ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

CONTROL OF VIRTUAL STAUBLI RX160 MANIPULATOR BY PHANTOM PREMIUM HAPTIC DEVICE

M.Sc. THESIS

Aykut GÖREN

Department of Mechatronics Engineering

Mechatronics Engineering Programme

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

MAY 2014

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

CONTROL OF VIRTUAL STAUBLI RX160 MANIPULATOR BY PHANTOM PREMIUM HAPTIC DEVICE

M.Sc. THESIS

Aykut GÖREN
(518111003)

Department of Mechatronics Engineering

Mechatronics Engineering Programme

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

MAY 2014

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

SANAL STAUBLI RX160 MANİPÜLATÖRÜN PHANTOM PREMIUM HAPTIC CİHAZ İLE KONTROLÜ

YÜKSEK LİSANS TEZİ

Aykut GÖREN
(518111003)

Mekatronik Mühendisliği Anabilim Dalı

Mekatronik Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Zeki Yağız BAYRAKTAROĞLU

MAYIS 2014

Aykut GÖREN, a M.Sc. student of ITU Graduate School of Science Engineering and Technology student ID 518111003, successfully defended the thesis entitled "**CONTROL OF VIRTUAL STAUBLI RX160 MANIPULATOR BY PHANTOM PREMIUM HAPTIC DEVICE**", 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. Şeniz ERTUĞRUL**
Istanbul Technical University

Assist. Prof. Dr. Kadir ERKAN
Yıldız Technical University

Date of Submission : 5 May 2014
Date of Defense : 30 May 2014

To my family,

FOREWORD

This master’s thesis is about developing a software which controls a virtual Staubli RX160 manipulator by a Phantom Premium haptic device and was written in the Department of Mechatronics Engineering of Istanbul Technical University.
I would like to thank my master’s thesis advisor Assoc. Prof. Dr. Zeki Yağız BAYRAKTAROĞLU, my colleagues and friends in Arçelik Inc. and Istanbul Technical University and my family for their moral and material support.

May 2014

Aykut GÖREN
(Mechanical Engineer)

TABLE OF CONTENTS

Page

FOREWORD .... **ix**
TABLE OF CONTENTS .... **xi**
ABBREVIATIONS .... **xiii**
LIST OF TABLES .... **xv**
LIST OF FIGURES .... **xvii**
SUMMARY .... **xxi**
ÖZET .... **xxiii**
1. INTRODUCTION **1**
  1.1 Purpose of Thesis 1
  1.2 Literature Review 2
  1.3 Staubli RX160 3
  1.4 Phantom Premium 1.5 High Force 6DOF 5
2. HAPTICS **11**
  2.1 Virtual Reality (VR) 13
  2.2 Haptic Rendering 13
  2.3 OpenHaptics SDK(Software Development Kit) 14
    2.3.1 OpenGL(Open Graphics Library) API 15
    2.3.2 QuickHaptics micro API 16
    2.3.3 HDAPI 21
    2.3.4 HLAPI 22
    2.3.5 Multithreading 25
    2.3.6 Mapping the device space to world space 26
    2.3.7 Calibration 27
3. MODELING OF 3 DOF STAUBLI RX160 .... **31**
  3.1 Geometric Model 31
    3.1.1 Homogeneous transformation matrices 31
    3.1.2 Forward geometric model 33
    3.1.3 Inverse geometric model 36
  3.2 Kinematic Model 41
    3.2.1 Jacobian matrix 41
    3.2.2 Forward kinematic model 43
    3.2.3 Inverse kinematic model 43
  3.3 Static Model 43
  3.4 Dynamic Model 44
    3.4.1 Robot dynamic parameters 44
      3.4.1.1 Inertial parameters 45
      3.4.1.2 Friction 45
      3.4.1.3 Balancing system 47
    3.4.2 Newton-Euler formulation 48
      3.4.2.1 Newton-Euler recursive computation 51

4. VISUAL INTERFACE 53
4.1 Virtual Model of Staubli RX160 53
4.2 Motion Algorithm of the Staubli RX160 54
4.2.1 Hierarchy modeling 55
4.2.2 Proxy rendering 56
4.2.3 Motion algorithm of the virtual Staubli RX160 56
5. APPLICATION AND EXPERIMENTS 59
5.1 Application Setup 59
5.2 Experiments 60
6. CONCLUSION AND RECOMMENDATIONS 89
REFERENCES 91
APPENDICES 93
APPENDIX A 97
APPENDIX B 97
CURRICULUM VITAE 109

ABBREVIATIONS

HF : High Force
DOF : Degrees of Freedom
Hz : Hertz
VR : Virtual Reality
EPP : Enhanced Parellel Por
PDD : Phantom Device Drivers
HAVE : Hapto-Audio-Virtual-Environment
SDK : Software Development Kit
API : Application Programming Interface
IDE : Integrated Development Environment

LIST OF TABLES

Page

Table 1.1 : Amplitude, speed and resolution parameters of the Staubli RX160[7]. ... 4
Table 1.2 : Power Specifications[9]. 7
Table 3.1 : Types of equations encountered in Paul method[14]. 37

