}_A \times \boldsymbol{f}_A \end{pmatrix} \tag{2.12}$$

$\mathbf{E}(\boldsymbol{x}_e)$ is a mapping matrix composed from $\mathbf{E}_P(\boldsymbol{x}_{e_P})$ linear and $\mathbf{E}_R(\boldsymbol{x}_{e_R})$ angular mapping matrices. This mapping matrix has a form of block diagonal matrix as shown in the equation (2.13).

$$\mathbf{E}(\boldsymbol{x}_e) = \begin{bmatrix} \mathbf{E}_P(\boldsymbol{x}_{e_P}) & \mathbf{0} \\ \mathbf{0} & \mathbf{E}_R(\boldsymbol{x}_{e_R}) \end{bmatrix} \tag{2.13}$$

The vector stack of the linear velocity and the time-derivative representation parameters are mapped to the twist of the end-effector through $\mathbf{E}(\boldsymbol{x}_e)$ mapping matrix as shown in equation (2.14), which is also valid for mapping changes of pose terms in small time intervals approximately.

$$\mathbf{v}_e = \mathbf{E}(\boldsymbol{x}_e)\dot{\boldsymbol{x}}_e, \quad \Delta\mathbf{x}_e \approx \mathbf{E}(\boldsymbol{x}_e)\Delta\boldsymbol{x}_e \tag{2.14}$$

2.1.2 Virtual displacement and principle of virtual work

The kinematics and dynamics of a robot is derived from the virtual displacement and virtual work concepts. Extended information about *kinematics* and *dynamics* is given in their corresponding sections. A body is in equilibrium when total virtual work from external forces is zero for any virtual displacement for that body. The virtual displacement due to the variation of joints can be described as follows:

$$\begin{pmatrix} \delta \boldsymbol{p}_C \\ \delta \mathbf{R} \end{pmatrix} = \begin{pmatrix} \mathbf{J}_{CP} \\ \mathbf{J}_{CR} \end{pmatrix} \delta \boldsymbol{q} \tag{2.15}$$

In this equation, virtual displacement in terms of variation in position vector $\delta \boldsymbol{p}_C$ of a point C and the variation in rotation matrix $\delta \mathbf{R}$ are the infinitesimal changes in coordinates for a fixed instant in time. The principle of virtual work states that a constraint force $\boldsymbol{f}_C$ acted on a point $\boldsymbol{p}_C$ in the same direction does not contribute to any work.

$$\delta W = \delta \boldsymbol{p}_C^T \boldsymbol{f}_C = 0 \tag{2.16}$$

For a body system, the principle of virtual work is extended with the *d'Almbert's Principle*. Inverse dynamic formulations based on this extended principle due to its implementation on the rigid bodies.

2.1.3 Transformation matrices

A point in a three-dimensional space is transformed from one frame to another frame by a homogeneous transformation matrix. The homogeneous transformation matrix of coordinate frame i to coordinate frame j is denoted by ${}^{j}\mathbf{T}_i$. ${}^{j}\mathbf{T}_i$ is a 4×4 matrix combines translational ${}^{j}\boldsymbol{p}_i \in \mathbb{R}^3$ and rotational transformation ${}^{j}\mathbf{R}_i \in SO(3)$ as shown in equation (2.17). The bottom row of the transformation matrix shapes the matrix as square and equals to the bottom row of an identity matrix $\mathbf{1}_{4\times4}$ in an isometric space. That row is used for perspective transformations and scaling on graphical applications mostly.

$${}^{j}\mathbf{T}_i = \begin{bmatrix} {}^{j}\mathbf{R}_i & {}^{j}\boldsymbol{p}_i \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \tag{2.17}$$

A transformation of a point vector ${}^{i}\boldsymbol{r}$ to ${}^{j}\boldsymbol{r}$ is achieved through adding a dummy parameter as the last element of the position vectors in this process. This dummy parameter has a value of one and it is used in the transformations as follows:

$$\begin{pmatrix} {}^{j}\boldsymbol{r} \\ 1 \end{pmatrix} = {}^{j}\mathbf{T}_{i} \begin{pmatrix} {}^{i}\boldsymbol{r} \\ 1 \end{pmatrix} \tag{2.18}$$

The same point vector is transformed by ${}^{i}\mathbf{T}_{j}$ in the reverse direction. That homogeneous transformation matrix is equal to the inverse of the matrix ${}^{j}\mathbf{T}_{i}$ and can be shown as follows:

$${}^{j}\mathbf{T}_{i}^{-1} = {}^{i}\mathbf{T}_{j} = \begin{bmatrix} {}^{j}\mathbf{R}_{i}^{T} & -{}^{j}\mathbf{R}_{i}^{T}\,{}^{j}\boldsymbol{p}_{i} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{2.19}$$

A homogeneous transformation matrix ${}^{k}\mathbf{T}_{i}$ transforming from coordinate frame i to k can be described as a consecutive multiplications of multiple transformation matrices. This computation is shown as two transformation matrices with a intermediate frame j in the equation below.

$${}^{k}\mathbf{T}_{i} = {}^{k}\mathbf{T}_{j}\,{}^{j}\mathbf{T}_{i} \tag{2.20}$$

Transformations of screws such as twist and wrench occur in Plücker coordinates. The transformation from Plücker ray coordinate i to j is achieved with the spatial transformation matrix ${}^{j}\mathbf{X}_{i}$ given in equation (2.21). In this study, this is applied on wrench, which is described in this coordinate system. On the other hand, twist is described in Plücker axis coordinates in this study, which requires the transpose of that matrix ${}^{j}\mathbf{X}_{i}^{T}$ for the spatial transformation.

$${}^{j}\mathbf{X}_{i} = \begin{bmatrix} {}^{j}\mathbf{R}_{i} & \mathbf{0}_{3\times 3} \\ \mathbf{S}({}^{j}\boldsymbol{p}_{i})\,{}^{j}\mathbf{R}_{i} & {}^{j}\mathbf{R}_{i} \end{bmatrix} \tag{2.21}$$

In this matrix, $\mathbf{S}({}^{j}\boldsymbol{p}_{i})$ is the skew-symmetric matrix of the relative position vector ${}^{j}\boldsymbol{p}_{i}$ and ${}^{j}\mathbf{R}_{i}$ is the relative rotation matrix. For the reverse of this transformation, inverse of the ${}^{j}\mathbf{X}_{i}$ matrix is used.

$${}^{j}\mathbf{X}_{i}^{-1} = {}^{i}\mathbf{X}_{j} = \begin{bmatrix} {}^{i}\mathbf{R}_{j} & \mathbf{0}_{3\times 3} \\ -{}^{i}\mathbf{R}_{j}\mathbf{S}({}^{j}\boldsymbol{p}_{i}) & {}^{i}\mathbf{R}_{j} \end{bmatrix} \tag{2.22}$$

2.1.4 Geometric representation

In robotics, $Denavit - Hartenberg$ convention is used commonly for geometric representation of serial robots. The number of required parameters to locate one coordinate frame to another is reduced from six to four with this convention. A series of coordinate frames is assigned to the base and the moving links of a robot in an order by a set of rules. Links of the robot are assumed as perfectly rigid bodies. There are different ways of implementing the D-H convention and one of them suggested by *Khalil* and *Dombre* is described in this section [39]. According to this convention, coordinate frames are assigned to numbered links and joints for serial-chain mechanisms as described in Figure 2.3. The numbering of links and joints follows this convention:

- The base is numbered as 0, and n-number of moving links numbered from 1 to n.
- n-number of joints, with joint i located between link $i-1$ and i are numbered from 1 to n.

Attachment of coordinate frames to joints follows this convention:

- The $\hat{z}_i$ axis is located along the axis of joint i,
- The $\hat{x}_{i-1}$ axis is located along the common normal between the $\hat{z}_{i-1}$ and $\hat{z}_i$ axes.

The four parameters are given as two link parameters, the link length a_i and the link twist α_i, and two joint parameters, the joint offset d_i and the joint angle θ_i. These four parameters are defined in the following list.

- a_i is the distance from $\hat{z}_{i-1}$ to $\hat{z}_i$ along $\hat{x}_{i-1}$.
- α_i is the angle from $\hat{z}_{i-1}$ to $\hat{z}_i$ about $\hat{x}_{i-1}$.
- d_i is the distance from $\hat{x}_{i-1}$ to $\hat{x}_i$ along $\hat{z}_i$.
- θ_i is the angle from $\hat{x}_{i-1}$ to $\hat{x}_i$ about $\hat{z}_i$.

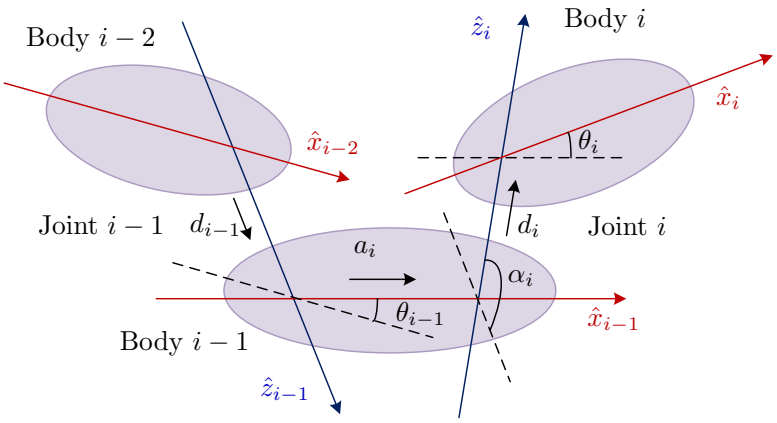

Figure 2.3 : Implementation of D-H convention on bodies and joints.

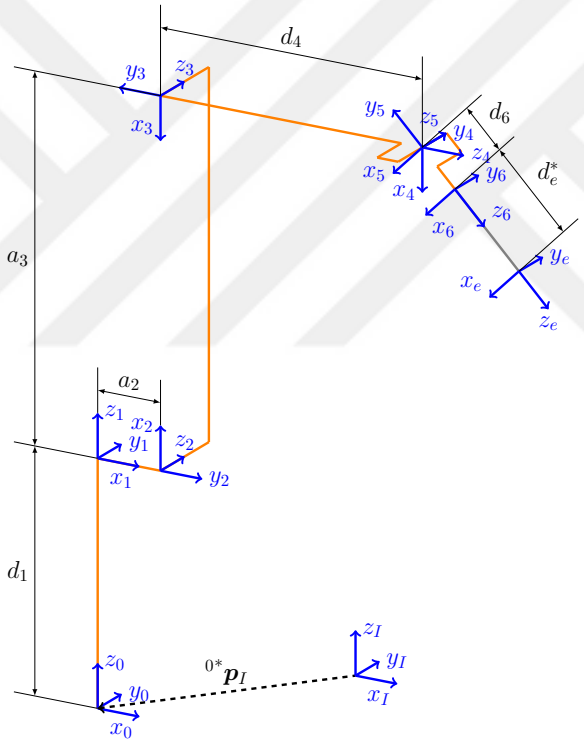

Figure 2.4 : Placement of coordinate frames on the robot joints, base and end-effector (for $\boldsymbol{q} = [0, 0, \pi/2, 0, \pi/4, 0]^T$).

Table 2.1 : D-H table for the RX160 series robot arms.

i	a_i (m)	α_i (rad)	d_i (m)	θ_i (rad)
1	0	0	0.55	θ_1
2	0.15	$-\pi/2$	0	$\theta_2 - \pi/2$
3	0.825	0	0	$\theta_3 + \pi/2$
4	0	$\pi/2$	d_4^*	θ_4
5	0	$-\pi/2$	0	θ_5
6	0	$\pi/2$	0.110	θ_6

*$d_4 = 0.925$ (RX160L), 0.625 (RX160)

The placement of the coordinate frames on the kinematic chain of RX160 family robot arms according to D-H convention is shown in Figure 2.4 as an example configuration. The placements of inertial frame and end-effector for each robot are described in the next section. The D-H parameters for the RX160 and RX160L robot manipulators are given in the Table 2.1.

Finally, the homogeneous transformation matrix from coordinate frame $i-1$ to i is constructed as shown in equation (2.23).

$$^{i-1}\mathbf{T}_i = \mathbf{Rot}(\hat{\boldsymbol{x}}_{i-1}, \alpha_i)\mathbf{Trans}(\hat{\boldsymbol{x}}_{i-1}, a_i)\mathbf{Rot}(\hat{\boldsymbol{z}}_i, \theta_i)\mathbf{Trans}(\hat{\boldsymbol{z}}_i, d_i) \tag{2.23}$$

The constructed homogeneous transformation matrices are used for obtaining the kinematics of the robot manipulator.

2.2 Kinematics

Kinematics relates the motion of bodies in a system without taking consideration of applied forces/torques. Kinematics in robotics deals with the relation between joint space coordinates and task space coordinates. Geometric locations of a robot in these spaces are given and converted to each other by a geometric model or simply kinematics. The time-variations of geometric variables are covered in the same manner by the differential kinematics. The conversion of variables from joint space to task space is defined as forward, and the reverse of it is defined as inverse. These conversions are achieved by matrices such as homogeneous transformation matrix and the Jacobian. The Jacobian matrix and its variants are used for relating velocities in different spaces and also force/joint torque. The detailed description of the Jacobian is given in following sections.

2.2.1 Forward kinematics

A geometrically represented end-effector pose of a robot can be computed from its joint positions by using forward kinematics. The mapping between joint coordinates $\boldsymbol{q}$ and end-effector of an n-DoF robot's configuration $\mathbf{x}_e$ is obtained by successive multiplications of homogeneous transformation matrices from inertial frame to end-effector frame as shown in equation (2.24).

$$ {}^{E}\mathbf{T}_I(\boldsymbol{q}) = {}^{0}\mathbf{T}_I \left(\prod_{k=1}^{n} {}^{k}\mathbf{T}_{k-1}(\boldsymbol{q}) \right) {}^{E}\mathbf{T}_n \tag{2.24} $$

For a fixed-base robot arm, transformation between inertial frame to base frame ${}^{0}\mathbf{T}_I$ is constant. In this study, the inertial frame is located at the base frame of the RX160 model robot for simplicity. The transformation matrix between inertial frame to base frame of RX160(1) and RX160L(2) models are shown in equation (2.25).

$$ {}^{0*}\mathbf{T}_I = \begin{bmatrix} \mathbf{1}_{3\times3} & {}^{0*}\boldsymbol{p}_I \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}, \quad \text{where} \quad {}^{0(1)}\boldsymbol{p}_I = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \text{and} \quad {}^{0(2)}\boldsymbol{p}_I = \begin{pmatrix} d_x \\ d_y \\ 0 \end{pmatrix} \tag{2.25} $$

In this equation, values of d_x and d_y are 2.43 m and -0.28 m respectively. The transformation matrix from inertial frame to base frame of the RX160(1) model robot ${}^{0(1)}\mathbf{T}_I$ simply equals to identity matrix $\mathbf{1}_{4\times4}$.

The transformation from the last link to end-effector ${}^{E}\mathbf{T}_n$ is constant on a rigid tool. However, the end-effector that is used in the experiments for RX160 model has a prismatic moving part that is changing dynamically. It can be assumed as constant under small forces even though it is not constant. The end-effector of the RX160L model is a gripper and has a fixed length. The transformation matrix of 6^{th} frame to end-effector frames of both RX160(1) and RX160L(2) are given in equation (2.26).

$$ {}^{E*}\mathbf{T}_6 = \begin{bmatrix} \mathbf{1}_{3\times3} & {}^{E*}\boldsymbol{p}_6 \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}, \text{ where } {}^{E(1)}\boldsymbol{p}_6 = \begin{pmatrix} 0 \\ 0 \\ d_e^{(1)}(f_e^z) \end{pmatrix} \text{ and } {}^{E(2)}\boldsymbol{p}_6 = \begin{pmatrix} 0 \\ 0 \\ d_e^{(2)} \end{pmatrix} \tag{2.26} $$

Position vector of the end-effector in 6^{th} frame ${}^{E}\boldsymbol{p}_6$ consists of summation of F/T transducer and end-effector length in z-axis. That length for RX160 model is a function of z-axis of end-effector force $d_e^{(1)}(f_e^z)$ and for RX160L that length $d_e^{(2)}$ has a constant value of 0.195 m. The maximum value of $d_e^{(1)}(f_e^z)$ occurs under there is no acting force in z-axis. The movement of the part is locked at maximum on tension. After a certain amount of force, the spring compresses completely. The identified length function of the spring mechanism with the end-effector and F/T transducer's length shown in Figure 2.5 is described in the equation (2.27), where f_e^z is simplified as f.

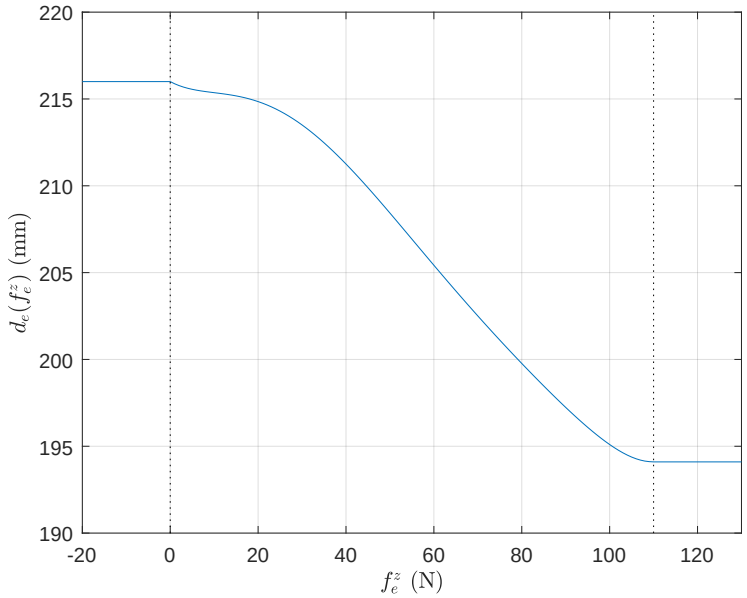

Figure 2.5 : Model of the F/T transducer and the end-effector length $d_e(f_e^z)$.

$$d_e^{(1)}(f) = \begin{cases} 0.216 \text{ m} & \text{if } f \leq 0 \text{ N}, \\ 0.216 - p_1 f^6 - p_2 f^5 - p_3 f^4 - p_4 f^3 - p_5 f^2 - p_6 f \text{ m} & \text{else if } f \leq 110 \text{ N}, \\ 0.194 \text{ m} & \text{otherwise.} \end{cases} \tag{2.27}$$

Parameters of the identified model of the end-effector's spring mechanism described in the equation (2.27) is given in the Table 2.2.

Table 2.2 : Parameters of the identified model of $d_e(f_e^z)$.

p_1	p_2	p_3	p_4	p_5	p_6
2.015e-13	7.036e-11	9.407e-09	5.66e-07	1.201e-05	0.000136

The resulting transformation matrix depending on joint coordinates ${}^E\mathbf{T}_I(\boldsymbol{q})$ can be written in the form of end-effector configuration. In order to describe the relationship between the joint positions and the end-effector pose, forward kinematics can be represented as a function $f(\boldsymbol{q})$ in the equation (2.28).

$$\mathbf{x}_e = f(\boldsymbol{q}), \quad \boldsymbol{x}_e = f_A(\boldsymbol{q}) \tag{2.28}$$

In consequence, this configuration can be represented in other end-effector configuration parameters such as Euler angles, angle-axis and unit quaternions.

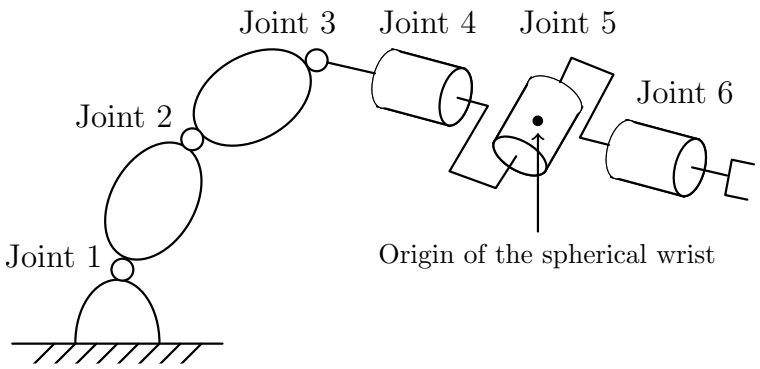

Figure 2.6 : 6-DoF robot arm with a spherical wrist.

2.2.2 Inverse kinematics

The inverse kinematics or the inverse geometric model provides the joint coordinates of a robot from its end-effector configuration in the task space. Inverse kinematics as a function can be represented as the inverse of the forward kinematics function $f(\boldsymbol{q})$ in the equation below.

$$\boldsymbol{q} = f^{-1}(\mathbf{x}_e) = f_A^{-1}(\boldsymbol{x}_e) \tag{2.29}$$

Computations of the inverse kinematics are getting more complex with the increasing number of freedoms of the system. The subjected end-effector should be located inside the workspace of the robot to compute inverse kinematics. The number of inverse kinematics solutions is depending on the configuration of the end-effector. Singular configurations of the robot arm can give undefined solutions and some of the solutions of inverse kinematics cannot be achieved due to physical limitations of the robot. For redundant robots with more than 6-DoFs can have infinitely many valid solutions, for a six-axis robot there can be finite number of solutions from inverse kinematics. The unique solution can be selected by evaluating the robot's current configuration and by other criteria like optimization or task planning. The geometrically impossible or hard-to-implement models can be computed by some iterative numerical techniques not presented in this study.

There are multiple methods to be used to obtain the inverse geometric model of a robot. Those methods are making simplifications by taking advantages of joint configurations of robots. In this study, the advantage of spherical wrist feature of the Stäubli RX160 robot manipulator is used to simplify some of the calculations. A simplified drawing of a 6-DoF robot arm with a spherical wrist is shown in Figure 2.6. This method leads to a decomposition of the 6-DoF problem in two 3-DoFs as position and orientation. In this method, first we compute the position of the center of the spherical wrist joint in terms of first three joints, and then we get the orientation in terms of last three joints.

The C-code of the inverse geometric model of a 6-DoF robot with a spherical wrist is presented in Appendix C.

2.2.3 Forward differential kinematics

The task space velocity of a point on the robot manipulator can be computed by multiplying joint space velocities with the Jacobian matrix for that point. That point subjected for a robot manipulator is generally chosen as the origin of the end-effector frame. The matrix mapping the joint space velocities to a twist is named as geometric or basic Jacobian.

$$\mathbf{v}_e = \mathbf{J}_e(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{2.30}$$

On small time intervals, the Jacobian approximately maps the change in joint position to change in task space position.

$$\Delta\mathbf{x}_e \approx \mathbf{J}_e(\boldsymbol{q})\Delta\boldsymbol{q} \tag{2.31}$$

The Jacobian mentioned above is the geometric Jacobian of the end-effector w.r.t. the base. In addition to this, the Jacobian of any frame originated at point C on link k of an n-DoF robot is described in following equations.

$$\mathbf{J}_C(\mathbf{q}) = \begin{bmatrix}\mathbf{J}_{CP}\\ \mathbf{J}_{CR}\end{bmatrix} = \begin{bmatrix}\boldsymbol{J}_{CP_i} & \cdots & \boldsymbol{J}_{CP_n}\\ \boldsymbol{J}_{CR_i} & \cdots & \boldsymbol{J}_{CR_n}\end{bmatrix} \tag{2.32}$$

The geometric Jacobian is a $6 \times n$ matrix consists of $\boldsymbol{J}_{CP_i}$ and $\boldsymbol{J}_{CR_i}$ 3×1 vectors as for each joint $i = 1 \ldots n$. These vectors are grouped as the position Jacobian $\mathbf{J}_{CP}$ and the

rotation Jacobian $\mathbf{J}_{CR}$ matrices sized as $3 \times n$. The joint velocities are related to the linear and angular velocities of the selected frame w.r.t. the base with the position and the rotation Jacobian matrices of that frame relatively. Eventually, the Jacobian matrix is a combination of position and rotation Jacobian matrices. Positional and rotational parts of the Jacobian is ordered in the same Plücker coordinate form of the twist.

The geometric Jacobian is derived using the arm geometry. The position Jacobian vector for link i with a revolute joint is derived as,

$$\boldsymbol{J}_{CP_i} = \begin{cases} \hat{\boldsymbol{z}}_i \times (\boldsymbol{r}_C - \boldsymbol{O}_i) & \text{if } i \leq k, \\ 0 & \text{otherwise.} \end{cases} \tag{2.33}$$

Where r_C is the position vector of point C relative to the base frame. Similarly, the rotation Jacobian vector link i with a revolute joint is derived as,

$$\boldsymbol{J}_{CR_i} = \begin{cases} \hat{\boldsymbol{z}}_i & \text{if } i \leq k, \\ 0 & \text{otherwise.} \end{cases} \tag{2.34}$$

For the kinematic computations of the end-effector, this frame is selected as the end-effector frame. In dynamics, the Jacobian matrices for CoM of each link are derived in order to compute the dynamic effects.

The orientation and the rotational velocities of the end-effector can be represented as described in a previous section. While the joint space velocities $\dot{\boldsymbol{q}}$ are mapped to angular velocities of the end-effector w.r.t. the base frame by using geometric Jacobian $\mathbf{J}_e(\boldsymbol{q})$, another type of Jacobian is required for mapping of $\dot{\boldsymbol{q}}$ to time-derivative of the minimal rotation representation parameters. In order to achieve this minimal rotation representation mapping, the matrix called analytic (task) Jacobian $\mathbf{J}_A(\boldsymbol{q})$ is used [37]. The analytic Jacobian is derived by partial differentiation of position and orientation parameterization vector for each joint coordinate as given in the following equation.

$$\mathbf{J}_A(\boldsymbol{q}) = \begin{bmatrix} \frac{\partial \boldsymbol{x}_{e_1}}{\partial q_1} & \cdots & \frac{\partial \boldsymbol{x}_{e_1}}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{x}_{e_m}}{\partial q_1} & \cdots & \frac{\partial \boldsymbol{x}_{e_m}}{\partial q_n} \end{bmatrix} \tag{2.35}$$

The number of rows n is depending on the DoF of the robot, and the number of columns m is depending on the number of parameters on the position and orientation representation vector $\boldsymbol{x}_e$. This leads to the analytic Jacobian to be a $m \times n$ matrix.

Due to the fact that the analytic Jacobian is used only for the end-effector w.r.t. the base frame, there is no need to specify it like the geometric Jacobian in this study. The analytic Jacobian is obtained from the geometric Jacobian of the end-effector by using inverse of the $\mathbf{E}(\boldsymbol{x}_e)$ mapping matrix as shown in the following equation.

$$\mathbf{J}_A(\boldsymbol{q}) = \mathbf{E}^{-1}(\boldsymbol{x}_e)\mathbf{J}_e(\boldsymbol{q}) \tag{2.36}$$

The stack of first-order time-derivatives of linear and rotation representation parameters of end-effector $\dot{\boldsymbol{x}}_e$ can be computed using $\mathbf{J}_A(\boldsymbol{q})$ as follows:

$$\dot{\boldsymbol{x}}_e = \begin{pmatrix} \dot{\boldsymbol{x}}_{eP} \\ \dot{\boldsymbol{x}}_{eR} \end{pmatrix} = \mathbf{J}_A(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{2.37}$$

For computing the end-effector acceleration as time-derivative of the twist $\mathbf{a}_e$ or the stack of second-order time-derivative of position and rotation representation parameters $\ddot{\boldsymbol{x}}_e$, both sides of equation (2.36) and equation (2.42) are taken respectively as follows:

$$\mathbf{a}_e = \begin{pmatrix} \boldsymbol{a}_e \\ \boldsymbol{\alpha}_e \end{pmatrix} = \dot{\mathbf{J}}_e(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \mathbf{J}_e(\boldsymbol{q})\ddot{\boldsymbol{q}} \tag{2.38}$$

$$\ddot{\boldsymbol{x}}_e = \begin{pmatrix} \ddot{\boldsymbol{x}}_{eP} \\ \ddot{\boldsymbol{x}}_{eR} \end{pmatrix} = \dot{\mathbf{J}}_A(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \mathbf{J}_A(\boldsymbol{q})\ddot{\boldsymbol{q}} \tag{2.39}$$

The vectors $\mathbf{a}_e$ and $\ddot{\boldsymbol{x}}_e$ is directly mapped as shown in the following equation by deriving the both sides of equation (2.38) w.r.t. time.

$$\mathbf{a}_e = \dot{\mathbf{E}}(\boldsymbol{x}_e, \dot{\boldsymbol{x}}_e)\dot{\boldsymbol{x}}_{\boldsymbol{e}} + \mathbf{E}(\boldsymbol{x}_e)\ddot{\boldsymbol{x}}_{\boldsymbol{e}} \tag{2.40}$$

The Jacobian is useful for not only velocity mapping but also singularity analysis and relating link wrenches and joint torques. In singular configurations, the rank of the Jacobian is smaller than the number of controllable DoFs.

2.2.4 Inverse differential kinematics

The inverse kinematics is used for computing the joint space velocities for a given task space velocity. In order to obtain joint space velocities, equation (2.36) is solved for $\dot{\boldsymbol{q}}$ by passing the geometric Jacobian to the other side of the equality as its inverse. The same is valid with the analytic Jacobian for obtaining joint space velocities from task space velocity of the end-effector in terms of time-derivative of the rotation representation parameters as follows:

$$\dot{\boldsymbol{q}} = \mathbf{J}_e^{-1}(\boldsymbol{q})\mathbf{v}_e = \mathbf{J}_A^{-1}(\boldsymbol{q})\dot{\boldsymbol{x}}_e \tag{2.41}$$

This computation requires the inversion of a selected type of Jacobian. Computation of the inverse of a Jacobian is not possible for singular or computed with a high error for close to singular configurations. Computations for close to singular configurations result with extremely high joint velocities. To minimize the least square error in this computations, the Moore-Penrose inverse method can be used. In singular configurations, inverse of the Jacobian does not exist, because the determinant of the matrix vanishes. Some of the singularities can be avoided by choosing different configurations when planning the motion. Additional singular conditions exist for the analytic Jacobian if the Euler angles is chosen for rotation representation. This type of singularities are caused by when the determinant of the $\mathbf{E}(\boldsymbol{x}_e)$ mapping matrix vanishes. In order to avoid this type of singularity, the Euler angle set is selected by examining the end-effector orientation. To achieve this, different sets of Euler angles that farther away from singularity for current end-effector orientation can be selected.

For the second-order inverse kinematics, the joint space accelerations can be obtained from solving equation (2.46) for $\ddot{\boldsymbol{q}}$ by taking derivative of both sides as follows:

$$\ddot{\boldsymbol{q}} = \dot{\mathbf{J}}_e^{-1}(\boldsymbol{q},\dot{\boldsymbol{q}})\mathbf{v}_e + \mathbf{J}_e^{-1}(\boldsymbol{q})\mathbf{a}_e = \dot{\mathbf{J}}_A^{-1}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{x}}_e + \mathbf{J}_A^{-1}(\boldsymbol{q})\ddot{\boldsymbol{x}}_e \tag{2.42}$$

A second formulation derived from equation (2.44) is more desired due to lack of inversion of the time-derivative Jacobian computation as follows:

$$\ddot{\boldsymbol{q}} = \mathbf{J}_e^{-1}(\boldsymbol{q})[\mathbf{a}_e - \dot{\mathbf{J}}_e(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}] = \mathbf{J}_A^{-1}(\boldsymbol{q})[\ddot{\boldsymbol{x}}_e - \dot{\mathbf{J}}_A(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}] \tag{2.43}$$

2.3 Dynamics

Dynamics of a system describes the relationship between actuation and applied forces/torques and the motion of the bodies. The dynamic models are classified according to the cause and effect relationship between forces and motion. The motion response of a system to any applied forces is modeled by using forward dynamics. It is used mostly on simulations for analyzing purposes. The inverse dynamics computes the required actuation torques of a system in order to achieve a motion. For improvements on robot control and trajectory planning, the inverse dynamic model is used commonly. The system models are computed analytically or numerically with using dynamical parameters. These dynamic parameters such as mass, center of mass and inertia tensor of the links have to be identified if they are not already available.

Dynamics of a robot can be modeled in either its joint space or task (operational) space. While joint space dynamics is simpler to be modeled and implemented in a joint space controller, task space dynamics is more straight-forward to implemented in a task space controller. However derivation of task space dynamic terms requires more computation due to the fact that they derive from their joint space counterparts.

2.3.1 Inverse dynamics

The inverse dynamics computes the joint torques required for the execution of a planned trajectory for a robot. In general, an inverse dynamic model is used in robot control and trajectory planning to increase the performance for tasks require high robot joint accelerations. In order to have a dynamic model approximates the physical system closely, all of the dynamic effects required to be modeled as possible. The inverse dynamic relation of these dynamic effects can be demonstrated with joint torques in joint space or with end-effector wrench in task space. The joint space EoM in compact form is shown as follows:

$$\boldsymbol{\tau} = \mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \mathbf{f}_e) \tag{2.44}$$

Where for an n-DoF robot,

- $\mathbf{M}(\boldsymbol{q})$ $\in \mathbb{R}^{n\times n}$, Matrix of the inertia of the robot,
- $\mathbf{h}(\boldsymbol{q},\dot{\boldsymbol{q}},\mathbf{f}_e)$ $\in \mathbb{R}^n$, Vector of dynamic effects in terms of $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\mathbf{f}_e$,
- $\boldsymbol{\tau}$ $\in \mathbb{R}^n$, Vector of joint actuation torques.

This compact form of the EoM gives how the joint actuation torques is related with the joint actuation torques, the joint variables included general coordinates with their first and second time-derivatives and the wrench exerted by the end-effector. It is important to note that this form of EoM in (2.44) is used for deriving the forward dynamics. The $\mathbf{h}(\boldsymbol{q},\dot{\boldsymbol{q}},\mathbf{f}_e)$ vector is the sum of the torque vectors from contact forces and the joint position and velocity dependent effects. The components of this vector can be used together or separately as a compensation in robot control schemes in order to improve the performance of the control algorithm. This vector is decomposed as,

$$\mathbf{h}(\boldsymbol{q},\dot{\boldsymbol{q}},\mathbf{f}_e) = \mathbf{b}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \mathbf{g}(\boldsymbol{q}) + \boldsymbol{\tau}_e(\boldsymbol{q},\mathbf{f}_e) + \boldsymbol{\tau}_{frc}(\dot{\boldsymbol{q}}) + \boldsymbol{\tau}_{spr}(\boldsymbol{q}) \tag{2.45}$$

where,

- $\mathbf{b}(\boldsymbol{q},\dot{\boldsymbol{q}})$ $\in \mathbb{R}^n$, Vector of Coriolis and centrifugal torque,
- $\mathbf{g}(\boldsymbol{q})$ $\in \mathbb{R}^n$, Vector of gravitational torque,
- $\boldsymbol{\tau}_e(\boldsymbol{q},\mathbf{f}_e)$ $\in \mathbb{R}^n$, Vector of torque caused by the forces exerted by the end-effector,
- $\boldsymbol{\tau}_{frc}(\dot{\boldsymbol{q}})$ $\in \mathbb{R}^n$, Vector of joint friction torque,
- $\boldsymbol{\tau}_{spr}(\boldsymbol{q})$ $\in \mathbb{R}^n$, Vector of spring torque.

The inverse dynamics can be formulated by Euler-Lagrange and Newton-Euler formulations. These are two different approaches that give identical results in terms of joint torques. Both of these formulations has its own advantages and inconveniences as described in their own sections.

While Euler-Lagrange and Newton-Euler formulations covers modeling of most of the dynamics of a robot, some dynamic effects requires modeling separately. One of that dynamic effects is the joint friction and other one is spring mechanism founded in the RX160 series robots.

In addition to the joint space formulation of the EoM in (2.44), the EoM can be demonstrated at the end-effector in the task space form as,

$$\mathbf{f}_e = \boldsymbol{\Lambda}_e(\mathbf{x}_e)\mathbf{a}_e + \boldsymbol{\mu}(\mathbf{x}_e, \mathbf{v}_e) + \boldsymbol{\rho}(\mathbf{x}_e) \tag{2.46}$$

where in a 3-D Cartesian space,

- $\boldsymbol{\Lambda}_e(\mathbf{x}_e) = (\mathbf{J}_e\mathbf{M}^{-1}\mathbf{J}_e^T)^{-1}$ $\in \mathbb{R}^{6\times6}$, Generalized inertia (mass) matrix,
- $\boldsymbol{\mu}(\mathbf{x}_e, \mathbf{v}_e) = \boldsymbol{\Lambda}_e\mathbf{J}_e\mathbf{M}^{-1}\mathbf{b} - \boldsymbol{\Lambda}_e\dot{\mathbf{J}}_e\dot{\boldsymbol{q}}$ $\in \mathbb{R}^6$, Vector of Coriolis and centrifugal terms,
- $\boldsymbol{\rho}(\mathbf{x}_e) = \boldsymbol{\Lambda}_e\mathbf{J}_e\mathbf{M}^{-1}\mathbf{g}$ $\in \mathbb{R}^6$, Vector of gravity terms,
- $\mathbf{f}_e$ $\in \mathbb{R}^6$, Generalized force vector.

Using task space inverse dynamics in a task space controller simplifies the controller structure, because the space transformations occur on the dynamic terms. However, deriving the task space dynamic terms requires additional computations due to the fact that those terms derived from joint space dynamics.

2.3.1.1 Euler-Lagrange method

One of the methods for formulating the EoM of a robot is the Euler-Lagrange method. It is an energy-based formulation originated from analytical mechanics. This method is useful for analyzing the dynamic effects due to the closed-form description of EoM as follows:

$$\boldsymbol{\tau}^* = \mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \mathbf{g}(\boldsymbol{q}) + \boldsymbol{\tau}_e(\boldsymbol{q}, \mathbf{f}_e) \tag{2.47}$$

The result of this formulation $\boldsymbol{\tau}^*$ in equation (2.48) is the joint actuation torque vector excluding the joint friction and the spring torque vectors.

$$\boldsymbol{\tau}^* = \boldsymbol{\tau} - (\boldsymbol{\tau}_{frc} + \boldsymbol{\tau}_{spr}) \tag{2.48}$$

The Coriolis and centrifugal torque vector $\mathbf{b}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is founded as $\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ in this formulation, where $\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{n\times n}$ is the Coriolis and centrifugal matrix. The torque

vector from end-effector wrench $\boldsymbol{\tau}_e(\boldsymbol{q}, \mathbf{f}_e)$ is computed using virtual work principle as follows:

$$\boldsymbol{\tau}_e(\boldsymbol{q}, \mathbf{f}_e) = \mathbf{J}_e^T(\boldsymbol{q})\mathbf{f}_e \tag{2.49}$$

This method is based on the Lagrangian function L and the Euler-Lagrange equation given below.

Lagrangian function:

$$L = T - U \tag{2.50}$$

Euler-Lagrange equation:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \left(\frac{\partial L}{\partial q_i}\right) = \tau_i \text{ for } i = 1 \ldots n \tag{2.51}$$

In the Lagrangian function, T is kinetic energy and U is the potential energy of the system. In robotics, the Lagrangian is shown as a function of generalized coordinate $\boldsymbol{q}$, generalized velocity $\dot{\boldsymbol{q}}$ and time as $L(t, \boldsymbol{q}, \dot{\boldsymbol{q}})$. Likewise kinetic energy and potential energy functions are shown as $T(t, \boldsymbol{q}, \dot{\boldsymbol{q}})$ and $U(t, \boldsymbol{q})$ respectively.

After the Euler-Lagrange equation is applied to each joint, the terms $\mathbf{M}(\boldsymbol{q})$, $\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and $\mathbf{g}(\boldsymbol{q})$ are obtained from the computations shown in the works of Akbaş [35] and Szczesiak [42].

2.3.1.2 Newton-Euler method

Another method for formulating the EoM of a robot is the Newton-Euler method. It is a recursive formulation based on the conservation of linear and angular momentum. For a single body, the change in linear and angular momentum and resultant forces and torques are related as following the Newton and Euler formulations:

$$\begin{pmatrix} \dot{\mathbf{p}}_C \\ \dot{\mathbf{N}}_C \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_{ext,C} \\ \boldsymbol{\tau}_{ext} \end{pmatrix} \tag{2.52}$$

The C is a point on the CoG for that body in equation (2.52). In this equation, $\boldsymbol{f}_{ext,C}$ is the resultant external forces acting on the point C and $\boldsymbol{\tau}_{ext}$ is the resultant external torques acting on that body. Integrating these with the principle of virtual work gives equation (2.53).

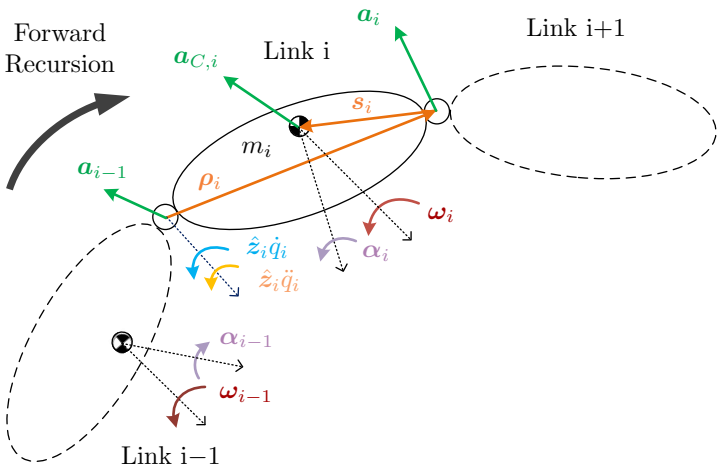

Figure 2.7 : Velocity and acceleration vectors for link i.

$$0 = \delta W = \begin{pmatrix} \delta p_C \\ \delta \mathbf{R} \end{pmatrix}^T \left(\begin{pmatrix} \dot{\mathbf{p}}_C \\ \dot{\mathbf{N}}_C \end{pmatrix} - \begin{pmatrix} \boldsymbol{f}_{ext,C} \\ \boldsymbol{\tau}_{ext} \end{pmatrix} \right) \quad \forall \begin{pmatrix} \delta p_C \\ \delta \mathbf{R} \end{pmatrix} \tag{2.53}$$

The change in linear and angular momentum vectors are shown in open form as follows:

$$\begin{pmatrix} \dot{\mathbf{p}}_C \\ \dot{\mathbf{N}}_C \end{pmatrix} = \begin{bmatrix} \mathbf{1}_{3\times3} m & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_C \end{bmatrix} \begin{pmatrix} \boldsymbol{a}_C \\ \boldsymbol{\alpha} \end{pmatrix} + \begin{pmatrix} \mathbf{0}_{3\times1} \\ [\boldsymbol{\omega}]_\times \mathbf{I}_C \boldsymbol{\omega} \end{pmatrix} \tag{2.54}$$

For multi-body systems like robot manipulators, the computations are performed in two successive recursions. The first recursion shown in Figure 2.7 is the computation of velocities and accelerations for each link. This computation is called forward recursion because it is performed for each link starting from base to end. Forward recursion equations for each joint i from 1 to n are given as follows:

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1} + \hat{\boldsymbol{z}}_i \dot{q}_i \tag{2.55}$$

$$\boldsymbol{\alpha}_i = \boldsymbol{\alpha}_{i-1} + \hat{\boldsymbol{z}}_i \ddot{q}_i + \boldsymbol{\omega}_{i-1} \times (\hat{\boldsymbol{z}}_i \dot{q}_i) \tag{2.56}$$

$$\boldsymbol{a}_i = \boldsymbol{a}_{i-1} + \boldsymbol{\alpha}_i \times \boldsymbol{\rho}_i + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \boldsymbol{\rho}_i) \tag{2.57}$$

$$\boldsymbol{a}_{C,i} = \boldsymbol{a}_i + \boldsymbol{\alpha}_i \times \boldsymbol{s}_i + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \boldsymbol{s}_i) \tag{2.58}$$

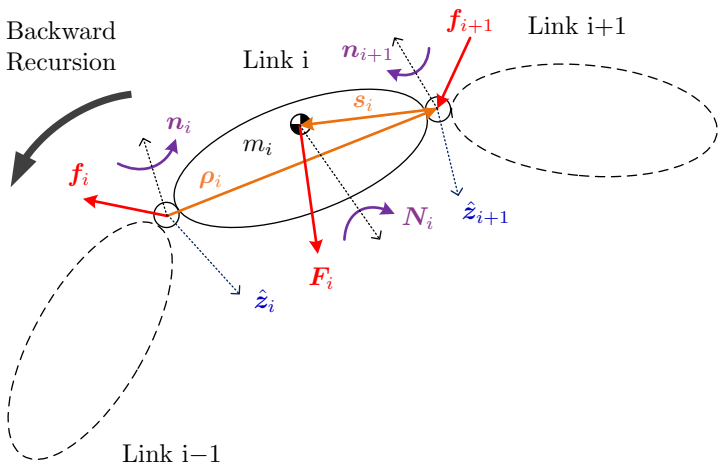

Figure 2.8 : Force and moment vectors for link *i*.