LIST OF FIGURES

Page

Figure 1.1 : The Staubli RX160[7]. 3
Figure 1.2 : The physical dimension of the Staubli RX160[7]. 4
Figure 1.3 : The work envelope of the Staubli RX160[7]. 5
Figure 1.4 : The work envelope of the Staubli RX160. 6
Figure 1.5 : Sockets and switches of the Phantom Premium 1.5 HF 6 DOF. 7
Figure 1.6 : Phantom Test Interface. 8
Figure 2.1 : The spectrum and trend for HAVE[11]. 11
Figure 2.2 : Approximate sensing frequencies of the cultaneous mechanoreceptors, kinesthetic and the range of the kinesthetic control[12]. 12
Figure 2.3 : Basic architecture for a virtual reality application incorporating visual, auditory and haptic feedback[11]. 13
Figure 2.4 : Haptic rendering block diagram[11]. 14
Figure 2.5 : The OpenHaptics overview[8]. 15
Figure 2.6 : Diagram of the OpenGL rendering pipeline[13]. 16
Figure 2.7 : Default QuickHaptics camera location[8]. 17
Figure 2.8 : Default clipping planes for world space[8]. 18
Figure 2.9 : Haptic interface point. 19
Figure 2.10 : The Cursor class association[8]. 20
Figure 2.11 : QuickHaptics micro API program flow[8]. 20
Figure 2.12 : HDAPI program flowchart[8]. 22
Figure 2.13 : The proxy position[8]. 24
Figure 2.14 : The HLAPI program flowchart [8]. 24
Figure 2.15 : The haptic frames inside the HLAPI program flowchart [8]. 25
Figure 2.16 : Thread structures. 26
Figure 2.17 : The haptic device workspace coordinates to world coordinates mapping[8]. 27
Figure 2.18 : Calibration position. 28
Figure 2.19 : Calibration interface. 29
Figure 3.1 : Transformation between frame Ri and frame Rj 32
Figure 3.2 : Robot Views: (a)Front View. (b)Right View. (c)Top View. (d)Perspective View. 34
Figure 3.3 : Link coordinate frames of 3 DOF Staubli RX 160 robot. 35
Figure 3.4 : Workspace of this study. 42
Figure 3.5 : Coulomb, viscous and static friction model[15]. 46
Figure 3.6 : Stribeck effect of friction model[15]. 46
Figure 3.7 : Staubli RX160 balancing system[7]. 48
Figure 3.8 : Force, torque and position vctors on Staubli RX160 manipulator. 50
Figure 3.9 : Forward and backward recursion shema. 52
Figure 4.1 : Links of the Staubli RX 160 manipulator. 53
Figure 4.2 : End-effector of the Staubli RX160 manipulator. 54

Figure 4.3 : Virtual model of the Staubli RX 160 manipulator. 54
Figure 4.4 : Tree-structured hierarchy model of the Staubli RX160 manipulator. ... 55
Figure 4.5 : The illustration of the motion algorithm for a position change of the end-effector. 57
Figure 4.6 : The visual interface window.. 58
Figure 5.1 : Application Setup. 60
Figure 5.2 : General scheme of the application.. 61
Figure 5.3 : Block diagram of the application.. 61
Figure 5.4 : First experiment path. 63
Figure 5.5 : Experimental results for first path, $K1 = 1$ and $K2 = 0{,}01$: (a)Position. (b)Velocity. (c)Acceleration. 64
Figure 5.6 : Force graphs for first path, $K1 = 1$ and $K2 = 0{,}01$: (a)Computed virtual end-effector force. b)Limited force applied to the haptic device. 65
Figure 5.7 : Experimental results for first path, $\mathrm{K1} = 1$ and $\mathrm{K2} = 0{,}001$: (a)Position. (b)Velocity. (c)Acceleration. 66
Figure 5.8 : Force graphs for first path, $\mathrm{K1} = 1$ and $\mathrm{K2} = 0{,}001$: (a)Computed virtual end-effector force. b)Limited force applied to the haptic device. 67
Figure 5.9 : Experimental results for first path, $\mathrm{K1} = 1$ and $\mathrm{K2} = 0{,}0001$: (a)Position. (b)Velocity. (c)Acceleration. 68
Figure 5.10 : Force graphs for first path, $K1 = 1$ and $K2 = 0{,}0001$: (a)Computed virtual end-effector force. b)Limited force applied to the haptic device. 69
Figure 5.11 : Experimental results for first path, $K1 = 10$ and $K2 = 0{,}01$: (a)Position. (b)Velocity. (c)Acceleration. 70
Figure 5.12 : Force graphs for first path, $K1 = 10$ and $K2 = 0{,}01$: (a)Computed virtual end-effector force. b)Limited force applied to the haptic device. 71
Figure 5.13 : Experimental results for first path, $K1 = 10$ and $K2 = 0{,}001$: (a)Position. (b)Velocity. (c)Acceleration. 72
Figure 5.14 : Force graphs for first path, $K1 = 10$ and $K2 = 0{,}001$: (a)Computed virtual end-effector force. b)Limited force applied to the haptic device. 73
Figure 5.15 : Experimental results for first path, $K1 = 10$ and $K2 = 0{,}0001$: (a)Position. (b)Velocity. (c)Acceleration. 74
Figure 5.16 : Force graphs for first path, $K1 = 10$ and $K2 = 0{,}0001$: (a)Computed virtual end-effector force. b)Limited force applied to the haptic device. 75
Figure 5.17 : Second experiment path... 76
Figure 5.18 : Experimental results for first path, $K1 = 1$ and $K2 = 0{,}01$: (a)Position. (b)Velocity. (c)Acceleration. 77
Figure 5.19 : Force graphs for first path, $K1 = 1$ and $K2 = 0{,}01$: (a)Computed virtual end-effector force. b)Limited force applied to the haptic device. 78
Figure 5.20 : Experimental results for first path, $K1 = 1$ and $K2 = 0{,}001$: (a)Position. (b)Velocity. (c)Acceleration. 79

Figure 5.21 : Force graphs for first path, $\mathrm{K}1 = 1$ and $\mathrm{K}2 = 0{,}001$: (a)Computed virtual end-effector force. b)Limited force applied to the haptic device. 80
Figure 5.22 : Experimental results for first path, $\mathrm{K}1 = 1$ and $\mathrm{K}2 = 0{,}0001$: (a)Position. (b)Velocity. (c)Acceleration. 81
Figure 5.23 : Force graphs for first path, $K1 = 1$ and $K2 = 0{,}0001$: (a)Computed virtual end-effector force. b)Limited force applied to the haptic device. 82
Figure 5.24 : Experimental results for first path, $K1 = 10$ and $K2 = 0{,}01$: (a)Position. (b)Velocity. (c)Acceleration. 83
Figure 5.25 : Force graphs for first path, $K1 = 10$ and $K2 = 0{,}01$: (a)Computed virtual end-effector force. b)Limited force applied to the haptic device. 84
Figure 5.26 : Experimental results for first path, $K1 = 10$ and $K2 = 0{,}001$: (a)Position. (b)Velocity. (c)Acceleration. 85
Figure 5.27 : Force graphs for first path, $K1 = 10$ and $K2 = 0{,}001$: (a)Computed virtual end-effector force. b)Limited force applied to the haptic device. 86
Figure 5.28 : Experimental results for first path, $K1 = 10$ and $K2 = 0{,}0001$: (a)Position. (b)Velocity. (c)Acceleration. 87
Figure 5.29 : Force graphs for first path, $K1 = 10$ and $K2 = 0{,}0001$: (a)Computed virtual end-effector force. b)Limited force applied to the haptic device. 88