In the forward recursion equations, angular velocity $\boldsymbol{\omega}_0$ and angular acceleration $\boldsymbol{\alpha}_0$ are equal to zero vector $\mathbf{0}_{3\times1}$ because the base of the robot is fixed. Gravity can be added into the forward recursion equations via linear acceleration of base in (2.59).

$$\boldsymbol{a}_0 = \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} \tag{2.59}$$

The later recursion is computation of forces and torques acted on each link. This computation is called backward recursion because the computation of forces and torques starts from the external wrench acting on the end-point and ends on base link as shown in Figure 2.8. Backward recursion equations for each joint *i* from *n* to 1 are given as follows:

$$\boldsymbol{F}_i = m_i\, \boldsymbol{a}_{C,i} \tag{2.60}$$

$$\boldsymbol{N}_i = \boldsymbol{I}_{C,i}\, \boldsymbol{\alpha}_i + \boldsymbol{\omega}_i \times (\boldsymbol{I}_{C,i}\, \boldsymbol{\omega}_i) \tag{2.61}$$

$$\boldsymbol{f}_i = \boldsymbol{f}_{i+1} + \boldsymbol{F}_i \tag{2.62}$$

$$\boldsymbol{n}_i = \boldsymbol{n}_{i+1} + \boldsymbol{\rho}_i \times \boldsymbol{f}_{i+1} + (\boldsymbol{\rho}_i + \boldsymbol{s}_i) \times \boldsymbol{F}_i + \boldsymbol{N}_i \tag{2.63}$$

In the backward recursion equations, the contact force and moment can be included by equating them to last acting force and moment as in equations (2.64) and (2.65).

Finally, the input torque for each joint i is computed from following equation:

$$\boldsymbol{f}_{n+1} = -\boldsymbol{f}_e = \boldsymbol{f}_c \tag{2.64}$$

$$\boldsymbol{n}_{n+1} = -\boldsymbol{n}_e = \boldsymbol{n}_c \tag{2.65}$$

In these equations contact wrench acting on the end-effector $\mathbf{f}_c$ is equal to the wrench exerted by the end-effector $\mathbf{f}_e$ in the negative direction according to Newton's third law.

$$\boldsymbol{\tau}_i^* = \boldsymbol{n}_i \cdot \hat{\boldsymbol{z}}_i \tag{2.66}$$

This formulation gives the identical result $\boldsymbol{\tau}^*$ from EoM (2.47) computed with the Euler-Lagrange formulation.

$$\boldsymbol{\tau}^* = NE(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, g, \mathbf{f}_e) \tag{2.67}$$

As opposed to Euler-Lagrange formulation, Newton-Euler formulation does not give the EoM in closed-form. Since this formulation gives the result directly, it is commonly preferred for control schemes that require real-time computations. However, it is possible to compute the vectors from closed-form EoM by manipulating the inputs of $NE(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, g, \mathbf{f}_e)$ function. Similar to E-L formulation, this makes it possible to analyze the dynamics effects separately. It is also necessary for using N-E method for forward dynamics. The details are given in that section for these computations.

A different approach to N-E formulation named *Projected Newton-Euler method* that combines the dynamic equilibrium in Cartesian coordinates with the constraint compliant Euler-Langrange formulation using generalized coordinates [38]. This method allows to compute $\mathbf{M}(\boldsymbol{q})$, $\mathbf{b}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and $\mathbf{g}(\boldsymbol{q})$ terms in order to obtain EoM in closed-form without calling back N-E function repetitively.

2.3.1.3 Joint friction and balancing forces

It is shown that both E-L and N-E formulations compute a joint torque vector without taking other dynamic effects into consideration. These effects have a significant impact on the accuracy of the dynamic model of the robot. The inclusion of these dynamic effects to the control scheme improves its performance. One of these effects is the joint friction and the other one is the balancing forces.

The motion of the joints generates friction at the surface of the moving parts. The joint friction becomes a dominant effect on the joint motions starting from rest and changing velocity. A joint friction model approximates the actual system more with the inclusion of different types of friction models such as Coulomb, static and viscous frictions. The joint friction model used in this study includes these stated friction models. The friction parameters used in the joint friction model are identified by Zengin [43] and later the model is modified by Akbaş.

The first four joints of the robot are decoupled, so that their friction models are independent from each other. On contrary, joints five and six are coupled and their friction models are dependent with each other.

The balancing force is compensating the gravitational forces acting on the robot links. Without a balancing force, torque values that are supposed to be generated by the joint actuators can exceed the limits due to high weights of the links. This balancing force is dependent on the displacement of the second joint. This dynamic effect is caused by the spring mechanism that is integrated in the robot arm. This spring mechanism is located in the second link (the arm) and connected to second link from one end and the shoulder of the robot from the other end. A nonlinear mathematical model of the spring mechanism is used in this study. This spring model with identified parameters are taken from a previous study of Akbaş.

2.3.1.4 End-effector and F/T transducer dynamics

The vector τ_e shown in equation (2.45) is caused by the external interaction wrench exerted by the end-effector. It can be also expressed in terms of forces acting on the system τ_c as in equation (2.68).

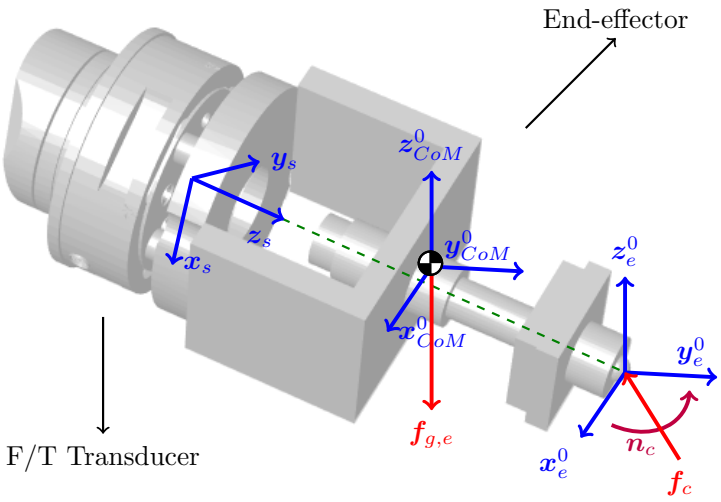

Figure 2.9 : Coordinate placement of sensor and end-effector tip/CoM.

$$\tau_c(q, \mathbf{f}_c) = -\tau_e(q, \mathbf{f}_e) \tag{2.68}$$

Modeling the effects of the wrench acting on the end-effector allows to compensate the extra torques applied on the joints. The relationship between end-effector wrench $\mathbf{f}_c$ and the torque vector τ_c is described in equation (2.69) in the same way as equation (2.49).

$$\tau_c(q, \mathbf{f}_c) = \mathbf{J}_e^T(q)\mathbf{f}_c \tag{2.69}$$

The wrench acting on the end-effector $\mathbf{f}_c$ is required for the computation of τ_c is obtained by the F/T transducer mounted at the wrist. However, the force and torque values it reads are not representing the pure external interaction wrench. The weight of the end-effector is included in the forces that F/T transducer reads. In order to obtain the pure external interaction wrench, the wrench that occurs from the end-effector weight $\mathbf{f}_{g,e}$ is required to be excluded from the forces and torques that are read through the sensor $\mathbf{f}_s$. The forces and torques acting on the end-effector and F/T transducer and the coordinate frames are shown in Figure 2.9. In this figure, while the sensor frame is attached on its reference frame, other two frames have the orientation of the base frame. It is important to note that the addition or subtraction of wrenches is valid if

they represented on the same coordinate frame, which is the coordinate frame of the sensor S in equation (2.70).

$$\mathbf{f}_c^{(S)} = \mathbf{f}_s^{(S)} - \mathbf{f}_{bias}^{(S)} - \mathbf{f}_{g,e}^{(S)} \tag{2.70}$$

In order to achieve this, the end-effector weight wrench $\mathbf{f}_{g,e}$ have to be computed. The method used for this computation is based on gathering raw sensor data on different end-effector and F/T transducer orientations and processing the data to compute $\mathbf{f}_{g,e}$. The raw sensor data gathering happens on three different orientations of sensor frame in a short period of time. The average of data for each orientations are taken separately to reduce the measurement noise. These computations are also used for eliminating the available sensor bias $\mathbf{f}_{bias}$. This computations in C-code are available in Appendix D.

As the result of these computations, the CoM of the end-effector and the total wrench from the end-effector weight $\mathbf{f}_{g,e}$ w.r.t. the CoM frame are obtained. In the next step, $\mathbf{f}_{g,e}$ transformed from base frame to the sensor frame with the spatial transformation matrix ${}^{s}\mathbf{X}_{(CoM)}$ as follows:

$$\mathbf{f}_{g,e}^{(S)} = {}^{S}\mathbf{X}_{CoM}\mathbf{f}_{g,e}^{(CoM)} \tag{2.71}$$

In the final step, $\mathbf{f}_c^{(S)}$ computed in equation (2.70) is transformed from the sensor frame to end-point of the end-effector w.r.t. base frame as follows:

$$\mathbf{f}_e = -\mathbf{f}_c = -{}^{e,0}\mathbf{X}_S\mathbf{f}_c^{(S)} \tag{2.72}$$

The computation of $\mathbf{f}_e$ is completed with these transformations and the compensation. It becomes available for using in the dynamic formulations and control schemes.

2.3.2 Forward dynamics

The forward dynamics computes the predicted motion of a robot as a response to acted actuation and contact forces/torques. It is useful for the simulations of the robots to analyze system and design the controller. The EoM in terms of the forward dynamics is derived from its inverse dynamics counterpart from equation (2.44) as follows:

$$\ddot{\boldsymbol{q}} = \mathbf{M}^{-1}(\boldsymbol{q})[\tau - \mathbf{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \mathbf{f}_e)] \tag{2.73}$$

The generalized acceleration $\ddot{\boldsymbol{q}}$ is computed as a result of the forward dynamics. The generalized velocity $\dot{\boldsymbol{q}}$ and position $\boldsymbol{q}$ is computed by the first and second integration of $\ddot{\boldsymbol{q}}$ over time.

The computation of equation (2.73) is straightforward by using Euler-Lagrange formulation due to its ability of finding the dynamic effects separately. The main difficulty of this computation is the inversion of the 6×6 inertia matrix $\mathbf{M}(\boldsymbol{q})$. In order to use the Newton-Euler formulation for computing the forward dynamics, one of the approaches is manipulating the inputs of the $NE(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, g, \mathbf{f}_e)$ from equation (2.67) iteratively to find the dynamic effects. $\mathbf{M}(\boldsymbol{q})$, $\mathbf{b}(\boldsymbol{q}, \dot{\boldsymbol{q}})$, $\mathbf{g}(\boldsymbol{q})$ and $\tau_e(\boldsymbol{q}, \mathbf{f}_e)$ terms are computed as follows:

$$m_{*,j} = NE(\mathbf{u}_j, \mathbf{0}, \mathbf{0}, 0, \mathbf{0}), \text{where} \quad \mathbf{M}(\boldsymbol{q}) = m_{i,j} \in \mathbb{R}^{n \times n} \tag{2.74}$$

where $\mathbf{u}_j$ is a $n \times 1$ unit vector with a 1 on its j^{th} element and 0 for the rest of the elements for $j = 1 \ldots n$.

$$\mathbf{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = NE(\boldsymbol{q}, \dot{\boldsymbol{q}}, \mathbf{0}, 0, \mathbf{0}) \tag{2.75}$$

$$\mathbf{g}(\boldsymbol{q}) = NE(\boldsymbol{q}, \mathbf{0}, \mathbf{0}, g, \mathbf{0}) \tag{2.76}$$

$$\tau_e(\boldsymbol{q}, \mathbf{f}_e) = NE(\boldsymbol{q}, \mathbf{0}, \mathbf{0}, 0, \mathbf{f}_e) \tag{2.77}$$

where $\mathbf{0}$ is $n \times 1$ vector of zeros. Instead of finding $\mathbf{b}(\boldsymbol{q}, \dot{\boldsymbol{q}})$, $\mathbf{g}(\boldsymbol{q})$ and $\tau_e(\boldsymbol{q}, \mathbf{f}_e)$ terms separately, the sum of them can be computed as below:

$$\mathbf{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \mathbf{f}_e)^* = \mathbf{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \mathbf{g}(\boldsymbol{q}) + \tau_e(\boldsymbol{q}, \mathbf{f}_e) = NE(\mathbf{0}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, g, \mathbf{f}_e) \tag{2.78}$$

where $\mathbf{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \mathbf{f}_e)^*$ is the $\mathbf{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \mathbf{f}_e)$ vector without τ_{frc} and τ_{spr}. Those two vectors are required to be implemented additionally.

3. CONTROL SYSTEM DESIGN

In this chapter, the trajectory generation and controller design are presented in two sections. Trajectory generation describes how a robot behaves in time. Different types of trajectories for motion are given in basic forms. A controller is used in the system of the robot to achieve the generated trajectories. Design of motion and force controllers are described separately at first and later integrated in the compliance control schemes. The relationship between trajectory generation, controller and a robot can be visualized simply as in the Figure 3.1. The difference between generated input and robot output are processed through controller schemes and fed into robot's actuators.

3.1 Trajectory Generation

Trajectories are used as reference inputs in robot control loops. In task planning, trajectory generation plays an important role. Generation of trajectories are formulated mathematically in consideration of physical limitations. Motion trajectories are the main references for robots. How a robot moves to a point is as important as where a robot goes. That means, velocity and acceleration profile of a robot motion is required to be planned to achieve a stable and smooth movement. The motion trajectory for a robot manipulator can be generated in joint-space or task-space. For basic motions, trajectories are classified as point-to-point, via points and periodic trajectories. For more complex motions, trajectories can be generated using parametric equations. A constant force reference is used in the majority cases in which tasks require force control; however more complex force trajectories can be also generated.

Point-to-point trajectories can be generated using polynomial or piece-wise position, velocity and acceleration profiles. Trajectories with multiple way-points can be achieved by sequencing point-to-point trajectories back-to-back or by giving via points in space to guide the trajectory and filling or interpolating the points between them. Linear interpolations with continuous acceleration blends are used as a method to

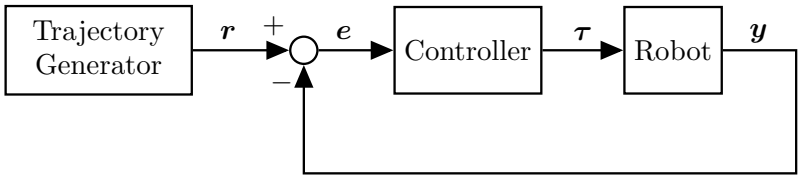

Figure 3.1 : Trajectory generator and controller.

achieve this. In this method, a trajectory is generated as via points connected with linear interpolations and then the connection points are replaced with blends with respect to the acceleration limits. A second method is to use cubic spline functions by placing cubic functions between via points in order to have a continuous trajectory. Trajectories using via points as opposed to sequenced point-to-points result in reduced finishing times and smooth continuous motions.

3.1.1 Joint space trajectories

A robot manipulator's joints can move separately or together as a result of the planned joint space trajectories. In case of a single joint trajectory, one joint follows the given trajectory while other joints stay in their initial positions. In case of a multiple joint trajectory, all moving joints need to be synchronized to achieve a single motion for the manipulator. To synchronize multiple trajectories, those trajectories have to be calculated according to the most time-consuming trajectory. There are multiple factors that limit a motion trajectory in terms of joint specifications. A range of joint limits the positions that a joint can reach. Velocity and acceleration limits for nominal and maximum speeds identified or provided by manufacturer determines the trajectories final time.

A joint space trajectory can be converted to a task space trajectory via forward kinematics. This can be useful for planning a trajectory while considering the workspace limitations.

3.1.2 Task space trajectories

In task space, the trajectory of a robot is defined by the position and orientation of the coordinate frame attached to its end-effector w.r.t. its base frame. While position trajectory is represented generally on the three axes of a Cartesian coordinate frame,

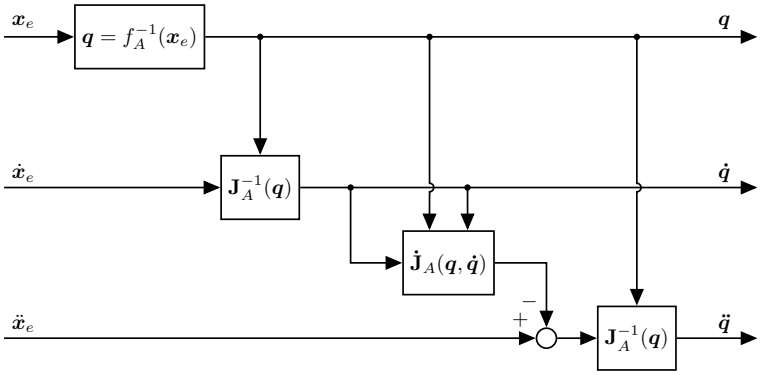

Figure 3.2 : Inverse kinematics in block diagram.

orientation trajectory is represented with one of the rotation representations like Euler angles, angle-axis etc. Position and orientation trajectories on each axis also have to be synchronized if the duration of both motions are requested to be the same. The duration of task space trajectories is also dependent on estimated task space velocity and acceleration limits. When planning a trajectory in task space, workspace limitations and singularity configurations have to be taken into consideration.

A task space trajectory can be generated with respect to the base frame, the inertial frame or the end-effector frame of the robot in order to satisfy the requirements of the given tasks. A trajectory generated with respect to a frame can be transformed by homogeneous and spatial transformation matrices. This operation is useful for some control approaches on specific tasks.

Transformation of task space trajectories into joint space trajectories is achieved through the inverse geometric and kinematic models. The implementation of this transformation is represented in a control scheme as shown in Figure 3.2. This transformation is useful for the observation of joint limits during the planning of a motion trajectory.

Generating a task space trajectory for the applied force is also possible. Rather than giving a constant desired force value, an altering force trajectory can be planned for the tasks that require force sensitivity.

3.2 Controller Design

Control system design for dynamical systems aims at obtaining desired behavior in presence of internal and external disturbances. Internal disturbances consist of imperfections in kinematic and dynamic modeling and time-varying parameters. Some of the external disturbances can be specified as contact forces and changes in heat and humidity. Any disturbance can cause an error in desired behavior. A compensation of errors can be ensured by designing a closed-loop control system. The output of the system can be measured directly from a sensor device or computed from another measured data. In robotics, motion and force variables are the quantities that are intended to be controlled generally. Motion control and force control of the robot can be implemented separately without intersecting; however, implementing both of these controllers together in a single control scheme requires a compliance between them. In this section, motion and force control are described briefly and two main compliance control methods are discussed and applied in the experiments.

One of the most common control approaches is the decentralized PID (proportional, integral, derivative) or its variants such as PD and PI for both motion and force control schemes. This approach can be implemented on each degree of freedom of control variables by tuning the gains that are modeled after a linear second order differential equation. The tuning process can be done via trial-and-error by following the tuning methods which can be also very time-consuming. The advantage of this approach is the simplicity of implementation and ability to work without any dynamic model; however, it shows poor accuracy for highly dynamic situations. In order to improve the performance of the control scheme, nonlinear control schemes with implemented models and adaptive techniques can be applied at a complexity and computational load costs. The gains for PID controllers are positive definite diagonal matrices named as proportional $\mathbf{K}_P$, integral $\mathbf{K}_I$ and derivative $\mathbf{K}_D$ gains respectively.

A typical PID control scheme can be seen in Figure 3.3. In this scheme, the error is multiplied by $\mathbf{K}_P$ gain directly and $\mathbf{K}_I$ gain with its integration over time, with $\mathbf{K}_D$ gain with its time derivative. The sum of those control signals are named as control output of a PID controller.

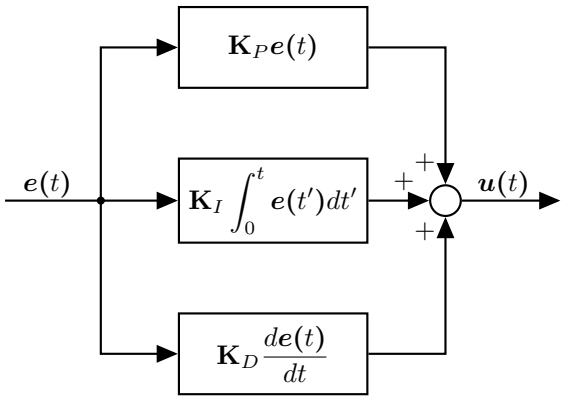

Figure 3.3 : PID controller.