Figure A.1 : QuickHaptics micro API classes and properties[8]. 95
Figure B.1 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force. 97
Figure B.2 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force. 98
Figure B.3 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force. 99
Figure B.4 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force. 100
Figure B.5 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force. 101
Figure B.6 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force. 102
Figure B.7 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force. 103
Figure B.8 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force. 104
Figure B.9 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force. 105
Figure B.10 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force. 106
Figure B.11 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force. 107
Figure B.12 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force. 108

CONTROL OF VIRTUAL STAUBLI RX160 MANIPULATOR BY PHANTOM PREMIUM DEVICE

SUMMARY

Robots can be used for a variety of purposes in diverse application areas. Therefore, a wide range of tasks can be defined for robots. For satisfying these complicated tasks, robots need to be intelligent. Making robots intelligent is a continuously developable area and one of the effective ways for this purpose is improving the robot sensations. In this context, integrating tactile sensation to a robotic application is one of the objectives of this thesis.

Haptics is the science of incorporating the tactile and/or kinesthetic sensations into the human-computer interaction. It has a broad range of applications in both commercial and scientific researches. Some of these application areas are: Virtual reality, robotic control, teleoperations or telerobotics, rehabilitation, tele-rehabilitation, training simulations such as medical, surgical and dental simulations, virtual assembly, collision detection, molecular modeling, FEM (Finite Element Method) applications, nano manipulation, entertainment and games, remote manipulations for nuclear and hazardeus applications and 3D modeling.

In this thesis, a computer software is written to interact with the virtual model of the 3 DOF Staubli RX160 manipulator via Phantom Premium 1.5 High Force 6 DOF haptic device. OpenHaptics SDK which is based on the C/C++ programming language is used for programming. OpenHaptics have 3 APIs: QuickHaptics micro API, HLAPI and HDAPI. All of these 3 APIs are used in this study to take advantage of each API. A visual interface is designed to obtain a visual feedback of the application by using OpenHaptics commands which are based on the OpenGL API.

In the application, human operator moves the haptic interface point and a position change occurs. By using this position change information, the linear velocity and acceleration of the haptic interface point are computed by considering time. Then, using the position change, velocity and acceleration information of the haptic interface point, the haptic rendering algorithm computes the resulting forces of the dynamic model of the virtual 3 DOF Staubli manipulator and the graphics rendering algorithm computes the resulting motion of the virtual Staubli manipulator in virtual environment.

Human tactile and visual perception frequencies are approximately 1000 and 30 Hz respectively. Therefore, to obtain the sense of reality that occurs in user, the haptic rendering algorithm and the graphics rendering algorithm operating frequencies are determined as 1000 Hz and 30 Hz respectively. In the developed software, these two algorithms are programmed in seperate threads which run in parallel by utilizing

the OpenHaptics APIs.

A virtual tool is designed as an end-effector of the Staubli manipulator and placed to the top of the tool flange of the Staubli manipulator. First 3 DOF of Staubli RX160 manipulator provides to access to the major part of its workspace and the last 3 DOF provides the orientation of the end-effector. First 3 DOF of Staubli RX160 are modeled in this study. The geometric, inverse geometric, kinematic, inverse kinematic, static, dynamic and inverse dynamic model of the 3 DOF Staubli RX160 manipulator are derived. Second link of the Staubli RX160 is equipped with a spring ballance system and it is provided by Staubli. The dynamic and inverse dynamic model of the Staubli manipulator include this spring model and the joint friction models.

The position change of the haptic interface point is mapped to the position change of the end-effector of the virtual 3 DOF Staubli RX160. Then, the resulting forces of the dynamic model of the virtual 3 DOF Staubli RX160 are mapped to a limited range of force which is inside the Phantom haptic device capabilities. Position and force mappings are uniform, in other words, all the axes are mapped in the same proportion. The position and force scaling coefficients express the position and force gain respectively.

In the experiment stage, some specific conditions are determined and the experiments are realized for these conditions. These conditions consist of diverse motion paths, position gains and force gains. During the experiments, the end-effector of the virtual 3 DOF Staubli RX160 manipulator follows the haptic interface point movement, the resulting visual and force feedback are applied to the human operator and the developed software records the position, velocity, acceleration and force informations. After the application is stopped, the software writes all of the recorded data to a file.

Consequently, the determined experiments are realized and the resulting graphs are plotted. Then, the stability of the system is investigated. Numerical derivations and other numerical computations caused to instability and “force kicking” in the system. Decreasing the position and the force gain improves the stability, however, sense of reality decreases. Considering the results, some suggestions are made to improve the stability and future works.

SANAL STAUBLI RX160 MANİPÜLATÖRÜN PHANTOM PREMIUM HAPTIC CİHAZ İLE KONTROLÜ

ÖZET

Robotlar çeşitli amaçlarla çok farklı uygulama alanlarında kullanılabilmektedirler. Bu nedenle robotlar için çok çeşitli görevler tanımlanabilir. Bu karmaşık görevleri yerine getirebilmek için, robotların akıllı sistemler olmaları gerekmektedir. Robotları daha fazla akıllı hale getirebilmek sürekli olarak geliştirilebilir bir alandır ve bunu gerçekleştirmenin etkin yollarından biri robot algılarını geliştirmektir. Bu bağlamda, robot algılarının robotik bir uygulamaya entegre edilmesi bu tezin amaçlarından birisidir.

Haptics dokunsal ve/veya kinestetik duyuların insan-bilgisayar etkileşimine dahil edilmesi bilimidir. Hem ticari hem de bilimsel araştırmalarda geniş bir yelpazede uygulamalara sahiptir. Bu uygulama alanlarından bazıları şunlardır: Sanal gerçeklik, robotik kontrol, medikal, ameliyat ve diş ile ilgili simülasyonlar gibi eğitim simülasyonları, sanal montaj, çarpışma algılama, moleküler modelleme, SEM (Sonlu Elemanlar Metodu) uygulamaları, nano manipülasyon, eğlence ve oyun, nükleer ve tehlikeli uygulamalar için uzaktan manipülasyon ve 3 boyutlu modelleme.

Bu tezde, Phantom Premium 1.5 High Force 6 DOF haptic cihaz kullanılarak Staubli RX160 manipülatörün 3 serbestlik dereceli sanal modeli ile etkileşim sağlayabilen bir yazılım geliştirilmiştir. Kullanıcı haptic arayüz noktasını hareketlendirerek sistem için pozisyon, hız ve ivme girdisi oluşturmaktadır. Geliştirilen algoritma ile, bu girdiler kullanarak kullanıcıya geri besleme olarak kuvvet ve sanal robot görsel hareketi uygulanmaktadır. Kuvvet geri beslemesi Staubli manipülatörün oluşturulan dinamik modeli aracılığı ile hesaplanmaktadır. Görsel geri besleme ise 3 serbestlik dereceli Staubli RX160 için geliştirilen hareket algoritması tarafından heaplanmaktadır.

Geliştirilen yazılım açık kaynak kodlu bir yazılımdır ve konu ile ilgili yapılacak yeni çalışmalara entegre edilebilmesine imkan sağlamaktadır. Gerçek robot davranışlarının ilk aşama olarak, sanal ortamda simüle edilmesi, gerçek robotlar ile yapılan deneyler esnasında oluşabilecek hasarların test ve algoritma geliştirme aşamalarında fark edilip önlenmesi açısından önem taşımaktadır. Ayrıca, ilk aşama olarak uygulamaların sanal ortamda geliştirilmesi harcanacak enerji ve zaman açılarından da tasarruf sağlamaktadır. Bu doğrultuda Staubli manipülatör ve Phantom haptic cihaz ile ilgili yapılacak çalışmaların sanal ortamda simüle edilmesi, bu tezin hedeflerinden birini oluşturmaktadır.

Programlama aşamasında, C/C++ programlama dili temeline dayanan OpenHaptics yazılım geliştirme kiti kullanılmıştır. OpenHaptics QuickHaptics micro API, HLAPI ve HDAPI olmak üzere 3 UPA'ya (Uygulama Programlama Arayüzü) sahiptir. QuickHaptics micro API, az sayıda kod ile hızlı bir şekilde temel uygulamaların geliştirilmesinde kolaylıklar sağlamaktadır. HLAPI, grafiksel olarak ileri

uygulamalar geliştirilmesinde avantajlar sağlamaktadır. HDAPI ise doğrudan motor ve enkoderlerin kullanılması ve direk kuvvet ve tork yüklemeleri gibi kontrol algoritmaları ile çalışılacak ileri seviyede çalışmalarda avantajlar sağlamaktadır. Bu çalışmada, her bir UPA'nın avantajlarından faydalanmak amacı ile 3 UPA da kullanılmıştır. Sanal robot hareketlerinin kullanıcıya görsel geri besleme olarak sağlanması için bir görsel arayüz tasarlanmıştır. Tasarlanan görsel arayüz, içerisinde Staubli RX160 manipülatörün katı modelinin konumlandığı sanal bir ortamı göstermektedir. Oluşturulan sanal ortam ve görsel arayüz OpenGL UPA temeline dayanan OpenHaptics komutları kullanılarak geliştirilmiştir.

Uygulamada, kullanıcı haptic arayüz noktasını hareket ettirmekte ve bir pozisyon değişimi oluşmaktadır. Gerçekleşen bu pozisyon değişimi ile, zaman dikkate alınarak doğrusal hız ve ivme değerleri hesaplanmaktadır. Daha sonra, bu pozisyon, hız ve ivme değerleri kullanılarak haptic rendering algoritması 3 serbestlik dereceli sanal Staubli manipülatörün dinamik modelinin sonuçlanan kuvvetlerini hesaplamakta ve grafik rendering algoritması sanal Staubli manipülatörün sanal ortamda sonuçlanan hareketini hesaplamaktadır.

Staubli RX 160 manipülatörün gerçek boyutlu katı modelleri Staubli tarafından .stp dosya uzantılı olarak sağlanmıştır. Sanal uzaya bu parçalar .3ds dosya formatına çevrilerek alınmıştır. Sanal uzayda uzuvlar birbirinden bağımsız parçalar olarak konumlanmakta ve hareket etmektedirler. 3 serbestlik dereceli sanal Staubli manipülatör uzuvlarının sanal ortamdaki koordine hareketinin elde edilmesi için robot uzuvları ile bir hiyerarşik model oluşturulmuştur. Daha sonra, sanal modelin sanal uzayda hareketi için, oluşturulan hiyerarşiyi dikkate alarak, her bir uzvun ve eklem koordinat sistemlerinin dönmelerini yinelemeli olarak hesaplayan ve uzuv katı modellerini hareketlendiren bir hareket algoritması geliştirilmiştir. Bu algoritma, sanal Staubli manipülatör uzuvlarının sanal ortamdaki konum ve eksen bilgilerini elde etmek için 3 serbestlik dereceli Staubli manipülatörün çözülen geometrik modelini ve katı modellerin hareketi için OpenHaptics komutlarını kullanmaktadır.

İnsan için dokunsal algı frekansı 1000 Hz ve görsel algı frekansı 30 Hz civarındadır. Bu nedenle, kullanıcıda oluşan gerçeklik hissinin sağlanması için, haptic rendering algoritması çalışma frekansı 1000 Hz, grafik rendering algoritması çalışma frekansı 30 Hz olarak belirlenmiştir. Geliştirilen yazılımda, bu iki algoritma OpenHaptics UPA'larından faydalanılarak farklı iş parçacıklarında, parelel olarak koşacak şekilde programlanmıştır. Böylece, kullanıcıya uygulanacak kuvvet geri beslemeleri saniyede 1000 defa hesaplanarak kullanıcıya uygulanmaktadır. Sanal modelin hareketi ise saniyede 30 defa hesaplanmakta ve arayüzde saniyede 30 kare yenilenmektedir. Uygulanan bu iki ayrı frekans sayesinde kullanıcı, aslında ayrık zamanda gerçekleştirilen uygulamayı sürekli olarak hissediyor olmaktadır.