For the tasks that require high dynamic performances, linearizing and decoupling control methods can be utilized. The nonlinearities caused by the dynamic effects of the robot can be canceled by using the computed torques via the inverse dynamic model. An implementation of the computed torque in a control scheme is presented in Figure 3.4. The estimation of the dynamic terms are denoted by *hat symbol (circumflex)*. The PID controller will encounter a more linear part of the system with the computed torque method. An accurate inverse dynamic model will make a better approximation of the system, which results in a further increase in the control performance. The payoff of this method is the complexity it brings to the control scheme and it can be impossible to implement on the systems lacking computational capability.

Adaptive methods for control schemes can be implemented to improve the control of the robot via adapting to uncertainties and system changes. Adaptation algorithm can be applied on PID controller gains and inverse dynamic model. An adaptation algorithm can be derived from dynamics of a robot or a model independent method like fuzzy logic.

The joint velocity and end-effector wrench feedback signals include high frequency noise due to derivation of discrete position values and analog measurement respectively. In order to reduce these noises from signals Kalman filtering is applied to these feedback signals of the control loops. A trade-off between noise reducing and keeping the characteristics of the signal happens with this filtering process. So that, signal noise cannot be eliminated completely and its effect can be seen on the control signal due to the derivative action of the PID controllers.

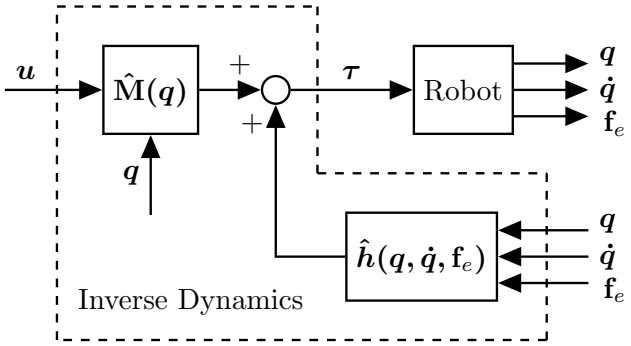

Figure 3.4 : Computed torque method.

3.2.1 Motion control

One of the primary control subjects of a robot manipulator is its motion. Even if to keep a robot in its current pose, it has to control its variable when the robot's motors are working. As mentioned in the previous sections, the motion of the robot can be described in joint space or task space. A robot manipulator includes sensing devices such as encoders and resolvers, which track the feedback joint positions or incremental changes in these positions. Also, a robot manipulator is the plant of the control system that accepts the system input in terms of joint torques. So that, it is simpler to implement a motion control scheme in joint space than task space. The block diagram of motion control in joint space is shown in Figure 3.5. In a joint space PID motion controller, $n \times n$ diagonal gain matrices apply to joint errors of an n-DoF robot. PID motion control law in the joint-space is given as follows:

$$\boldsymbol{\tau} = \mathbf{K}_P(\boldsymbol{q}^d - \boldsymbol{q}) + \mathbf{K}_D(\dot{\boldsymbol{q}}^d - \dot{\boldsymbol{q}}) + \mathbf{K}_I \int_0^t (\boldsymbol{q}^d - \boldsymbol{q})dt \tag{3.1}$$

A task-space motion control scheme requires forward and inverse kinematic transformations between joint-space and task-space. However, more complex tasks given to a robot are generally generated on task-space environments and it can be appropriate to use a task-space motion control in those situations. In a task-space PID motion controller, 6×6 diagonal gain matrices apply to errors on the 6-DoFs of the three-dimensional space.

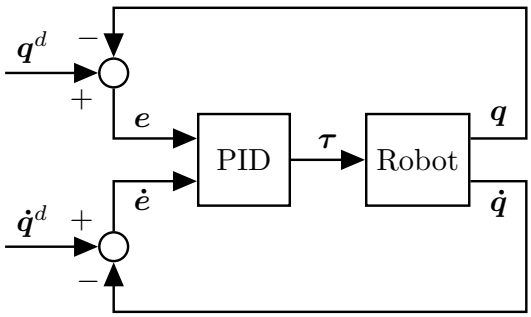

Figure 3.5 : Motion control in joint space.

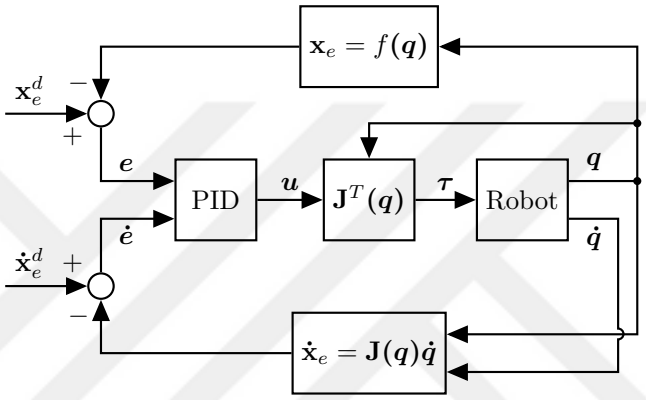

Figure 3.6 : Motion control in task space.

The block diagram of PID control of a task space motion in Figure 3.6 shows the additional kinematic transformations. At this time the errors are need to be in the task space, so that the joint space feedback transforms through the forward kinematics. As the PID controller in task space motion deals with the task space motion errors of the end-effector, the output vector of this controller is generated in terms of forces acting on the end-effector. The control output vector in terms of end-effector forces transforms through the transpose of the Jacobian matrix to joint torque vector as system input. PID motion control law in the task space is given as follows:

$$\boldsymbol{\tau} = \mathbf{J}^T(\boldsymbol{q})[\mathbf{K}_P(\mathbf{x}_e^d - \mathbf{x}_e) + \mathbf{K}_D(\dot{\mathbf{x}}_e^d - \dot{\mathbf{x}}_e) + \mathbf{K}_I \int_0^t (\mathbf{x}_e^d - \mathbf{x}_e)dt] \tag{3.2}$$

It is also possible to control a task space motion in a joint space control scheme. In order to achieve this, a task space trajectory is transformed through inverse kinematics. This leads to the rest of the process is same as the joint space controller with a joint space trajectory.

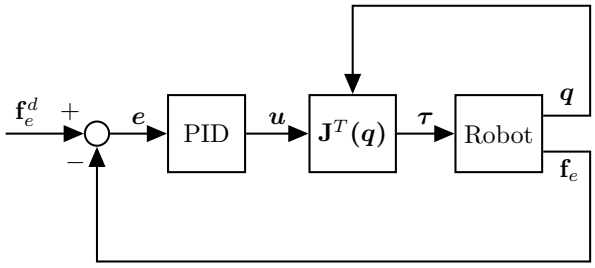

Figure 3.7 : Explicit force control scheme.

3.2.2 Force control

When a robot manipulator's end-effector interacts with its environment, contact forces and moments occur at that interaction location. This contact forces can be controlled by a force control scheme. Controlling the contact forces makes the robot possible to apply a desired force value to its end-effector contacts. This is crucial to keep a contact with the interacted object with avoiding a damage from excessively applied forces. It is very important for collaborating tasks, especially for the ones that involve human interactions. An explicit force control makes it possible to move and repositions the manipulator by hand on the applied axis. The controlled force data comes to the control scheme through F/T transducer mounted before the end-effector. The raw sensor data is required to be refined to contact wrench by removal of unwanted effects as described in the dynamics section. After that, the external wrench becomes available to be used in the control loop.

For the force control schemes, PID controller as shown in Figure 3.7 is also commonly used; however, the derivative effect is not implemented in case of high amplitude sensor noises. For this case, the negative value of the task-space velocity feedback of the end-effector can be used if it has less noise. Moreover, it is beneficial to implement a feedforward action to the force control scheme because of the need for a continuity of applied force. PID force control law is given as follows:

$$\boldsymbol{\tau} = \mathbf{J}^T(\boldsymbol{q})[\mathbf{K}_P(\mathbf{f}_e^d - \mathbf{f}_e) + \mathbf{K}_D(\dot{\mathbf{f}}_e^d - \dot{\mathbf{f}}_e) + \mathbf{K}_I \int_0^t (\mathbf{f}_e^d - \mathbf{f}_e)dt] \tag{3.3}$$

3.2.3 Compliance control

In some cases, robot manipulators are required to be controlled both for its motion and the wrench generated from their physical interactions. Those cases can be interactions between environment and collaboration with another robot or a human. Different priorities need to be considered for different types of interactions. For example, for a robot and a static environment interaction the precision of the motion can be more important than considering force inputs from different directions. For a robot and a human interaction, the force control can be more important than the motion control due to safety reasons. In order to accomplish different types of interactions of a robot, different types of compliance control methods have been developed. In this study, hybrid position/force control and parallel position/force control schemes are presented as compliance control schemes.

For the compliance control in task space, a compliance (task) frame is defined in order to control the each of the 6-DoF of the 3-D space. The diagonal elements of gain matrices are corresponding to the axes of a Cartesian coordinate frame that is selected as the compliance frame. The determination of the compliance frame is depending on the requirements of a given task. A task that involves an interaction between the end-effector of a robot and its environment necessitates an appropriate compliance frame to control desired axes.

3.2.3.1 Hybrid position/force control

The implementation of motion and force control schemes in a single control scheme with a separation of two subspaces is named as hybrid position/force control scheme. This control scheme consists of two complementary loops of motion control and force control in either joint space or task space. The hybrid position/force control in joint-space as shown in Figure 3.8 is proposed by Craig and Raibert and later corrected by Fisher. The task space implementation shown in Figure 3.9 used in this study is based on the design described in Khalil and Dombre [39].

In implementations of this control scheme, either motion or force control is applied on the defined compliance frame, which is decided by considering the interaction of the end-effector. In general, the compliance frame is selected on the end-effector frame in

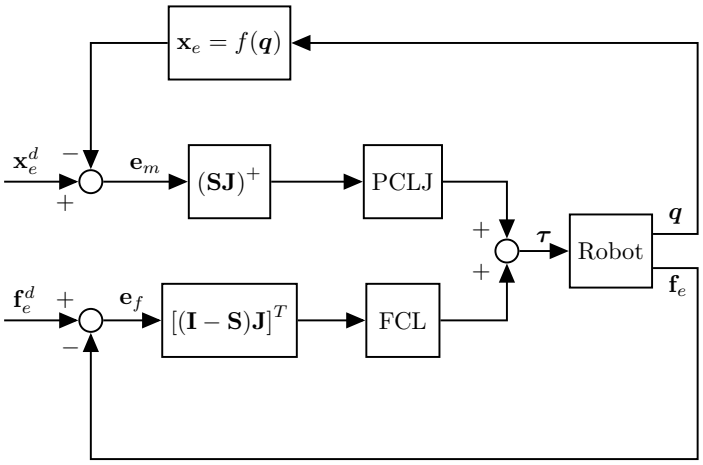

Figure 3.8 : Hybrid position/force control in joint space.

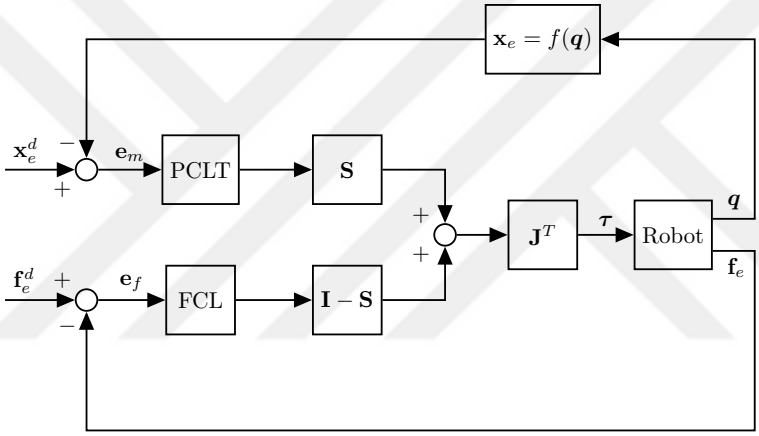

Figure 3.9 : Hybrid position/force control in task space.

order to control its each DoFs in the 3-D task space. The decision of which control scheme to be enabled is achieved by a 6×6 matrix denoted as the compliance selection matrix. The compliance selection matrix $\mathbf{S}$ is a square diagonal matrix that has diagonal elements for each degree of freedom. Each diagonal element acts as an on/off switch between the motion and the force control schemes for its corresponding axis by taking 1 or 0 values. It is implemented in the motion control part of the parallel loops. Its complementary part is implemented on the force control loop as $\mathbf{I}-\mathbf{S}$ matrix. As a result, when a diagonal element of $\mathbf{S}$ matrix has a value of 1, motion control is on for that DoF and the same element of the $\mathbf{I}-\mathbf{S}$ matrix becomes 0, which makes the force control is off on that DoF. The selected control outputs transform to joint torques to be combined later. The implementation of this parallel schemes as the hybrid position/force control in task space is shown in Figure 3.10.

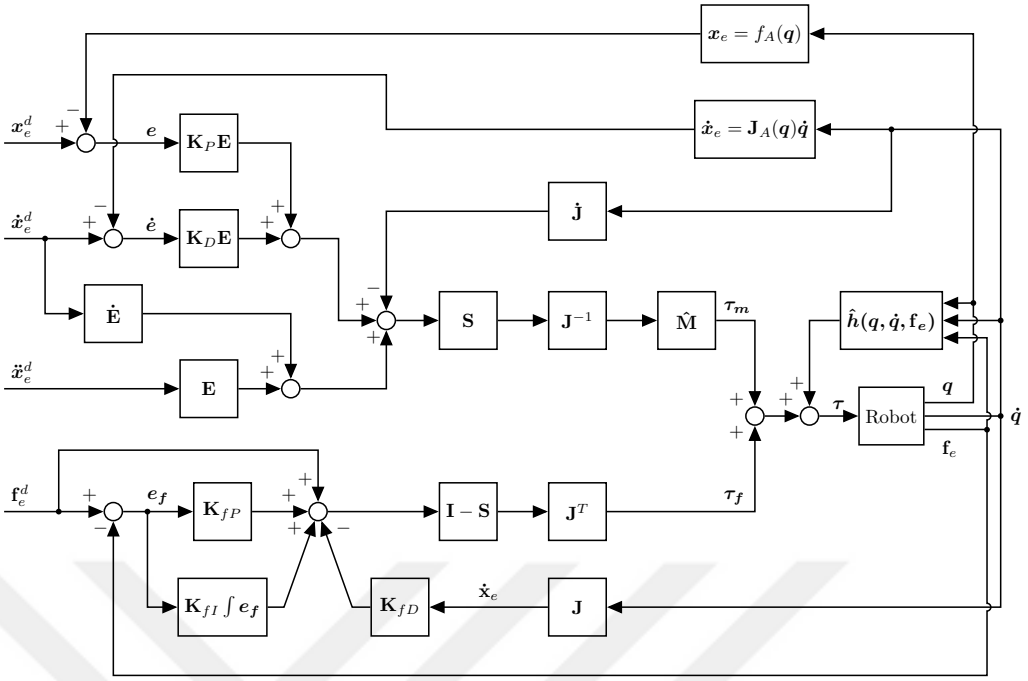

Figure 3.10 : Hybrid position/force control with computed torque.

The generalized control law for the hybrid position/force control scheme is given as follows:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_m + \boldsymbol{\tau}_f + \hat{\boldsymbol{h}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \mathbf{f}_e) \tag{3.4}$$

It is seen that the system input torque vector is composed of compensation torque vector $\hat{\boldsymbol{h}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \mathbf{f}_e)$ and torque vectors from motion control $\boldsymbol{\tau}_m$ and force control $\boldsymbol{\tau}_f$. The motion control part of this control is chosen as PD with computed torque scheme. The integral action is not implemented for simplicity and comparability reasons. The control law for motion control part is designed as follows:

$$\boldsymbol{\tau}_m = \hat{\mathbf{M}}\mathbf{J}^{-1}\mathbf{S}[(\mathbf{E}\ddot{x}_e^d + \dot{\mathbf{E}}\dot{x}_e^d) + \mathbf{K}_P\mathbf{E}(x_e^d - x_e) + \mathbf{K}_D\mathbf{E}(\dot{x}_e^d - \dot{x}_e) - \dot{\mathbf{J}}\dot{\boldsymbol{q}}] \tag{3.5}$$

The control law of the PID with a feedforward action for the force control part of the general scheme is given in equation (3.6). The end-effector feedback twist is used rather than the wrench in the following equation.

$$\boldsymbol{\tau}_f = \mathbf{J}^T(\mathbf{I}-\mathbf{S})[\mathbf{f}_e^d + \mathbf{K}_{fP}(\mathbf{f}_e^d - \mathbf{f}_e) + \mathbf{K}_{fI}\int(\mathbf{f}_e^d - \mathbf{f}_e)dt - \mathbf{K}_{fD}\dot{\mathbf{x}}] \tag{3.6}$$

3.2.3.2 Parallel position/force control

Another approach for achieving a compliance control of a robot is controlling the dynamic behavior of the system in terms of relationship between produced forces and its motion. The dynamic behavior of a system can be represented as a force output produced from its velocity input is called as mechanical impedance. Moreover, the inverse of this relationship is named as mechanical admittance. The mechanical impedance Z is described in the frequency domain as follows:

$$Z(s) = F(s)/\dot{X}(s) \tag{3.7}$$

In the impedance control scheme, the end-effector of the robot is desired to behave like a 2^{nd} order linear dynamics of a mass-spring-damper system shown as,

$$\frac{F(s)}{X(s)} = sZ(s) = \Lambda s^2 + Bs + K \tag{3.8}$$

The desired behavior of the close-loop system for this system is represented in time-domain by the following equation:

$$\mathbf{f} = \boldsymbol{\Lambda}(\ddot{\mathbf{x}}^d - \ddot{\mathbf{x}}) + \mathbf{B}(\dot{\mathbf{x}}^d - \dot{\mathbf{x}}) + \mathbf{K}(\mathbf{x}^d - \mathbf{x}) \tag{3.9}$$

In this equation $\boldsymbol{\Lambda}$, $\mathbf{B}$ and $\mathbf{K}$ are 6×6 positive definite diagonal gain matrices for mass, damper and spring respectively. This can be described as six virtual independent mass-damper-spring systems for each DoF of 3-D space. The natural frequency ω_n and the damping ratio ζ of a linear mass-spring-damper model are shown in terms of its physical parameters (mass m, spring k and damper b coefficients) as follows:

$$\omega_n = \sqrt{\frac{k}{m}}, \qquad \zeta = \frac{b}{2\sqrt{km}} \tag{3.10}$$

The dynamic behavior of the system can be described by relative magnitudes of ω_n and ζ with their constitutive parameters. An increase on the inertial coefficient while keeping other coefficients constant, results in the decrease of ω_n and ζ. That causes the system to respond slower to an input. An increase only on the spring coefficient results in an increase of ω_n and a decrease of ζ. As a consequence, a faster response of the

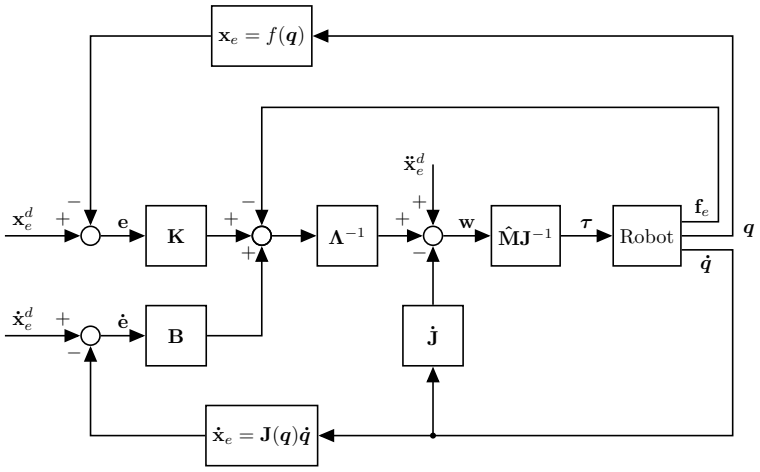

Figure 3.11 : Basic impedance controller.

system is generated. An increase only on the damping coefficient affects the transient response of the system by increasing the damping ratio. A smaller ω_n is desired for the direction of the contact for keeping the forces relatively low and a higher ω_n is desired for the directions of the motion for a good trajectory tracking.

A basic impedance control scheme is presented in Figure 3.11. The control law of the implemented impedance control scheme shown in Figure 3.12 is given as follows:

$$\boldsymbol{\tau} = \hat{\mathbf{M}}\mathbf{J}^{-1}\{[\mathbf{E}\ddot{\boldsymbol{x}}_e^d + \dot{\mathbf{E}}\dot{\boldsymbol{x}}_e^d] + \boldsymbol{\Lambda}^{-1}[\mathbf{BE}(\dot{\boldsymbol{x}}_e^d - \dot{\boldsymbol{x}}_e) + \mathbf{KE}(\boldsymbol{x}_e^d - \boldsymbol{x}_e) - \mathbf{f}_e] - \dot{\mathbf{J}}\dot{\boldsymbol{q}}\} + \hat{\boldsymbol{h}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \mathbf{f}_e) \quad (3.11)$$

Due to the nature of the impedance control scheme, the contact forces are controlled indirectly depending on the motion of the end-effector. An extended version of the impedance control scheme with explicit force control named as parallel position/force control is purposed by Chiaverini and Sciavicco and implemented in this study as in Figure 3.13. In this control scheme, the contact wrench is inserted to a force control-loop instead of feeding it directly into the closed-loop. Thus, the desired force reference can be achieved in addition to the controlling of the general dynamic behavior of the system. The desired behavior of the system after the addition of an explicit force control is described as follows:

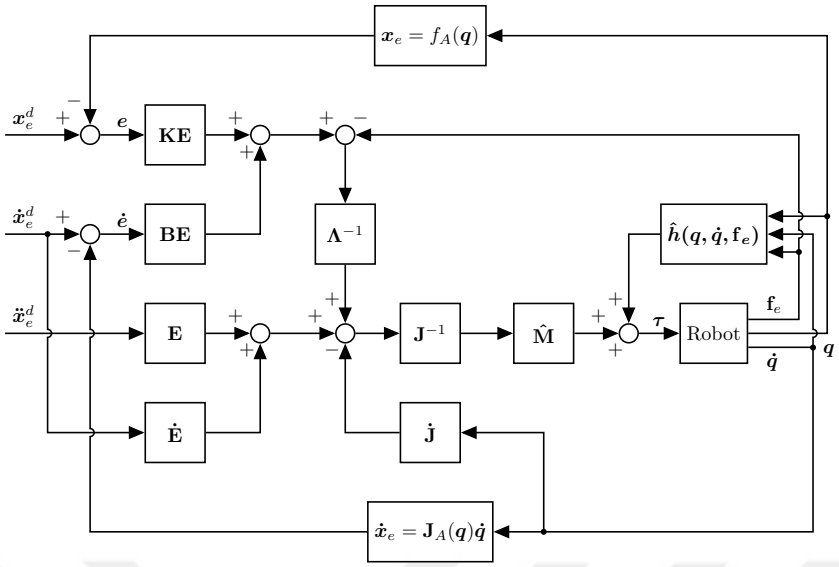

Figure 3.12 : Implemented impedance controller scheme.

$$\boldsymbol{\Lambda}(\ddot{\mathbf{x}}^d - \ddot{\mathbf{x}}) + \mathbf{B}(\dot{\mathbf{x}}^d - \dot{\mathbf{x}}) + \mathbf{K}(\mathbf{x}^d - \mathbf{x}) + \mathbf{K}_{fP}(\mathbf{f}^d - \mathbf{f}) + \mathbf{K}_{fI}\int(\mathbf{f}^d - \mathbf{f})dt - \mathbf{K}_{fD}\dot{\mathbf{x}} + \mathbf{f}^d = 0 \quad (3.12)$$

In this parallel position/force control scheme, motion and force control variables can be shown separately as in the hybrid position/force control scheme. The difference between these two control schemes is that, the separate control variables in terms of task-space accelerations are selected by **S** in hybrid position/force control scheme and combined in parallel position/force control scheme about each DoF of task-space. The control signals for motion $\mathbf{w}_x(t)$ and $\mathbf{w}_f(t)$ for parallel position/force control scheme are given in the following equations.

$$\mathbf{w}_x(t) = [\mathbf{E}\ddot{x}_e^d + \dot{\mathbf{E}}\dot{x}_e^d] + \boldsymbol{\Lambda}^{-1}[\mathbf{BE}(\dot{x}_e^d - \dot{x}_e) + \mathbf{KE}(x_e^d - x_e)] \quad (3.13)$$

$$\mathbf{w}_f(t) = \boldsymbol{\Lambda}^{-1}[\mathbf{f}_e^d + \mathbf{K}_{fP}(\mathbf{f}_e^d - \mathbf{f}_e) + \mathbf{K}_{fI}\int(\mathbf{f}_e^d - \mathbf{f}_e)dt + \mathbf{K}_{fD}\dot{\mathbf{x}}_e] \quad (3.14)$$

The motion control part remains the same as the basic impedance control scheme. The contribution of the force signal becomes a control output of a PID controller with a feed-forward action. This control scheme becomes identical to the basic impedance control scheme when there is no force reference and proportional gain is 1 and other gains are zero.

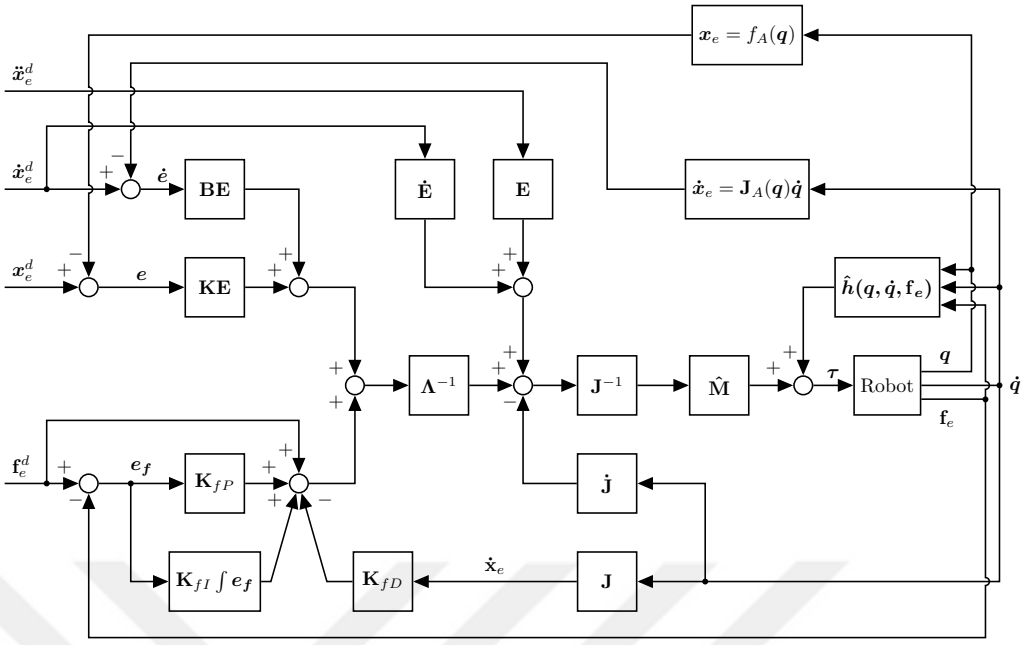

Figure 3.13 : Parallel position/force control scheme.

The total controller output in terms of end-effector acceleration is described as the sum of two controllers as follows:

$$\mathbf{w}(t) = \mathbf{w}_x(t) + \mathbf{w}_f(t) \tag{3.15}$$

The general output of this control scheme is dominated by the control loop having higher gains. For the contact direction, force control loop is desired to dominate the system in order to achieve the reference contact forces even on a variable motion. For other directions, motion control loop is desired to dominate the system for a good trajectory tracking. The domination of the force control over the motion control is achieved mostly by the integral action founded on the force control loop.

Due to the fact that control outputs appear in terms of task-space acceleration; an inverse dynamic model is required to be used in the impedance control schemes that include mass dynamics. The general control law of the impedance control scheme with explicit force control loop is shown in the following equation.

$$\boldsymbol{\tau} = \hat{\mathbf{M}}\mathbf{J}^{-1}[\mathbf{w}(t) - \dot{\mathbf{J}}\dot{\boldsymbol{q}}] + \hat{\boldsymbol{h}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \mathbf{f}_e) \tag{3.16}$$

4. EXPERIMENTATION

The two compliance control schemes presented in chapter 3 were implemented for the control of a single robot and collaborating robots. Firstly, hardware of the system was described in the experimental setup section. After that, the planned tasks of these two scenarios were explained in phases. At last, performances of the implementations of the *hybrid position/force control* and the *parallel position/force control* schemes were examined. Furthermore, the parallel position/force control scheme was tested with different control parameters in order to analyze resulting behaviors.

4.1 Experimental Setup

The experimental setup consists of mainly by two Stäubli RX160 series robots located side-by-side and a work-piece placed in between them as shown in Figure 4.1. The robot manipulators were equipped with F/T transducers on their wrists, and the one with the compliance control applied had the end-effector described in the system description section. Controllers of the robots have a sampling frequency of 250 Hz. Trajectories were planned using the robot application command terminal via a computer connected to both robot controllers. After the run command was given, both robots performed the tasks autonomously.

The work-piece used in this experimentation was a rigid body with a rectangular planar surface (60 $\times$ 60 cm) and a thickness of 5 cm. One of the large surfaces was facing towards the RX160 model robot manipulator executing the active compliance control by contacting the surface with its end-effector. The opposite surface was backed by the RX160L model robot manipulator via holding soft materials in-between the work-piece and its gripper.

Figure 4.1 : Location of robots and work-piece.

4.2 Task Description

The same task is planned to be completed by the robot controlled with an active compliance. The general task including both robot manipulators is separated in two parts by the task of the robot with a passive compliance control. Before describing the differences of those two tasks, the similarities of those tasks can be described in a general task.

The general task can be described in three phases as follows:

Phase 1: The end-effector of the robot controlled with the active compliance starts its motion from an arbitrary distance from the work-piece. It moves freely in constant velocity in the direction towards the work-piece until the contact is established. This motion for each cases is controlled by using motion control equivalents of the implemented control schemes.

Phase 2: Once the physical contact is established between the surface of the work-piece and the spherical object located on tip of the end-effector and the contact force in the moving direction exceeds a predefined magnitude, the switch flag becomes active on the trajectory generation and the control scheme.

Phase 3: In this phase, it is expected for the robot to complete a task of tracking a trajectory while keeping a desired contact force. After the switch flag activated, the reference motion trajectory at the tip point of the end-effector changes to a perfect

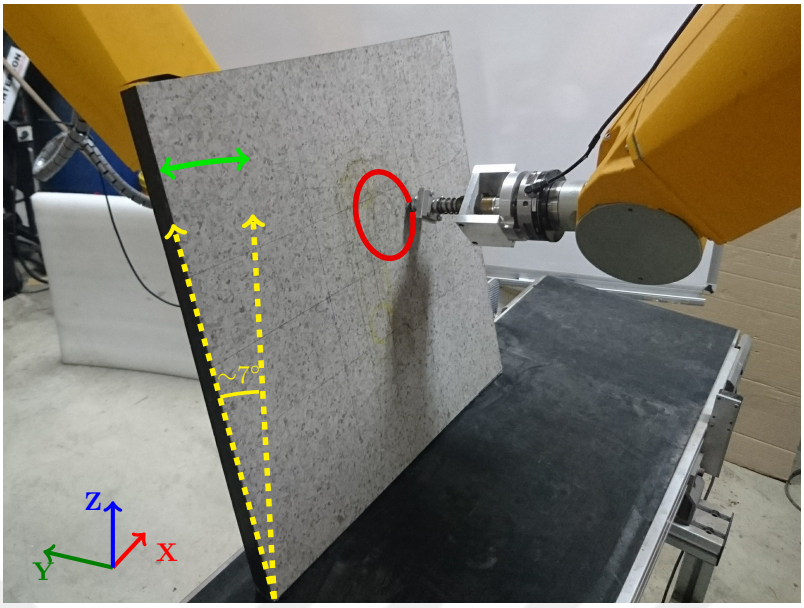

Figure 4.2 : Experimental setup.

circle path with a 5 cm radius. The circular trajectory is desired to have an angular 5^{th} order polynomial interpolation with a maximum tangential velocity of 0.02 m/s. Magnitude of the contact force controlled by the compliance control schemes is desired to be 30 N which remains in the linear range of the F/T transducer.

End-effector motion of the compliance controlled robot and the work-piece in phase 3 is described with colored drawings in the Figure 4.2. The compliance frame of actively compliance-controlled robot is chosen as a frame with origin on end-effector tip point with respect to its base frame for both control schemes. The initial free movement and active force control direction occured over y-axis and circular trajectory is generated on a xz-plane of the compliance frame.

4.3 Implementations

The tasks differ in two by the phase 3 with the motion of the robot with a passive compliance control. In the first task, named as single robot control, the passively compliance-controlled robot does not move, so that the work-piece remains in a fixed state. In the second task, named as collaboration control, the passively compliance-controlled robot moves in a previously planned periodic trajectory in the direction of the other robots contact force control. That causes the work-piece moving in a periodic motion.

Table 4.1 : PID gains in hybrid position/force control scheme.

Gain	x	y	z	α	β	γ
K_P	5500	3000	5750	4700	4500	5000
K_D	24	20	26	16	16	17.5
K_{fP}	14	14	14	14	14	14
K_{fD}	6.2	6.2	6.2	6.2	6.2	6.2
K_{fI}	20.2	20.2	20.2	20.2	20.2	20.2

Table 4.2 : Controller parameters in parallel position/force control scheme.

Gain	x	y	z	α	β	γ
Λ	1	7	1	1	1	1
K	5500	1000	5750	4700	4500	5000
B	24	20	26	16	16	17.5
K_{fP}	0.2	9	0.2	0.2	0.2	0.2
K_{fD}	0	4	0	0	0	0
K_{fI}	0	13	0	0	0	0

Each of these single robot control and collaboration control tasks are repeated in two cases. The first case is the implementation of the hybrid position/force control and parallel position/force control schemes with a single robot control. The second case is the implementation of the same two compliance control schemes with two robots collaborating.

The control parameters of the implemented hybrid position/force control and parallel position/force control schemes are given in Table 4.1 and 4.2 respectively. The rows of the tables are indicating the diagonal terms of control gains matrices and the columns are indicating the six independent axes of the task-space. The gray colored terms in the Table 4.1 indicate the inactive gains of the hybrid position/force control scheme. This selection happens by the compliance selection matrix S in between motion and force control as mentioned in the previous chapter.

4.3.1 Task 1: Compliance control of a single robot

The first task is described as a single robot interacting with a fixed work-piece in compliance control. The RX160L model robot is stationary in all phases of this task. The compliance control is achieved by the RX160 model robot by keeping the contact force in desired magnitude. The work-piece is leaned on the stationary robot with an inclination in order to establish disturbance on the contact forces while end-effector is tracking on a sloped surface. For this task, two types of compliance control schemes

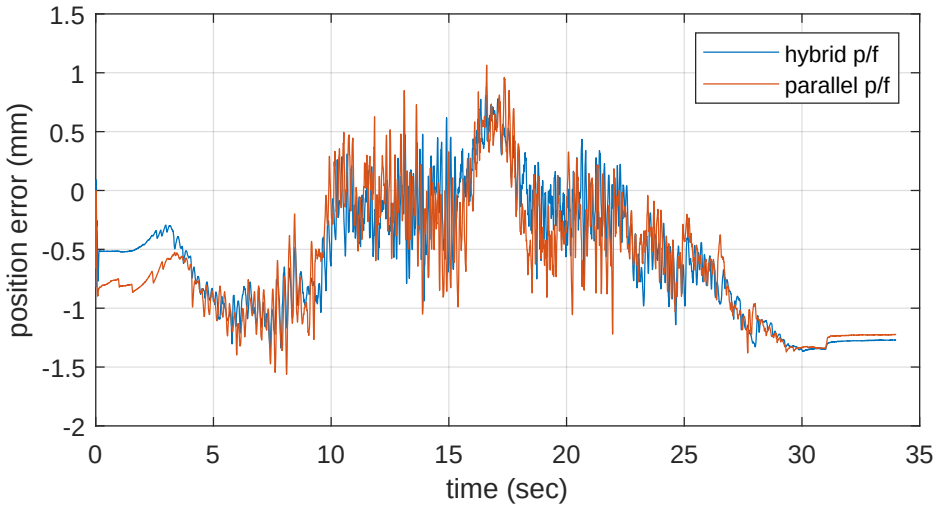

Figure 4.3 : Position tracking error of the end-point on the work-piece (O-xz) plane.

implemented and the results are given in the following cases. The experimental results of first task with hybrid position/force control and parallel position/force control schemes are shown in Figures 4.3 to 4.6.

4.3.1.1 Case 1: Hybrid position/force control

The results of the implementation of hybrid position/force control for the first task are displayed in the stated figures. The position tracking error of the tip of the end-effector over the reference circular path in a xz-plane with respect to base frame is shown in Figure 4.3. The maximum and mean absolute position tracking errors are computed as 1.4 mm and 0.65 mm respectively.

The force tracking performance along the control direction is given in Figure 4.4. The maximum and mean absolute force tracking errors are computed as 8.63 N and 1.12 N respectively.

Mean absolute position and force tracking errors and their standard deviations of both cases are given in Figure 4.7 for the first task. For the case 1, standard deviations of absolute position and force errors are 0.417 mm and 1.272 N respectively.

The translational motion of the end-point along the force controlled direction is given in Figure 4.5. The maximum displacement is 15.35 mm and the orientation error of the end-effector during the physical interaction in Euler angles is shown in Figure 4.6.

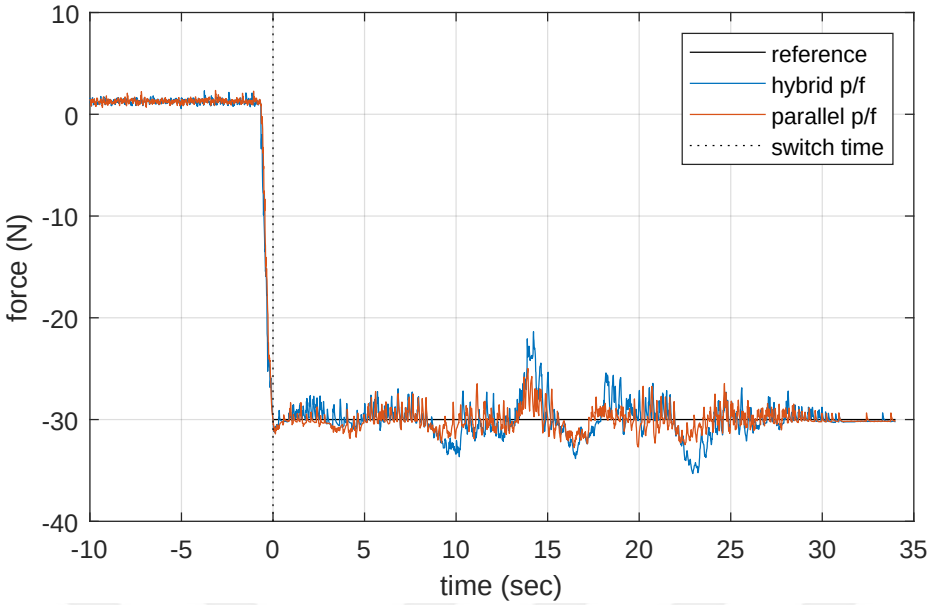

Figure 4.4 : Force control along the (O-y) axis to the work-piece plane.

4.3.1.2 Case 2: Parallel position/force control

The result of the implementation of parallel position/force control for the first task are displayed in the stated figures. The position tracking error of the end-point over the same reference circular path is given in Figure 4.3. The maximum and mean absolute position tracking errors are computed as 1.6 mm and 0.69 mm respectively.

Figure 4.4 shows the force tracking performance along the control direction. The maximum and mean absolute force tracking errors are computed as 5.02 N and 0.72 N respectively.

Mean absolute position and force tracking errors and their standard deviations of both cases are given in Figure 4.7 for the first task. For the case 2, standard deviations of absolute position and force errors are 0.404 mm and 0.681 N respectively.

The translational motion of the end-point along the force controlled direction is shown in Figure 4.5 and the maximum displacement is computed as 15.60 mm. The orientation error of the end-effector during the physical interaction is shown in Figure 4.6.

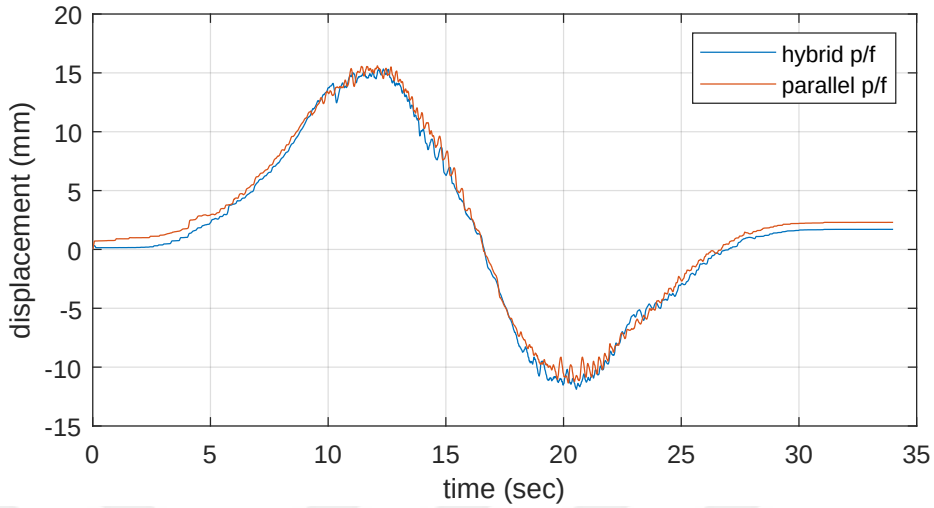

Figure 4.5 : Translational motion of end-effector along the (O-y) axis.

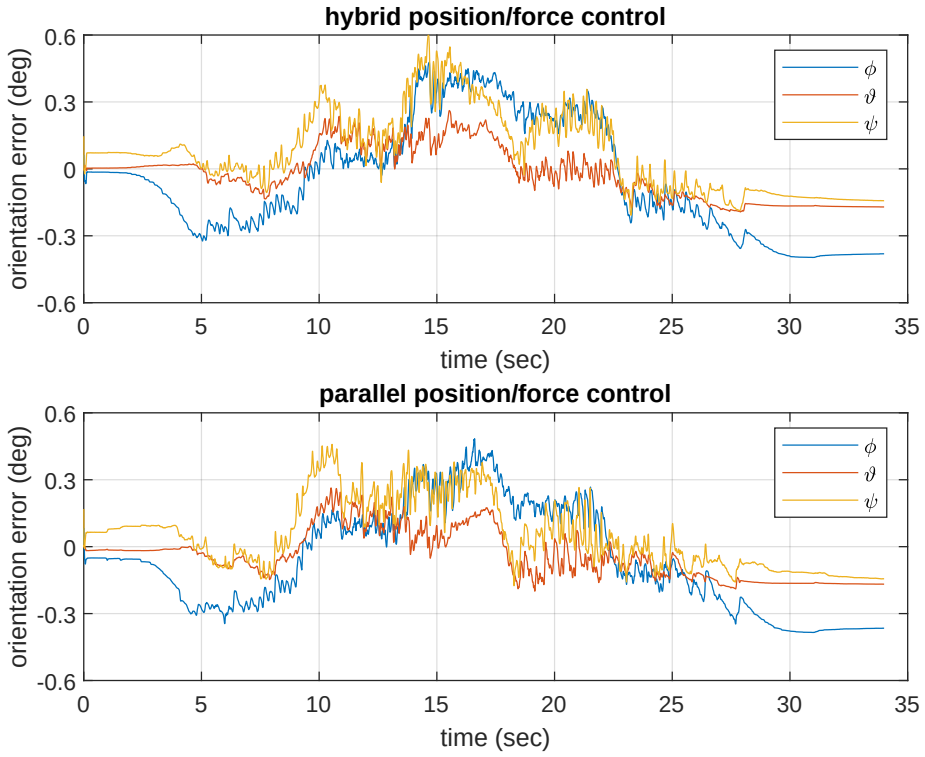

Figure 4.6 : Orientation error of the end-effector in terms of the Euler angles.