Staubli manipulator için, haptic device tarafından konum kontrolünün sağlanacağı bir sanal uç işlevci ekipmanı tasarlanmış ve sanal ortamda Staubli manipülatörün flanşına eklenmiştir. Staubli RX160 manipülatörün ilk 3 serbestlik derecesi robotun çalışma uzayının büyük kısmına erişimini ve son 3 serbestlik derecesi uç işlevcinin yönelimini sağlamaktadır. Bu çalışmada, Staubli RX160 manipülatörün ilk 3 serbestlik derecesi modellenmiştir. Uygulamada Staubli manipulatörün son 4 uzvu ve tasarlanan uç ekipman tek bir uzuv gibi hareket etmektedir ve böylece yönelim açıları devre dışı bırakılmıştır. Staubli RX160 manipülatör için geometrik, ters geometrik, kinematik, ters kinematik, statik, dinamik ve ters dinamik modelleri çözülmüştür. Staubli RX160 manipülatörün ikinci uzvu yay denge sistemi ile donatılmıştır ve bu model Staubli tarafından sağlanmıştır. Dinamik ve ters dinamik model bu yay modelini ve eklem sürtünme modellerini içermektedir.

Haptic arayüz noktasının pozisyon değişimi belirli bir bir katsayı ile ölçeklendirilerek Staubli manipülatörün uç işlevcisinin pozisyon değişimine dönüştürülmektedir. Daha sonra, 3 serbestlik dereceli sanal Staubli manipülatör dinamik modelinin sonuçlanan kuvvetleri haptic cihaz kuvvet limitleri içerisinde bir aralığa ölçeklenmektedir. Pozisyon ve kuvvet ölçeklendirmeleri uniform olarak, diğer bir deyişle her eksen için aynı oranda gerçekleştirilmiştir. Uygulanan pozisyon ölçeği katsayısı pozisyon kazancını ve kuvvet ölçeği katsayısı kuvvet kazancını ifade etmektedir.

Deney aşamasında, uygulama için şartlar belirlenmiştir ve deneyler bu koşullar altında gerçekleştirilmiştir. Bu koşullar farklı hareket yörüngeleri, pozisyon ve kuvvet kazançlarından oluşmaktadır. Deney süresince 3 serbestlik dereceli sanal Staubli manipülatörün uç işlevcisi haptic arayüz noktasını takip etmekte, sonuçlanan görsel geri beslemeler ve kuvvet geri beslemeleri kullanıcıya uygulanmaktadır ve geliştirilen yazılım pozisyon, hız, ivme ve kuvvet bilgilerini kayıt altına almaktadır. Uygulama sonlandırıldığında yazılım bu bilgileri bir dosyaya kaydetmektedir.

Yazılımın geliştirilmesinin ardından bu tezde incelenecek deney şartları belirlenmiştir. Öncelikle iki farklı hareket yörüngesi kurgulanmıştır. İlk yörüngede haptic arayüz noktasının x,y ve z eksenlerinde sıralı olarak hareket ettirilmesi planlanmıştır. İkinci yörüngede ise haptic arayüz noktasının x, y ve z eksenlerinde aynı anda hareketini sağlayacak dairesel bir yörünge planlanmıştır. Diğer deney şartları kuvvet ve pozisyon kazançlarının farklı değerleri için deneylerin tekrarlanmasını kapsamaktadır.

Yörüngelerin tasarlanmasından sonra, bu çalışma için uygun olacak şekilde iki adet pozisyon kazancı ve üç adet kuvvet kazancı belirlenmiştir. Tasarlanan her bir yörünge için, iki pozisyon kazancı ve her bir pozisyon kazancı için üç kuvvet kazancı deney şartlarını oluşturmaktadır. Böylece, bu şartlar altında 12 deney gerçekleştirilmiştir.

Belirlenen şartlar için deneyler gerçekleştirilmiş ve sonuçlanan pozisyon, hız, ivme ve kuvvet değerleri zamana bağlı olarak çizdirilerek, grafikler elde edilmiştir. Deney esnasında oluşan pozisyon, hız, ivme ve kuvvet bilgilerinin, x, y ve z eksenlerindeki etkilerinin detaylı olarak incelenebilmesi için grafiklerde, bu bilgilerin x, y ve z eksenlerine izdüşümleri çizdirilmiştir. Aynı sonuçların büyüklük eğrileri ise eklerde verilmiştir.

Sonuç olarak, hedeflenen yazılım geliştirilmiş ve çeşitli yörünge, pozisyon ve kuvvet kazançları için deneyler gerçekleştirilmiştir. Daha sonra, uygulanan deney koşulları için sistemin kararlılığı ve gerçeklik hissi incelenmiştir. Sayısal olarak hesaplanan türevlerin ve diğer sayısal hesaplamaların sonuçlarda gürültü ve düzensizliklere neden olduğu gözlemlenmiştir. Genel olarak, pozisyon ve kuvvet kazançlarının azaltılmasının kararlılığı iyileştirdiği fakat kullanıcı tarafından algılanan gerçeklik hissini düşürdüğü, hem sonuçlanan grafiklerden hem de haptic cihaz tarafından kullanıcıya uygulanan kuvvet geri beslemelerinden saptanmıştır. Sonuçları dikkate alarak, sistem kararlılığının iyileştirilmesi ve kuvvet düzensizliklerinin azaltılması amacıyla, çeşitli sinyal işleme filtrelerinin ve yapay zeka algoritmalarının sisteme uygulanması gibi bu çalışmanın geliştirilmesine ve ileriki çalışmalara yönelik öneriler sunulmuştur.

1. INTRODUCTION

A broad range of tasks can be defined for robots. Therefore, making the robots more intelligent is a continuously developable area. One of the effective ways for this purpose is improving the sensations of the robots. In this context, integrating the tactile unilateral or bilateral force/torque feedback to the robotic applications is the one of the state of the art in robotic researches.

Developing the robotic applications in a virtual environment is a safety and fast method to test the applications. These kinds of virtual reality applications can prevent the possible damages in the developing or the testing stage.