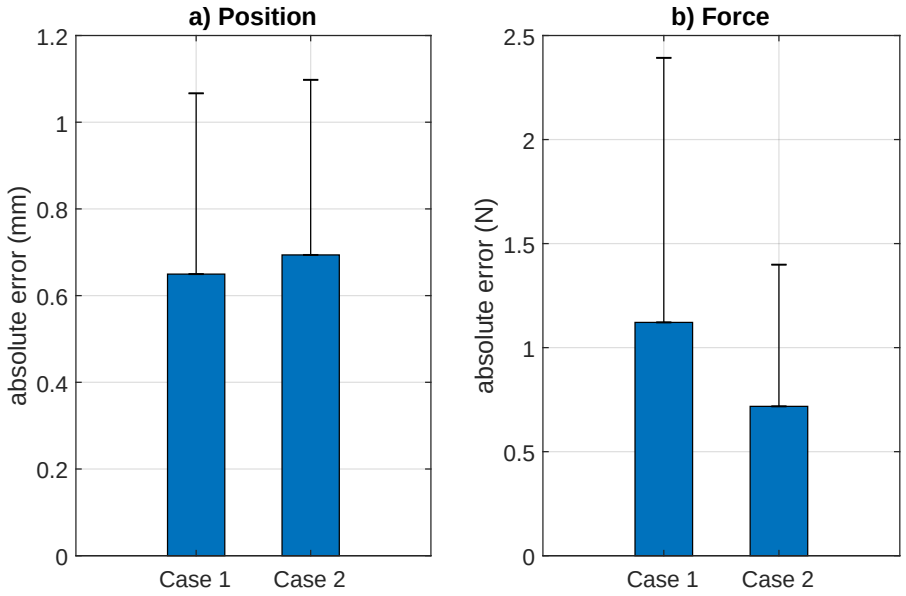

Figure 4.7 : Absolute mean error with standard deviations of both cases for task 1.

4.3.1.3 Discussion of the task 1

The position error over time plots of both controllers from Figure 4.3 appear similar in general. Both position tracking errors start and end around maximum negative values, however hybrid position/force controller starts with less errors. The steady-state errors are expected because there is no integrator action in the position control schemes. When end-effector moves at the maximum and minimum distances on the z-axis, position tracking errors fluctuate around zero. While end-effector is at the half-way though the circular path, the position tracking error values become maximum positive. Figure 4.4 indicates that force tracking errors of both controllers increases with the velocity of the end-effector in the same axis. Velocity of the end-effector along the y-axis of compliance frame can be interpreted from the change in displacements at small instances from Figure 4.5. No significant differences in orientation errors in terms of XYZ Euler angles for both controller are seen from Figure 4.6. Steady-state errors for orientation error are observed because of the absence of the integrator action in motion control parts of both compliance control schemes. The orientation error is smaller in general at the angle ϑ for both controllers due to minimal moments on the y-axis corresponding to rotation in ϑ. The spherical moving tip of the end-effector reduces the moment produced from contact motion on that axis.

4.3.2 Task 2: Compliance control of collaborating robots

The second task is described as collaboration control with a moving piece. The RX160L model robot is stationary in the first two phases and periodically moving in the third phase. The compliance control is achieved actively by the RX160 model robot by controlling the contact force while moving harmoniously with the passively compliance-controlled robot. The purely motion controlled periodic motion of the RX160L robot is a sinusoidal motion with a period of 12 seconds for two repeats. The periodic angular motion of the work-piece about x-axis in orientation of base frame caused by this motion is given as 3.5 deg for amplitude and $\pi/6$ rad/s for frequency. The contact force is continuously changing due to the work-piece motion and the changing surface slope in this task. The results of implementation of two compliance control schemes to achieve this task is are discussed in the following cases. The experimental results of the first task with hybrid position/force control and parallel position/force control are shown in Figures 4.8 to 4.11.

This task is repeated with different control parameters resulting in various desired natural frequencies and damping ratios for the parallel position/force control scheme. The experimental results of the different natural frequencies and damping ratios are shown in Figures 4.13 to 4.15.

4.3.2.1 Case 1: Hybrid position/force control

The results of the implementation of hybrid position/force control for the second task can be found in the stated figures. The position tracking error of the tip of the end-effector over the reference circular path in a xz-plane with respect to base frame is shown in Figure 4.8. The maximum and mean absolute position tracking errors are computed as 1.31 mm and 0.46 mm respectively.

The force tracking performance along the control direction is given in Figure 4.9. The maximum and mean absolute force tracking errors are computed as 6.95 N and 2.01 N respectively.

Mean absolute position and force tracking errors and their standard deviations of both cases are given in Figure 4.12 for the second task. For the case 1, standard deviations of absolute position and force errors are 0.336 mm and 1.535 N respectively.

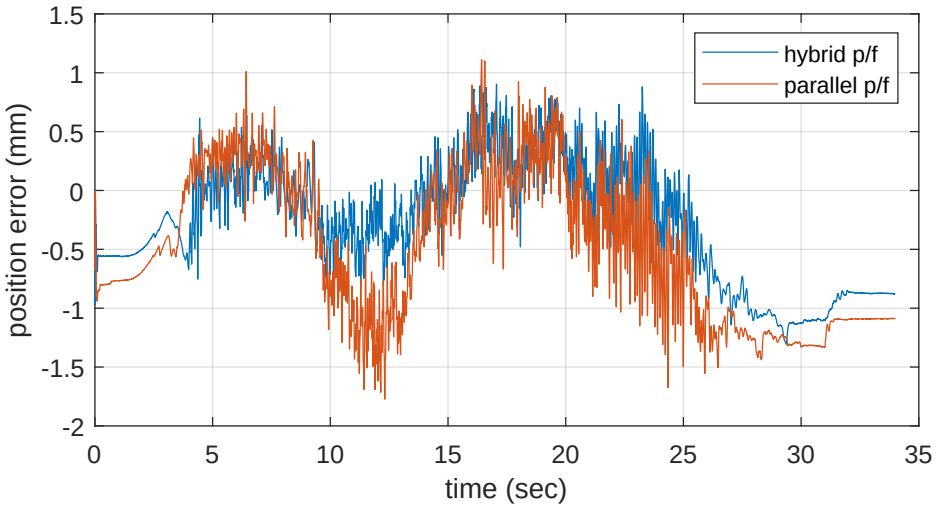

Figure 4.8 : Position tracking error of the end-point on the work-piece (O-xz) plane.

The translational motion of the end-point along the force controlled direction is given in Figure 4.10. The maximum displacement is 62.08 mm and the orientation error of the end-effector during the physical interaction in Euler angles is shown in Figure 4.11.

4.3.2.2 Case 2: Parallel position/force control

The result of the implementation of parallel position/force control for the second task can be found in the stated figures. The position tracking error of the end-point over the same reference circular path is given in Figure 4.8. The maximum and mean absolute position tracking errors are computed as 1.77 mm and 0.6 mm respectively.

Figure 4.9 shows the force tracking performance along the control direction. The maximum and mean absolute force tracking errors are computed as 5.94 N and 1.29 N respectively.

Mean absolute position and force tracking errors and their standard deviations of both cases are given in Figure 4.12 for the second task. For the case 2, standard deviations of absolute position and force errors are 0.433 mm and 1.115 N respectively.

The translational motion of the end-point along the force controlled direction is shown in Figure 4.10 and the maximum displacement is computed as 60.95 mm. The orientation error of the end-effector during the physical interaction is shown in Figure 4.11.

The same task for second case is repeated with alternative impedance references in order to evaluate the parallel position/force controller. Two more experiments with

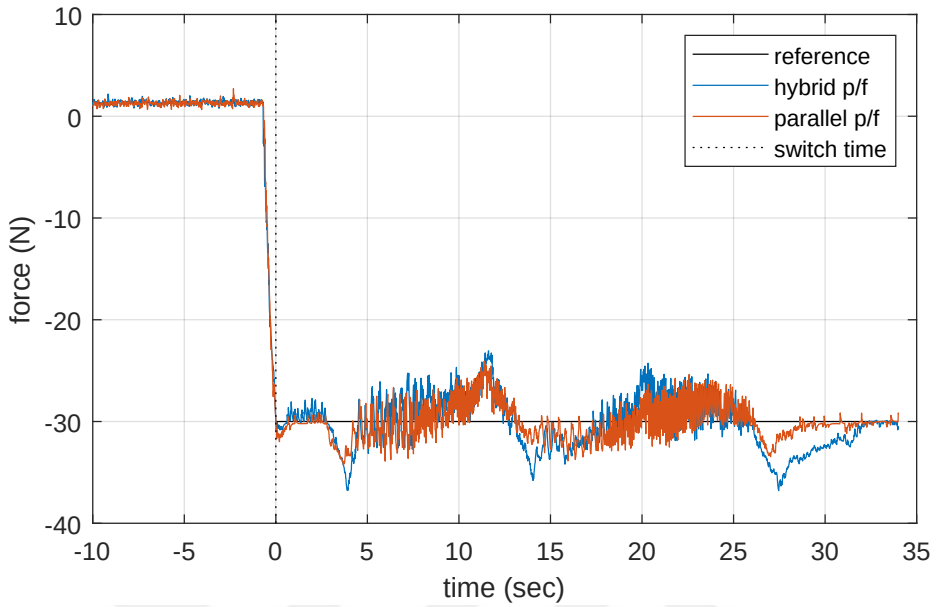

Figure 4.9 : Force control along the (O-y) axis to the work-piece plane.

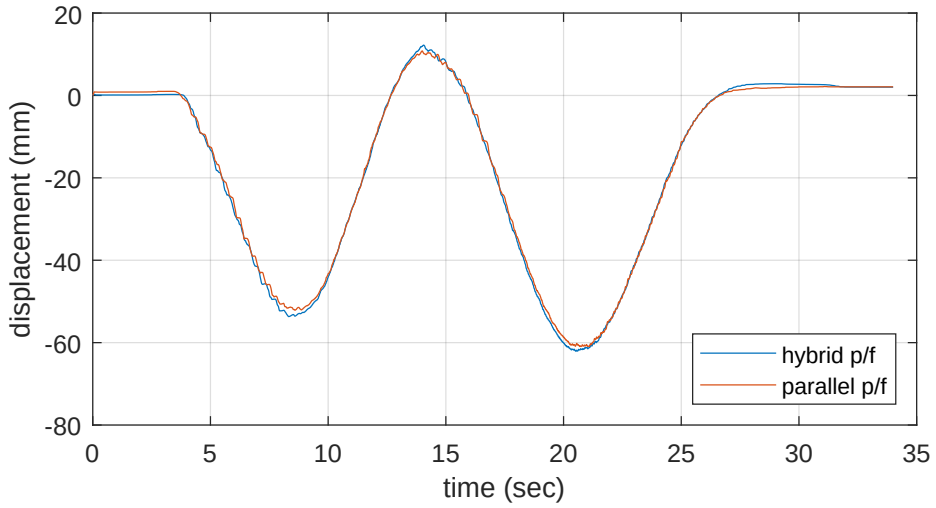

Figure 4.10 : Translational motion of end-effector along the (O-y) axis.

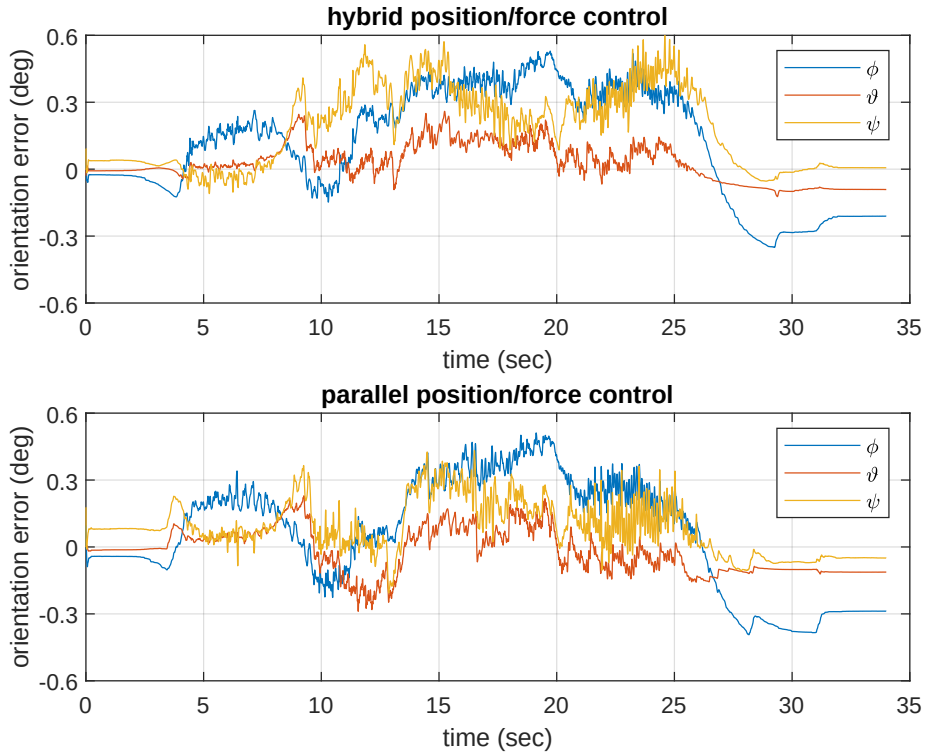

Figure 4.11 : Orientation error of the end-effector in terms of the Euler angles.

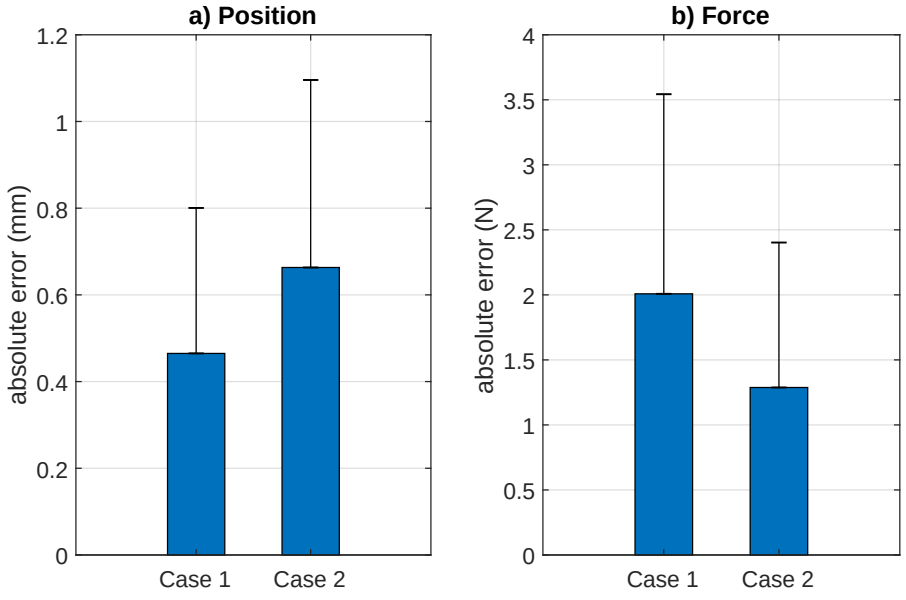

Figure 4.12 : Absolute mean error with standard deviations of both cases for task 2.

different control parameters are conducted in addition to the previous case with the parallel position/force controller. These three controllers with different impedance references indicate the different levels of explicit force control effect on an impedance control scheme.

Table 4.3 : Control parameters for different impedances.

Z#	Λ	K	B	K_{fP}	K_{fI}	K_{fD}	ω_n	ζ
1	7	1000	20	9	13	4	11.95	0.12
2	7	10000	20	4.5	6.5	0	37.8	0.04
3	2.5	1500	15	1	0	0	24.5	0.12

The first controller for impedance (Z_1) is the one used in the parallel position/force controller. This one is used for comparing with hybrid position/force controller shown in the previous case. This controller prioritizes the force control over impedance control. The second controller for impedance (Z_2) has a balanced force and impedance control with reduced force control parameters and an increased stiffness parameter in the direction of contact in the compliance frame. Control parameters of the last controller for impedance (Z_3) were selected to act as a basic impedance controller. The force reference is set as zero together with the derivative and integral force control gains. Proportional force control gain is set as one in order to feed force feedback value directly to the system as in basic impedance control schemes. The parameters of these three controllers are presented in Table 4.3. Their corresponding natural frequencies and damping ratios shown in the right columns of the table are calculated using mass-spring-damper system parameters.

Table 4.4 : Performances of the parallel position/force control in terms of position and force tracking errors.

Z#	e_{max} (mm)	e_{mean} (mm)	$e_{f_{max}}$ (N)	$e_{f_{mean}}$ (N)
1	1.77	0.66	5.94	1.29
2	1.67	0.65	28.08	9.65
3	1.82	0.91	48.38	18.78

The position tracking errors and the force control performances of the parallel position/force control scheme with alternative parameters sets are shown in Figures 4.13 and 4.14 respectively. Calculated maximum position error and mean absolute position tracking errors are denoted by e_{max} and e_{mean} respectively in Table 4.4. Similarly, calculated maximum force tracking error and mean absolute force tracking

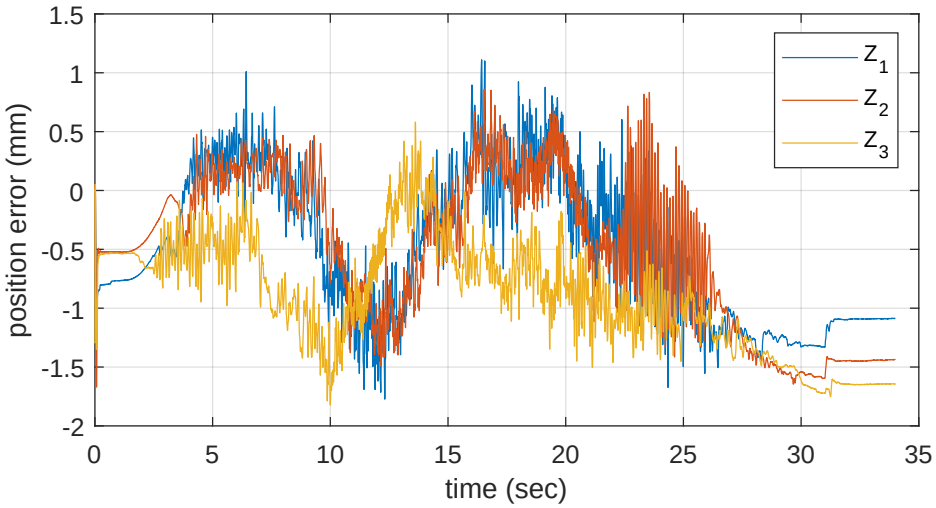

Figure 4.13 : Position tracking error of the end-point on the work-piece (O-xz) plane, with different impedances.

errors are denoted by $e_{f_{max}}$ and $e_{f_{mean}}$ respectively in the same table. The maximum displacement along the y-axis of the compliance frame shown in Figure 4.15 are calculated as 60.95 mm, 54.62 mm and 43.57 mm respectively.

4.3.2.3 Discussion of the task 2

In the collaboration task, the position tracking error over time plots of both controllers from Figure 4.8 also appear similar in form roughly. However, hybrid position/force controller resulted with smaller negative position tracking errors, while positive errors are reaching closer limits. When looking at the changes of position tracking errors in time, it is observed that errors with highest magnitude occur at the force unloading situations. On the contrary, force tracking in Figure 4.9 are closer to the reference values for parallel position/force controller in the same situations. Considering motion along the (O-y) axis in Figure 4.15 together with the force tracking plots, deviations from reference happen when the end-effector of the robot is moving with higher velocities. Due to integral action of the integrated PID force controller exists on both controllers, there are no significant steady-state force tracking errors among each other. Figure 4.11 is showing the orientation tracking errors of the robot's end-effector in terms of XYZ Euler angles over time, it is indicated that there is no remarkable differences for both controllers.