The tactile perception of the robots is the objective of this study. A master-slave robot system is used. The master robot is the Phantom Premium 1.5 6DOF /1.5HF 6DOF haptic device and the slave robot is the virtual model of the Staubli RX160 manipulator. The stylus of the haptic device controls the end-effector position of the virtual model of the Staubli manipulator. The haptic device provides position, velocity and acceleration informations for the virtual Staubli manipulator. Then, by calculating the resulting forces of the derived dynamic model of the 3 DOF Staubli RX160 manipulator, the haptic device applies the force feedback toward the human operator.

12 experiments are realized for two different paths and diverse position and force gains. After 12 experiments are completed, position, velocity, acceleration and computed force values are obtained. Then, the stability of the system is investigated for these experiments.

1.1 Purpose of Thesis

The purpose of this thesis is to visualize the motions of the Staubli manipulator and feel the resulting forces at the end-effector of the Staubli manipulator controlling the end-effector by the haptic device. A software application developed in this context can be used to simulate the manipulator behaviours in the virtual environment. Thus, the application allows to analyze and avoid the unexpected behaviors before run the applications on the real manipulator. The software developed in the scope of this study can be also used for further researches about the Staubli manipulator and the haptic device.

1.2 Literature Review

In the literature, there is a wide range of applications including the communication between Staubli manipulators and Phantom haptic devices. Some of these applications are stated below.

In reference [1], an open robotic platform based on Staubli RX60 manipulator is introduced, different possibilities of software configurations explaining the advantage and drawbacks of these software configurations are described and experimental results are given to validate the performance of the robotic setup. The experimental setup consists of a Staubli RX60 manipulator, a CS8 controller, a 6 DOF ATI wrist force-torque sensor, a 3 DOF capacitive accelerometer, a 3 DOF gyroscope, an acquisition board integrated to the robot controller and a 6 DOF Phantom haptic device.

In reference [2], bilateral haptic guided robot teleoperation using Internet Protocol version 6 (IPv6) with high Quality of Servis (QoS) is achieved. The Phantom 1.5 6 DOF haptic device, Staubli TX90, CS8 controller and JR3 force-torque sensor are used.

In reference [3], eliminating and scaling the non-contact forces and torques from the measurements of a wrist-mounted force-torque sensor are studied. The Phantom Premium 1.5 HF 6 DOF haptic device, either Staubli RX60 or RX90, CS7B controller and a combined force-torque sensor are used.

In reference [4], path planning techniques based on harmonic functions are used to generate force feedbacks for teleoperated assembly tasks. A phantom haptic device and a Staubli TX90 manipulator are used. The cross-platform tools Qt (as application frame-work), Coin3D (as graphics toolkit) and OpenHaptics Toolkit are used for programming. The communication between the haptic device and the Staubli manipulator is achieved via internet protocol.

In reference [5], a teleoperation approach using a virtual spring and local contact force control on the slave robot is introduced. A Phantom 1.0A haptic device and PUMA560 manipulator are used as a master and a slave device respectively. LINUX is used for operating system. The servo rate is about 10 000 Hz and wireless LAN network used for communication.

1.3 Staubli RX160

The Staubli RX160 is a manipulator having 6 DOF and used in both industrial and academic applications. The first 3 DOF provides the major part of the end-effector position and the last 3 DOF provides the orientation of the end-effector.

The 6 basic components and 6DOF of the Staubli RX160 are shown in Figure 1.1.

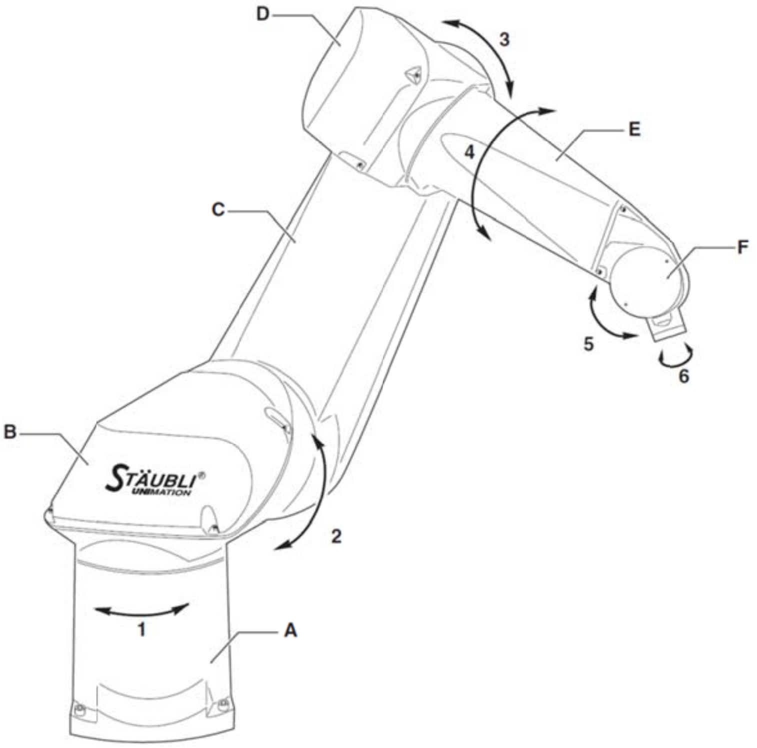


Figure 1.1 : The Staubli RX160[7].

where A, B, C, D, E and F are the base, the shoulder, the arm, the elbow, the forearm and the wrist respectively. 1, 2, 3, 4, 5, 6 indicate the revolute joints and their rotational directions of the robot.

The physical dimensions of the Staubli RX160 are shown in Figure 1.2. All the dimensions are given in mm scale.