The results of the experiments with different impedances for the parallel position/force controller are discussed in this paragraph. The position tracking errors for the radius of

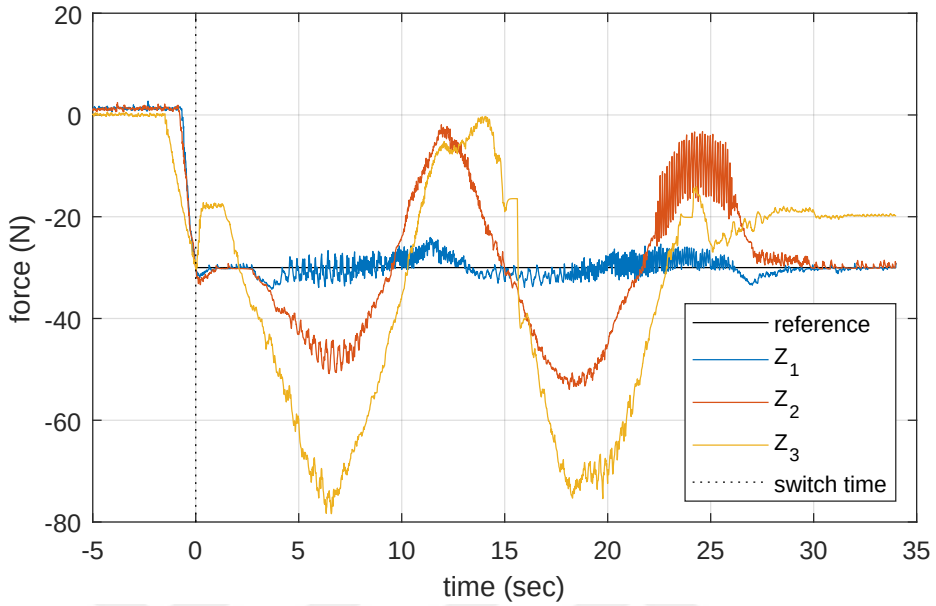

Figure 4.14 : Force control along the normal (O-y) axis to the work-piece plane, with different impedances.

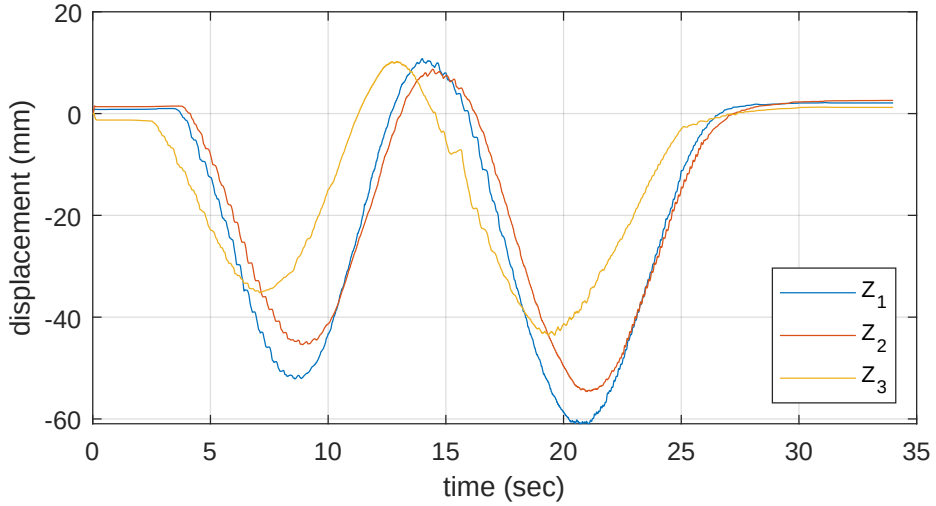

Figure 4.15 : Translational motion of end-effector along the normal (O-y) axis, with different impedances.

the circular trajectory about controllers with three different impedances are shown in Figure 4.13. As shown in the figure, it is observed that the first two controllers perform similarly; however, the second controller produced an oscillation at the final stage. This can be a result of the increase of the natural frequency and the reduced damping ratio, which can be seen in Table 4.3. The third controller has a smaller position tracking error, which is caused by the lack of an explicit force control. An explicit force control on an axis can generate disturbances to other axes that are motion controlled. The force control performances over time for the three controllers with different impedances are shown in Figure 4.14. The first controller shows a dominant explicit force control with an impedance control to a smaller extent. On the contrary, third controller does not have an explicit force control, which is apparent in the plot when the force control is observed. This is especially clear when the force control plot is evaluated with the displacement of the end-effector on the force controlled axis over time in Figure 4.15. When the displacement of the tip of the end-effector increases, the spring effect of the impedance control becomes dominant and the feedback force changes with a direct proportion. The second controller was designed to behave in-between other two controllers and the experimental results are supporting this prediction. The oscillation observed in the position tracking error plot is also observable in this plot. Finally, while ignoring the small delay caused by the initializing command of the second robot's motion, the effect of the interaction forces on the displacement of the end-effector occur in Figure 4.15. This can be calculated more accurately by implementing the spring model of the end-effector.

5. CONCLUSIONS

In this chapter, the results from the experimentation on both tasks with hybrid position/force control and extended impedance schemes are discussed; and based on these discussions, the conclusion of this study is stated.

The position and force tracking performances of the implemented hybrid position/force control and extended impedance control schemes are compared for single robot and collaborating robots tasks. Both compliance control schemes parameters are carefully tuned in order to achieve a stable robot control for the experimentation tasks. In order to ensure comparability between two compliance control schemes, the PID gains of these two control schemes have to be chosen as close as possible.

The experimental results of both fixed and moving work-piece tasks generally show that the hybrid position/force controller performs better in terms of position tracking, while the extended impedance controller performs better in terms of force tracking on similar conditions.

It is observed that force tracking is affected worse than position tracking by the collaboration motion for the tasks of this study. While the hybrid position/force controller for moving work-piece performs the force tracking worse, outstandingly the position tracking performance gets slightly better. The extended impedance controller is, however, did not perform better on position tracking but had less increase of force tracking error in the collaboration task.

In conclusion, experimental results show that both hybrid position/force control and extended impedance control schemes can be used for collaborating robot tasks. According to the result of this study, the hybrid position/force control scheme is preferable for tasks that require more precision about position tracking; and for tasks that require more precision about force tracking, the extended impedance control scheme is preferable.

The hybrid position/force control scheme has a stricter separation of force and position control on compliance frame axes. Any contact force vector deviates from the force controlled axis acts as a disturbance to the motion controlled axes. If motion control has high gains to dominate the force control, position tracking performance gets better opposed to force tracking. In order to improve both of the tracking performances, the selection matrix can be applied on a dynamically changing compliance frame.

The extended impedance control scheme has motion and force control on all compliance frame axes at the same time. A force input on any axis contributes to the controller input with respect to force control gains for that axis. This can cause a compromise on position tracking performance in favor of force tracking. An advantage of extended impedance control scheme is that by manipulating the control parameters, it can behave more like a hybrid position/force control scheme or more like a basic impedance control scheme.

REFERENCES

[1] **Zeng, G. and Hemami, A.** (1997). An overview of robot force control, *Robotica*, *15*(5), 473–482.

[2] **Vukobratovic, M.** (2009). Dynamics and robust control of robot-environment interaction, *World Scientific*, Vol. 2.

[3] **Siciliano, B.**, **Sciavicco, L.**, **Villani, L. and Oriolo, G.** (2012). *Robotics - Modelling, Planning and Control*. Springer Science & Business Media.

[4] **Sariyildiz, E. and Ohnishi, K.** (2015). On the explicit robust force control via disturbance observer, *IEEE Transactions on Industrial Electronics*, *62*(3), 1581–1589.

[5] **Jung, S. and Hsia, T.C.** (2000). Robust neural force control scheme under uncertainties in robot dynamics and unknown environment, *IEEE Transactions on Industrial Electronics*, *47*(2), 403–412.

[6] **Sariyildiz, E. and Ohnishi, K.** (2014). A comparison study for force sensor and reaction force observer based robust force control systems, *IEEE 23rd International Symposium on In Industrial Electronics (ISIE)*, 1156–1161.

[7] **Yang, R.**, **Yang, C.**, **Chen, M. and Na, J.** (2017). Adaptive impedance control of robot manipulators based on Q-learning and disturbance observer, *Systems Science & Control Engineering*, *5*(1), 287–300.

[8] **Lu, Y.** (2017). Industry 4.0: A survey on technologies, applications and open research issues. *Journal of Industrial Information Integration*, *6*, 1–10.

[9] **Raibert, M.H. and Craig, J.J.** (1981). Hybrid position/force control of manipulators, *Journal of Dynamic Systems, Measurement, and Control*, *103*(2), 126–133.

[10] **Fisher, W.D. and Mujtaba, M.S.** (1992). Hybrid position/force control: a correct formulation, *The International journal of robotics research*, *11*(4), pp.299–311.

[11] **Khatib, O.** (1987). A unified approach for motion and force control of robot manipulators: The operational space formulation, *IEEE Journal on Robotics and Automation*, *3*(1), pp.43–53.

[12] **Hogan, N.** (1985). Impedance control: An approach to manipulation: Part II—Implementation, *Journal of dynamic systems, measurement, and control*, *107*(1), 8–16.

[13] **Anderson, R.J. and Spong, M.W.** (1988). Hybrid impedance control of robotic manipulators, *IEEE Journal on Robotics and Automation*, *4*(5), 549–556.

[14] **Chiaverini, S. and Sciavicco, L.** (1993). The parallel approach to force/position control of robotic manipulators, *IEEE Transactions on Robotics and Automation*, *9*(4), 361–373.

[15] **Gierlak, P.** (2014). Hybrid position/force control in robotised machining, *Trans Tech Publications In Solid State Phenomena 210*, pp. 192–199.

[16] **Solanes, J.E.**, **Gracia, L.**, **Muñoz-Benavent, P.**, **Miro, J.V.**, **Perez-Vidal, C. and Tornero, J.** (2019). Robust hybrid position-force control for robotic surface polishing. *Journal of Manufacturing Science and Engineering*, *141*(1): 011013.

[17] **Gao, C.**, **Cong, D.**, **Liu, X.**, **Yang, Z. and Tao, H.** (2016). Hybrid position/force control of 6-dof hydraulic parallel manipulator using force and vision, *Industrial Robot: An International Journal*, *43*(3), 274–283.

[18] **Vergara, C.A. and Rodriguez, C.F.** (2017). Hybrid Position-force Control for a Stewart Platform, *Journal of Automation and Control Engineering* 5(1).

[19] **Ott, C.**, **Mukherjee, R. and Nakamura, Y.** (2010). Unified impedance and admittance control, *IEEE Int. Conf. on Robotics and Automation (ICRA)*.

[20] **Jung, S.**, **Hsia, T.C. and Bonitz, R.G.** (2004). Force tracking impedance control of robot manipulators under unknown environment, *IEEE Trans. on Control Systems Technology*, *12*(3), 474–483.

[21] **Kim, Y.** (2015). Hybrid-mode impedance control for position/force tracking in motor-system rehabilitation, *International Journal of Advanced Robotic Systems*, *12*(6), 79.

[22] **Akdogan, E.**, **Aktan, M.E.**, **Koru, A.T.**, **Arslan, M.S.**, **Atlıhan, M. and Kuran, B.** (2018). Hybrid impedance control of a robot manipulator for wrist and forearm rehabilitation: Performance analysis and clinical results, *Mechatronics*, *49*77–91.

[23] **Marques, S.J.**, **Baptista, L.F. and da Costa, J.M.S.** (1997). Hybrid impedance control of robot manipulators with neural networks compensation. *IFAC Proceedings Volumes*, *30*(3), 373–378.

[24] **Sharifi, I.**, **Talebi, H.A.**, **Mousavi, S.A.R.**, **Shemshaki, S and Tavakoli, M.** (2017). Haptic Tele-cooperation of Multiple Robots, *In 2017 5th RSI International Conference on Robotics and Mechatronics (ICRoM)* (pp. 113–119). IEEE.

[25] **Tinós, R.**, **Terra, M.H. and Ishihara, J.Y.** (2006). Motion and force control of cooperative robotic manipulators with passive joints, *IEEE Transactions on Control Systems Technology*, *14*(4), 725–734.

[26] **Martínez-Rosas, J.C.**, **Arteaga, M.A. and Castillo-Sánchez, A.M.** (2006). Decentralized control of cooperative robots without velocity–force measurements, *Automatica*, *42*(2), 329–336.

[27] **Magrini, E. and De Luca, A.** (2016). Hybrid force/velocity control for physical human-robot collaboration tasks, *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*.

[28] **Gaz, C.**, **Magrini, E. and De Luca, A.** (2018). A model-based residual approach for human-robot collaboration during manual polishing operations, *Mechatronics 55*(19), 234–247.

[29] **Url-1**, `https://www.staubli.com/en/robotics/product-range/6-axis-scara-picker-industrial-robots/6-axis-robots/rx160/`, accessed: 01.05.2019.

[30] **Url-2**, `https://www.staubli.com/en/robotics/product-range/6-axis-scara-picker-industrial-robots/6-axis-robots/rx160l/`, accessed: 01.05.2019.

[31] **Url-3**, `https://www.ati-ia.com/products/ft/ft_models.aspx?id=Delta`, accessed: 01.05.2019.

[32] **ATI Industrial Automation** (n.d.). *F/T Controller - Six-Axis Force/Torque Sensor System: Installation and Operation Manual*.

[33] **Url-4**, `https://www.staubli.com/en/robotics/product-range/robot-controller/robot-controller-cs8/cs8c/`, accessed: 01.05.2019.

[34] **Singla, P.**, **Mortari, D. and Junkins, J.L.** (2004). How to avoid singularity when using Euler angles, *In Proceedings of the AAS/AIAA 14th Space Flight Mechanics Meeting*, Maui County, HI, USA pp. 8–12.

[35] **Akbaş, S.** (2015). Force control of Stäubli RC-160 manipulator, *M.Sc. thesis*, ITU, Istanbul.

[36] **ATI Industrial Automation** (2019). *F/T Transducer - Six-Axis Force/Torque Sensor System: Installation and Operation Manual*.

[37] **Chiaverini, S.**, **Oriolo, G. and Maciejewski, A.A.** (2016). Redundant Robots. In: Siciliano B and Khatib O (eds) *Springer Handbook of Robotics*, Heidelberg: Springer-Verlag Berlin, pp.221—242.

[38] **Hutter, M.**, **Hoepflinger, M.A.**, **Gehring, C.**, **Bloesch, M.**, **Remy, C.D. and Siegwart, R.** (2013). Hybrid operational space control for compliant legged systems, In: Roy N, Newman P and Srinivasa S (eds) *Robotics: Science and Systems VIII*. MIT Press, p.129–136.

[39] **Khalil, W. and Dombre, E.** (2002). *Modeling, Identification and Control of Robots*, Taylor & Francis.

[40] **Stäubli** (2008). *Arm RX series 160 family Instruction Manual*.

[41] **Stäubli** (2008). *CS8C controller, Instruction Manual*.

[42] **Szczesiak, M.** (2016). *Force Control of Robotic Manipulators in Cooperation*, *M.Sc. thesis*, Istanbul Technical University, Graduate School of Science Engineering and Technology.

[43] **Zengin, E.** (2013). Stäubli RX160 manipulatörün modellenmesi, tanınması ve kontrolü, *M.Sc. thesis*, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul.

APPENDICES

APPENDIX A : Technical details of Stäubli RX160 and RX160L model robot manipulators
APPENDIX B : Technical details of ATI Delta F/T transducers
APPENDIX C : C code for inverse kinematics for robots with a spherical wrist
APPENDIX D : C code for identification and compensation of end-effector wrench

APPENDIX A

The technical details of Stäubli RX160 and RX160L model industrial robot manipulators are obtained from *Arm - RX series 160 family* manual [40].

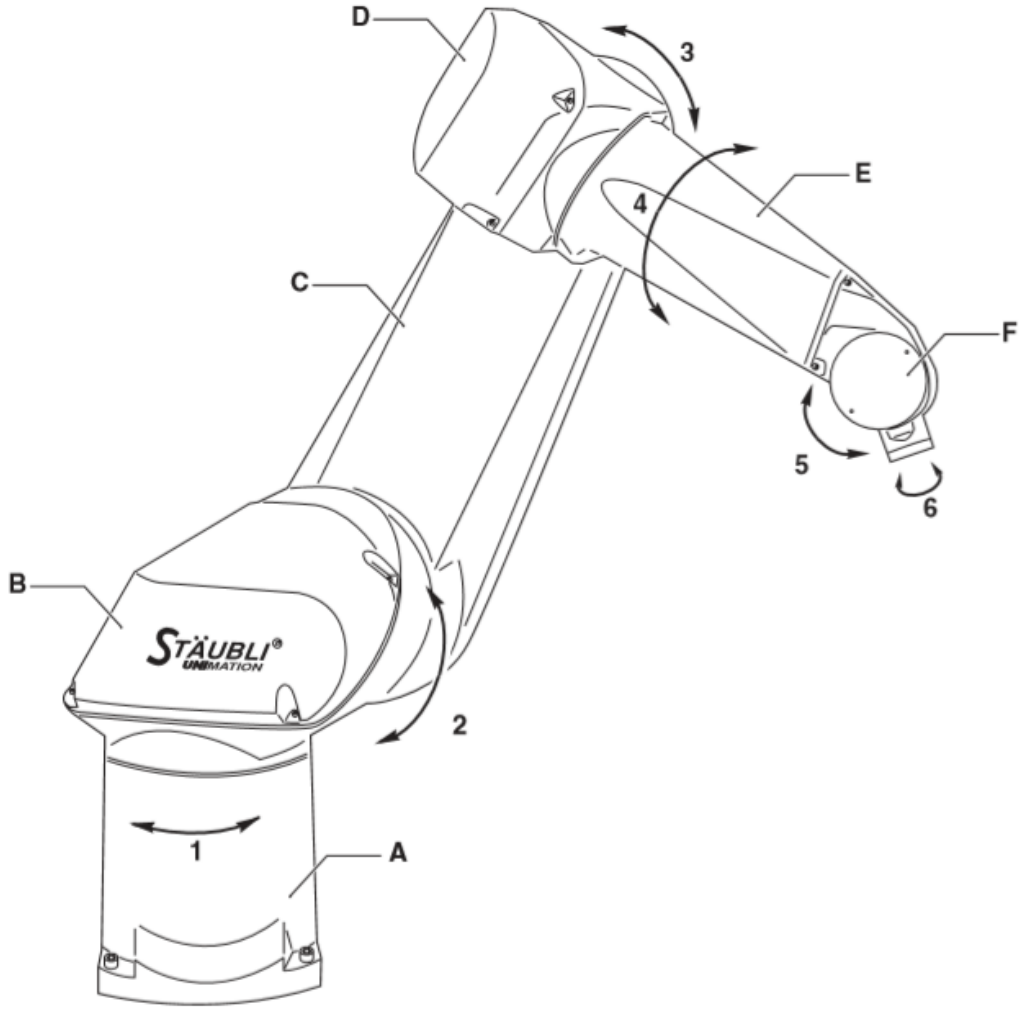

Figure A.1 : Describtion of links and joints of the RX160 family robots.

Figure A.1 describes the joints in sequenced numbers and links as: the base (A), the shoulder (B), the arm (C), the elbow (D), the forearm (E) and the wrist (F).

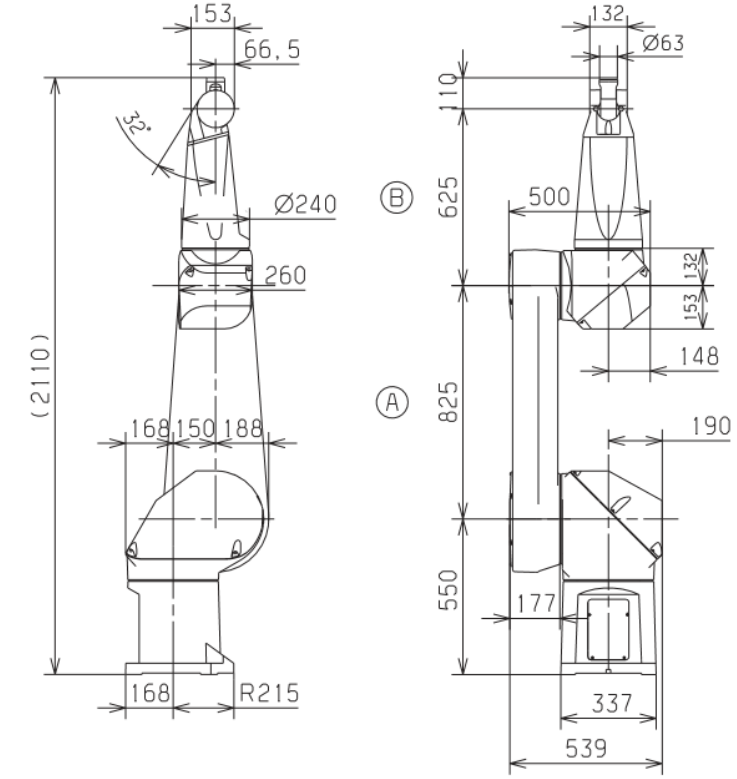

Figure A.2 : Dimensions of RX160.

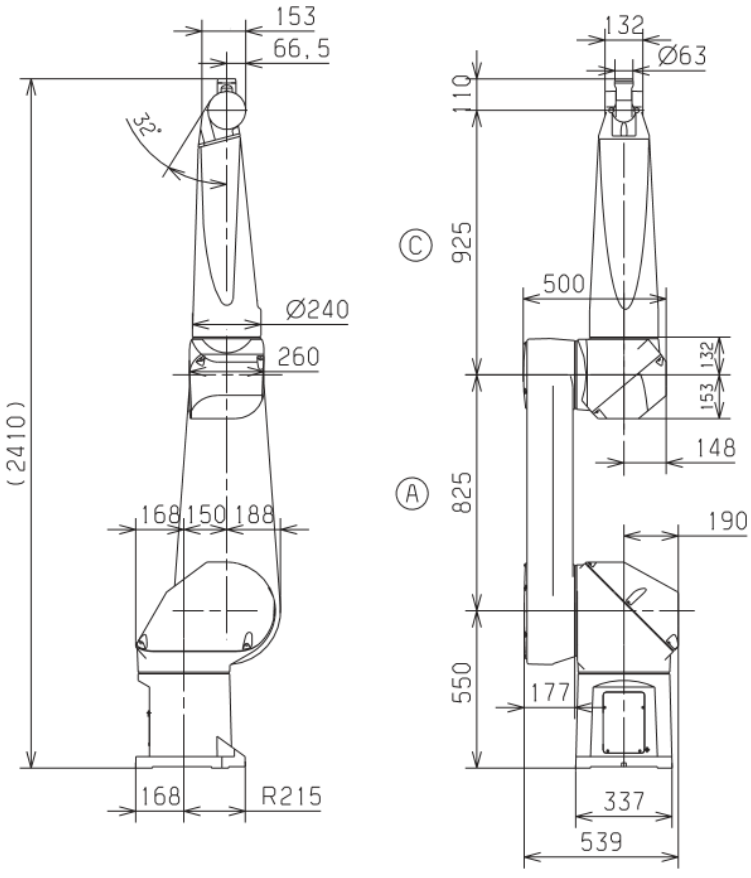

Figure A.3 : Dimensions of RX160L.

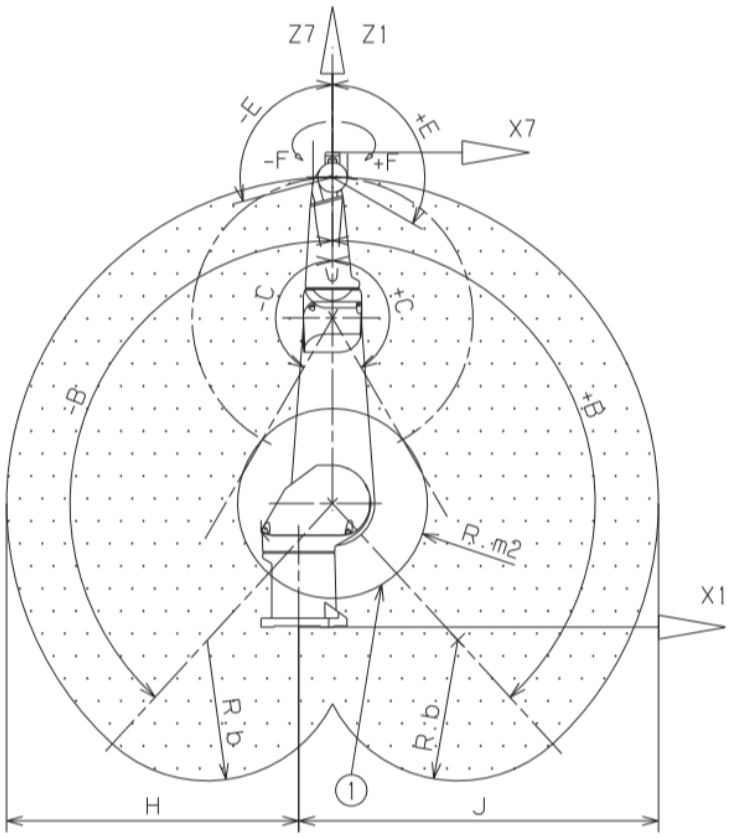

Figure A.4 : Work envelope of RX160 family robots on xz-plane.

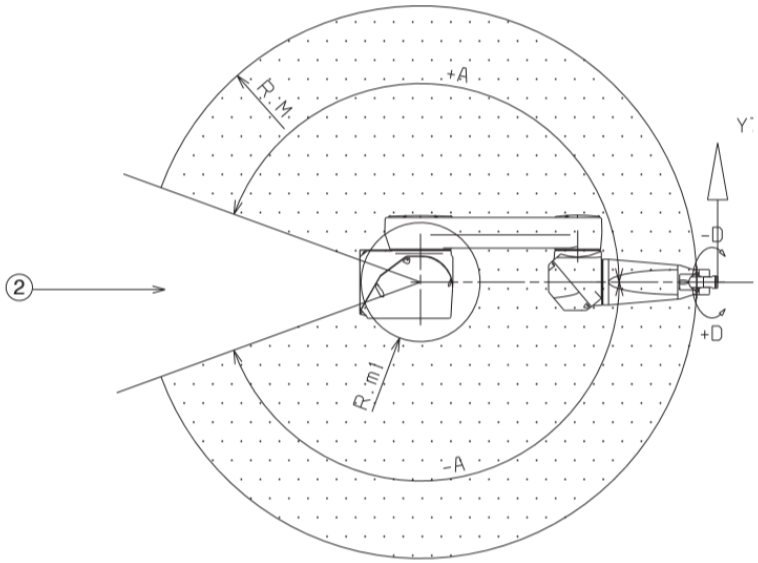

Figure A.5 : Work envelope of RX160 family robots on xy-plane.

Table A.1 : Work envelope of RX160 family.

	RX160	RX160L
R.M max. reach between axis 1 and 5	1600mm	1900mm
R.m1 min. reach between axis 1 and 5	422mm	463mm
R.m2 min. reach between axis 2 and 5	312mm	462mm
R.b reach between axis 3 and 5	625mm	925mm
H	1300mm	1600mm
J	1600mm	1900mm

Table A.2 : Amplitude, speed and resolution of RX160 family.

Axis	1	2	3	4	5	6
Amplitude (°)	320	275	300	540	225	540
Working range distribution (°)	A±160	B±137.5	C±150	D±270	E+120-105	F±270
Nominal speed (°/s)	165	150	190	295	260	440
Maximum speed* (°/s)	278	278	356	409	800	1125
Angular resolution ($°.10^{-3}$)	0.68	0.68	0.87	1.0	1.95	2.75

Maximum Cartesian speed: 2.5 m/s.
* Maximum speed for reduced conditions of load and inertia.

APPENDIX B

The technical details of ATI Delta F/T transducer is obtained from *F/T Transducer* and *F/T Controller* installation and operation manuals [36] and [32] respectively.

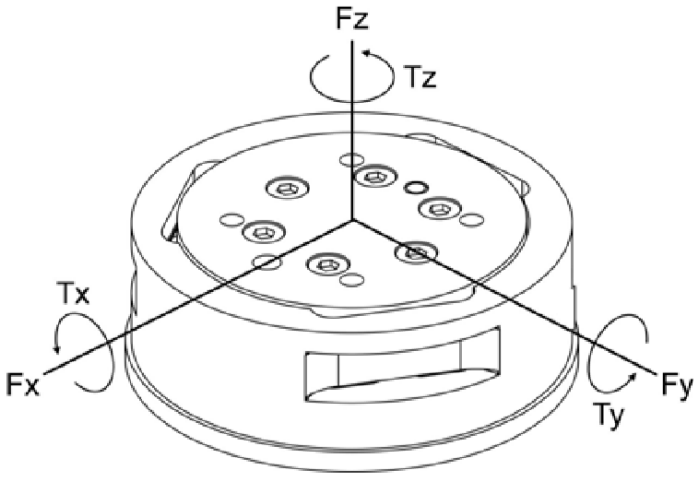

Figure B.1 : Placement of sensor frame on the F/T transducer.

Table B.1 : Delta calibrations.

(SI) Metric Calibration	Fx,Fy (N)	Fz (N)	Tx,Ty (Nm)	Tz (Nm)	Fx,Fy (N)	Fz (N)	Tx,Ty (Nm)	Tz (Nm)
SI-165-15	165	495	15	15	1/32	1/16	1/528	1/528
SI-330-30	330	990	30	30	1/16	1/8	5/1333	5/1333
SI-660-60	660	1980	60	60	1/8	1/4	10/1333	10/1333
	Sensing Ranges				Resolution (DAQ,Net F/T)			

Table B.2 : Delta physical properties.

Single axis overload	(SI) Metric Units
Fxy	$\pm$3700 N
Fz	$\pm$10000 N
Txy	$\pm$280 Nm
Tz	$\pm$400 Nm
Stiffness (Calculated)	
X-axis & Y-axis forces (Kx, Ky)	3.6×10^7 N/m
Z-axis forces (Kz)	5.9×10^7 N/m
X-axis & Y-axis torque (Ktx, Kty)	5.2×10^4 Nm/rad
Z-axis torque (Kz)	9.1×10^4 Nm/rad
Resonant Frequency	
Fx, Fy, Tz	1500 Hz
Fz, Tx, Ty	1700 Hz
Physical Specifications	
Weight[1]	0.913 kg
Diameter[1]	94.5 mm
Height[1]	33.3 mm

Note: 1. Specifications include standard interface plates.

APPENDIX C

The function in C code used for inverse kinematics for robots with a spherical wrist.

```
int getInvGeoKhalil(void)
{
double Px, Py, Pz;
double r1, r6, RL1, RL3, RL4, D2, D3;
double q[6], q_posFbk[6];
double W, X, Y, Z, Z1, Z2;
double B1, B2, B3;
double C2, C3, C5, C6, S2, S3, S5, S6;
int e;

unsigned int n, l_jnt;
double d[3], theta[3], a[3], alpha[3];
double SQ, CQ;

mdarray *Tj_i_h, *Tj_0_h, *A3_0, *FGH;

r1 = _kd.d[0];  // 0.550 + ...
r6 = _kd.d[5];  // 0.110
RL1 = 0.0;
RL3 = _kd.d[1];
RL4 = _kd.d[3]; // 0.625
D2 = _kd.a[1];  // 0.150
D3 = _kd.a[2];  // 0.825

for (l_jnt = 0; l_jnt < g_jntNb; l_jnt++)
q_posFbk[l_jnt] = mdarray1_get(_ad.q_posFbk, l_jnt + 1);

/// Transform position from base-to-end to shoulder-to-wrist

// Transform from base-to-end to base-to-wrist
//      Tend_wrist = [  1 0 0   0
//                                  0 1 0   0
//                                  0 0 1 - r(6)
//                                  0 0 0   1       ];
//
//      Tbase_wrist = Tbase_end * Tend_wrist

Px = mdarray_get(_kd.T0_6Des, 1, 4)
- (mdarray_get(_kd.T0_6Des, 1, 3) * r6);
Py = mdarray_get(_kd.T0_6Des, 2, 4)
```

```
- (mdarray_get(_kd.T0_6Des, 2, 3) * r6);
Pz = mdarray_get(_kd.T0_6Des, 3, 4)
- (mdarray_get(_kd.T0_6Des, 3, 3) * r6);

// Transform from base-to-wrist to shoulder-to-wrist
//      Tshoulder_wrist(3,4) = Tbase-to-wrist(3,4) - r(1);

Pz = Pz - r1;

// a) Computation of theta1, theta2, theta3

// theta 1
if (RL3 == 0)
{
q[0] = atan2(Py, Px);
}
else
{ // Type 2
X = -Px;
Y = Py;
Z = -RL3;

e = -1;

SQ = (X*Z + e * Y * sqrt(pow(X, 2) + pow(Y, 2)
- pow(Z, 2))) / (pow(X, 2) + pow(Y, 2));
CQ = (Y*Z - e * X * sqrt(pow(X, 2) + pow(Y, 2)
- pow(Z, 2))) / (pow(X, 2) + pow(Y, 2));

q[0] = atan2(SQ, CQ);

if (fabs(q_posFbk[0] - (q[0] - PI)) < fabs(q_posFbk[0] - q[0]))
q[0] = q[0] - PI;
else if (fabs(q_posFbk[0] - (q[0] + PI)) < fabs(q_posFbk[0] - q[0]))
q[0] = q[0] + PI;
}

// theta 2 & 3

// W SQj = X CQi + Y SQi + Z1
// W CQj = X SQi - Y CQi + Z2

W = -RL4;
X = -cos(q[0])*Px - sin(q[0])*Py + D2;
Y = Pz - RL1;
Z1 = -D3;
Z2 = 0;
```

```
// Type 6 Function

B1 = 2 * (Z1*Y + Z2 * X);
B2 = 2 * (Z1*X - Z2 * Y);
B3 = pow(W, 2) - pow(X, 2) - pow(Y, 2) - pow(Z1, 2)
- pow(Z2, 2);

if (_kd.elbow != LFT_POS)
e = -1;
else
e = 1;

C2 = (B2*B3 - e * B1*sqrt(pow(B1, 2) + pow(B2, 2)
- pow(B3, 2)))
/ (pow(B1, 2) + pow(B2, 2));
S2 = (B1*B3 + e * B2*sqrt(pow(B1, 2) + pow(B2, 2)
- pow(B3, 2)))
/ (pow(B1, 2) + pow(B2, 2));

q[1] = atan2(S2, C2);

S3 = (X*C2 + Y * S2 + Z1) / W;
C3 = (X*S2 - Y * C2 + Z2) / W;

q[2] = atan2(S3, C3);

// b) Computation of theta4, theta5, theta6

d[0] = 0.0;
d[1] = _kd.a[0];
d[2] = _kd.a[1];

a[0] = _kd.d[0];
a[1] = 0.0;
a[2] = 0.0;

alpha[0] = 0.0;
alpha[1] = _kd.alpha[0];
alpha[2] = 0.0;

theta[0] = q[0];
theta[1] = q[1];
theta[2] = q[2];

Tj_i_h = mdarray3_new(4, 4, 4);
Tj_0_h = mdarray3_new(4, 4, 4);
for (n = 1; n <= 4; n++)
{
```

```
// Transformation matrices (wrt previous coordinate)
if (n > 1)
{
mdarray3_set(Tj_i_h, 1, 1, n, cos(theta[n - 2]));
mdarray3_set(Tj_i_h, 1, 2, n, cos(alpha[n - 2])
* sin(theta[n - 2]));
mdarray3_set(Tj_i_h, 1, 3, n, sin(alpha[n - 2])
* sin(theta[n - 2]));
mdarray3_set(Tj_i_h, 1, 4, n, -d[n - 2] * cos(theta[n - 2]));
mdarray3_set(Tj_i_h, 2, 1, n, -sin(theta[n - 2]));
mdarray3_set(Tj_i_h, 2, 2, n, cos(alpha[n - 2])
* cos(theta[n - 2]));
mdarray3_set(Tj_i_h, 2, 3, n, sin(alpha[n - 2])
* cos(theta[n - 2]));
mdarray3_set(Tj_i_h, 2, 4, n, d[n - 2] * sin(theta[n - 2]));
mdarray3_set(Tj_i_h, 3, 1, n, 0.0);
mdarray3_set(Tj_i_h, 3, 2, n, -sin(alpha[n - 2]));
mdarray3_set(Tj_i_h, 3, 3, n, cos(alpha[n - 2]));
mdarray3_set(Tj_i_h, 3, 4, n, -a[n - 2]);
mdarray3_set(Tj_i_h, 4, 1, n, 0.0);
mdarray3_set(Tj_i_h, 4, 2, n, 0.0);
mdarray3_set(Tj_i_h, 4, 3, n, 0.0);
mdarray3_set(Tj_i_h, 4, 4, n, 1.0);
}
else
{
mdarray3_setpag_identity(Tj_i_h, n);
}

// Transformation matrices (wrt base)
if (n == 1)
mdarray3_memcpy_page(Tj_0_h, n, Tj_i_h, n);
else
mdarray3_mul_pags(Tj_0_h, n, Tj_i_h, n, Tj_0_h, n - 1);
}

mdarray_free(Tj_i_h);

A3_0 = mdarray_new(3, 3);
mdarray3_submatrix(A3_0, 1, Tj_0_h, 1, 3, 1, 3, 4);

mdarray_free(Tj_0_h);

// FGH = [ F G H ] = A3_0 * sna;
FGH = mdarray_new(3, 3);
mdarray_mul(FGH, A3_0, _kd.A0_6Des);

mdarray_free(A3_0);
```

```
// theta 4
q[3] = atan2(mdarray_get(FGH, 3, 3), mdarray_get(FGH, 1, 3));

if (fabs(q_posFbk[3] - (q[3] - PI))
< fabs(q_posFbk[3] - q[3]))
q[3] = q[3] - PI;
else if (fabs(q_posFbk[3] - (q[3] + PI / 2))
< fabs(q_posFbk[3] - q[3]))
q[3] = q[3] + PI / 2;
else if (fabs(q_posFbk[3] - (q[3] - PI / 2))
< fabs(q_posFbk[3] - q[3]))
q[3] = q[3] - PI / 2;
else if (fabs(q_posFbk[3] - (q[3] + 2 * PI))
< fabs(q_posFbk[3] - q[3]))
q[3] = q[3] + 2 * PI;

// theta 5
S5 = sin(q[3])*mdarray_get(FGH, 3, 3) + cos(q[3])
*mdarray_get(FGH, 1, 3);
C5 = -mdarray_get(FGH, 2, 3);

q[4] = atan2(S5, C5);

// theta 6
S6 = cos(q[3])*mdarray_get(FGH, 3, 1) - sin(q[3])
*mdarray_get(FGH, 1, 1);
C6 = cos(q[3])*mdarray_get(FGH, 3, 2) - sin(q[3])
*mdarray_get(FGH, 1, 2);

q[5] = atan2(S6, C6);

mdarray_free(FGH);

// finalize desired joint positions
for (l_jnt = 0; l_jnt < g_jntNb; l_jnt++)
mdarray1_set(_ad.q_posDes, l_jnt + 1, q[l_jnt]
+ _kd.theta[l_jnt]);

return 0;
}
```

APPENDIX D

The functions in C code used for identification of mass and CoM of attachments on top the F/T transducer.

```
// ------------------------------------------------------------
// computeCompensation
// ------------------------------------------------------------
void computeCompensation(void)
{
        mdarray *A6_0Fbk;
        mdarray *tool_weight_fVec_wrld, *tool_weight_fVec_sxth;

        tool_weight_fVec_wrld = mdarray1_new(3);
        tool_weight_fVec_sxth = mdarray1_new(3);

        // get rotation matrix end to base
        A6_0Fbk = mdarray_new(3, 3);
        mdarray_transpose(A6_0Fbk, _kd.A0_6Fbk);

        // tool weight vector [N]
        mdarray1_set3(tool_weight_fVec_wrld, 0.0, 0.0,
        -(_dd.tool_mass * GRAV));

        // transform tool weight vector from base frame to
        // tool frame
        mdarray_mul(tool_weight_fVec_sxth, A6_0Fbk,
        tool_weight_fVec_wrld);

        // transform tool weight vector from tool frame to
        // sensor frame
        rotateOnAnAxis(_sd.tool_weight_fVec_snsr
        , tool_weight_fVec_sxth, 'z', _kd.sensor_mount_ang);
        mdarray_cross(_sd.tool_weight_tVec_snsr, _dd.tool_CoM_sF
        , _sd.tool_weight_fVec_snsr);

        mdarray_free(A6_0Fbk);

        mdarray_free(tool_weight_fVec_wrld);
        mdarray_free(tool_weight_fVec_sxth);
}
```

```
// ------------------------------------------------------------
// sumData
// ------------------------------------------------------------
void sumData(int motNb)
{
        unsigned int i;

        if (_sd.data_summation[motNb] == 1)
        {
                if (_ad.m_time < _td.tf[motNb] / g_cycleTime)
                {
                        for (i = 0; i < 6; i++)
                        {
                                _sd.sensor_data_sum[i] +=
                                mdarray1_get(_ad.w_snsrFr
                                , i + 1);
                        }

                        ++_sd.s_cycle;
                }
        }
        else
        {
                _sd.s_cycle = 0;
        }
}

// ------------------------------------------------------------
// averageData
// ------------------------------------------------------------
void averageData(double outputVector[6])
{
        unsigned int i;

        printf("\ns_cycle: %d", _sd.s_cycle);
        for (i = 0; i < 6; i++)
        {
                printf("\nsensor_sum_data: %f"
                , _sd.sensor_data_sum[i]);
                outputVector[i] = _sd.sensor_data_sum[i]
                / _sd.s_cycle;

                _sd.sensor_data_sum[i] = 0.0;
        }
        printf("\n");
}
```

```
// ------------------------------------------------------------
// sensorIdentification
// ------------------------------------------------------------
void sensorIdentification(void)
{
        // compute final sensor force data using xMax
        // and yMax data
        mdarray1_set3(_sd.f_sensor_static_bias
        ,(_sd.sensor_FyMax_data[0]+_sd.sensor_FyMin_data[0])/2
        ,(_sd.sensor_FyMax_data[1]+_sd.sensor_FyMin_data[1])/2
        ,_sd.sensor_FyMin_data[2]);
        mdarray1_set3(_sd.t_sensor_static_bias
        ,(_sd.sensor_FyMin_data[3]+_sd.sensor_FyMax_data[3])/2
        ,_sd.sensor_FyMax_data[4]
        ,(_sd.sensor_FyMax_data[5]+_sd.sensor_FyMin_data[5])/2);

        printf("\n_sd.sensor_FyMax_data[1]:␣%f"
        , _sd.sensor_FyMax_data[1]);
        printf("\n_sd.sensor_FyMin_data[1]:␣%f"
        , _sd.sensor_FyMin_data[1]);
        printf("\ntool␣mass:␣%f"
        , (_sd.sensor_FyMax_data[1] - _sd.sensor_FyMin_data[1])
        / (2 * GRAV));
        _dd.tool_mass = (_sd.sensor_FyMax_data[1]
        - _sd.sensor_FyMin_data[1]) / (2 * GRAV);

        mdarray1_set3(_dd.tool_CoM_sF
        , (_sd.sensor_FyMax_data[5]
        - mdarray1_get(_sd.t_sensor_static_bias, 3))
        / (_dd.tool_mass * GRAV)
        , -(_sd.sensor_FxMax_data[5]
        - mdarray1_get(_sd.t_sensor_static_bias, 3))
        / (_dd.tool_mass * GRAV)
        , (_sd.sensor_FyMin_data[3]
        - mdarray1_get(_sd.t_sensor_static_bias, 1)
        / (_dd.tool_mass * GRAV));

        rotateOnAnAxis(_dd.tool_CoM_6F, _dd.tool_CoM_sF, 'z'
        , -_kd.sensor_mount_ang);
        mdarray1_set(_dd.tool_CoM_6F, 3
        , mdarray1_get(_dd.tool_CoM_6F, 3)
        + _kd.sensor_length);

        initDynPara();
        computeCompensation();
}
```

CURRICULUM VITAE

Name Surname: Mertcan Kaya

Place and Date of Birth: Edirne, Turkey, 5^{th} April 1991

E-Mail: mertcan.kaya@gmail.com

EDUCATION:

- **B.Sc.:** 2014, Koç University, Collage of Engineering, Mechanical Engineering