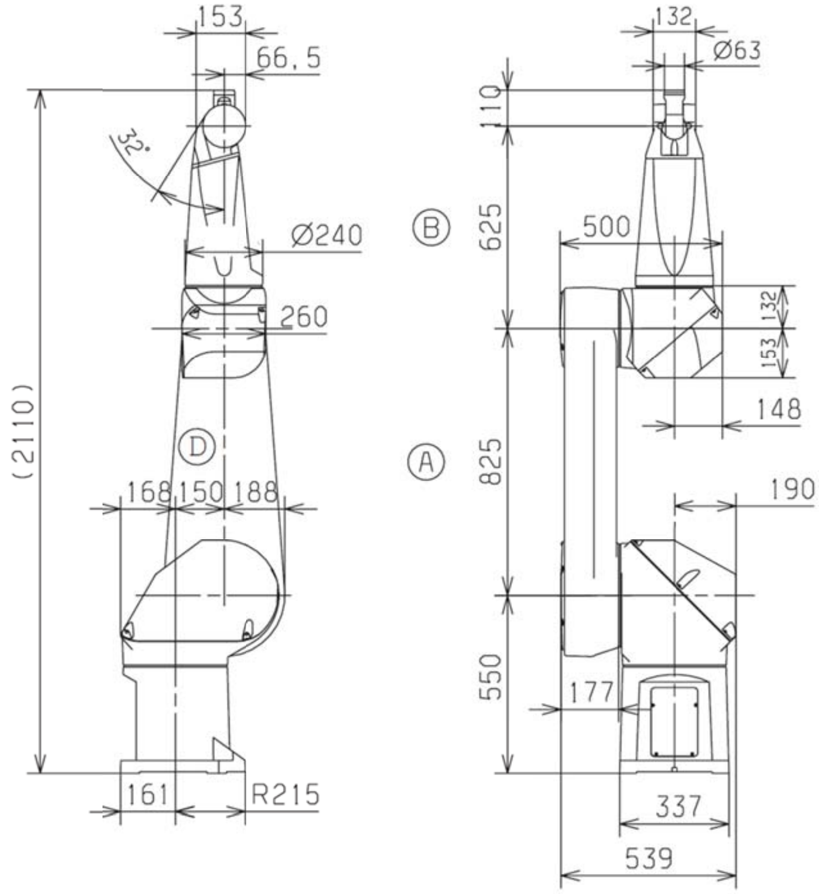


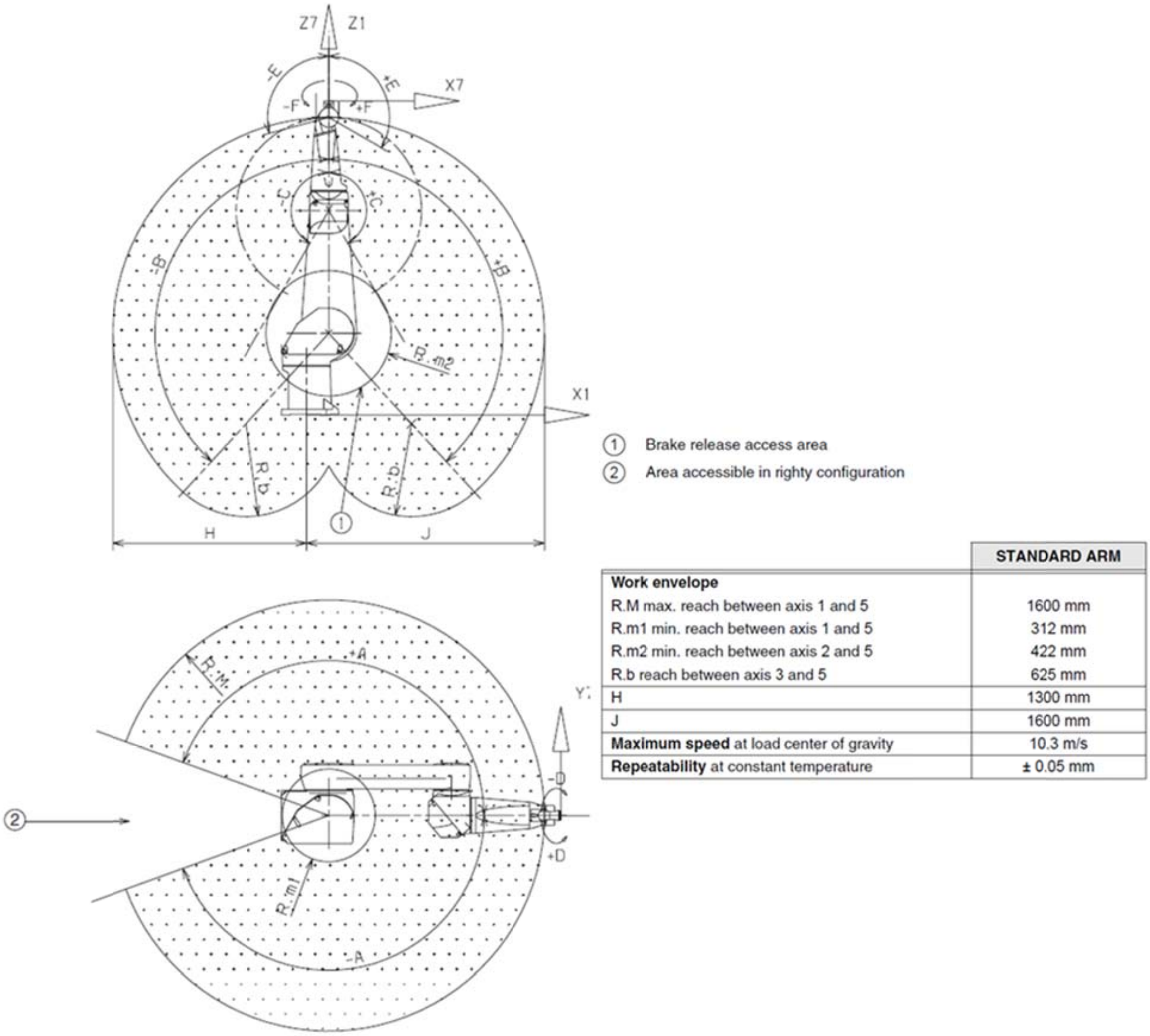
Figure 1.2 : The physical dimension of the Staubli RX160[7].

Each joint has a brushless motor to obtain the motion and all the motors have a parking brake system. The total arm weight is 248 kg and the maximum load capacity is 20 kg at the nominal speed. Amplitude, speed and resolution parameters of the Staubli RX160 are given in Table 1.1.

Table 1.1 : Amplitude, speed and resolution parameters of the Staubli RX160[7].

Axis	1	2	3	4	5	6
Amplitude (°)	320	275	300	540	225	540
Working Range Distribution (°)	±160	±137.5	±150	±270	+120 -105	±270
Nominal Speed (°/s)	165	150	190	295	260	440
Maximum Speed (°/s)	200	200	255	315	390	870
Angular Resolution ($°.10^{-3}$)	0.042	0.042	0.054	0.062	0.12	0.17

Work envelope is the range of the movements of the robot and the robot is able to move inside this area. Figure 1.3 illustrates the work envelope of the Staubli RX160.


Z7
Z1
X7
-E
+E
-F
+F
-C
+C
-B
+B
R.m2
X1
R.b
R.b
①
H
J
① Brake release access area
② Area accessible in righty configuration
R.M
+A
-A
Y1
-D
+D
②
R.m1

	STANDARD ARM
Work envelope	
R.M max. reach between axis 1 and 5	1600 mm
R.m1 min. reach between axis 1 and 5	312 mm
R.m2 min. reach between axis 2 and 5	422 mm
R.b reach between axis 3 and 5	625 mm
H	1300 mm
J	1600 mm
Maximum speed at load center of gravity	10.3 m/s
Repeatability at constant temperature	± 0.05 mm

Figure 1.3 : The work envelope of the Staubli RX160[7].

As an end-effector various tools can be mounted to the tool flange. Thus, the work envelope changes depending on the tool position.

There is a control unit to communicate with the robot and a hand control device to control it manually. Additional equipments can be selected for diverse purposes.

1.4 Phantom Premium 1.5 High Force 6DOF

The SensAble Technologies provides Phantom haptic device products. Properties like workspace, type of the communication, force/torque/velocity/acceleration limits vary depending on the Phantom haptic device that is chosen. The Phantom haptic device products are used for various puposes in haptics.

The Phantom Premium 1.5 High Force 6 DOF haptic device is shown in Figure 1.3. It has a high fidelity force feedback and senses motion in 6 degrees of freedom. The first 3 degrees of freedom are for positional sensing and the last 3 degrees of freedom are for orientational sensing.

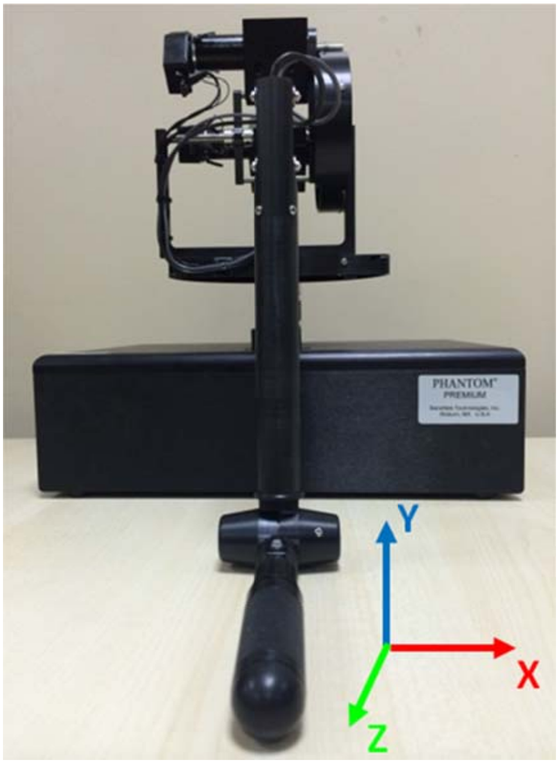


Figure 1.4 : The work envelope of the Staubli RX160.

The both software and hardware requirements of the Phantom Premium 1.5 High Force 6DOF haptic device are listed below[9]:

- An Intel® based personel computer (A minimum of Pentium® II class processor is recommended).
- IEEE 1284 EPP compliant parallel port interface; a Phantom Communication Converter (PCC) and FireWire Card (requires IEEE-1394a-2000 compliant FireWire Port).
- The most current version of the Phantom Device Driver (PDD).

Power specifications of the Phantom Premium 1.5 High Force 6DOF haptic device is given in Table 1.2.

Table 1.2 : Power Specifications[9].

	With Internal Auto-Switching Power Supply
Output Current Rating	8.3 A
AC Input	100~120 VAC 200~240 VAC (auto-switching power supply)
Input Frequency	50/60 Hz
AC Current	5 A / 115 V, 2.5 A / 230 V
Inrush Current	15 A / 115 V, 30 A / 230 V

The Phantom Premium 1.5 High Force 6DOF haptic device is controlled through the parallel port. To enable the communication, PDD must be installed on the PC.

Power supply and parellel port socket of the Phantom Premium 1.5 HF 6 DOF haptic device are shown in Figure 1.5. The two switches in the Figure 1.5 are the gimbal and base motor switches.

In some applications, the motors need to be deactivated for different purposes. These switches can be used to cut off the powers of the motors by turning of them. When the switches are turned off, the motor powers are cut off, however, the encoders continue to be powered.

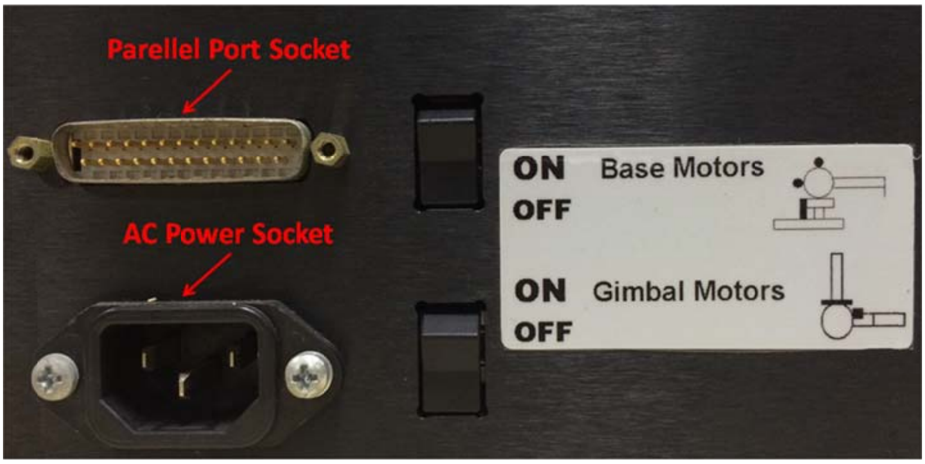
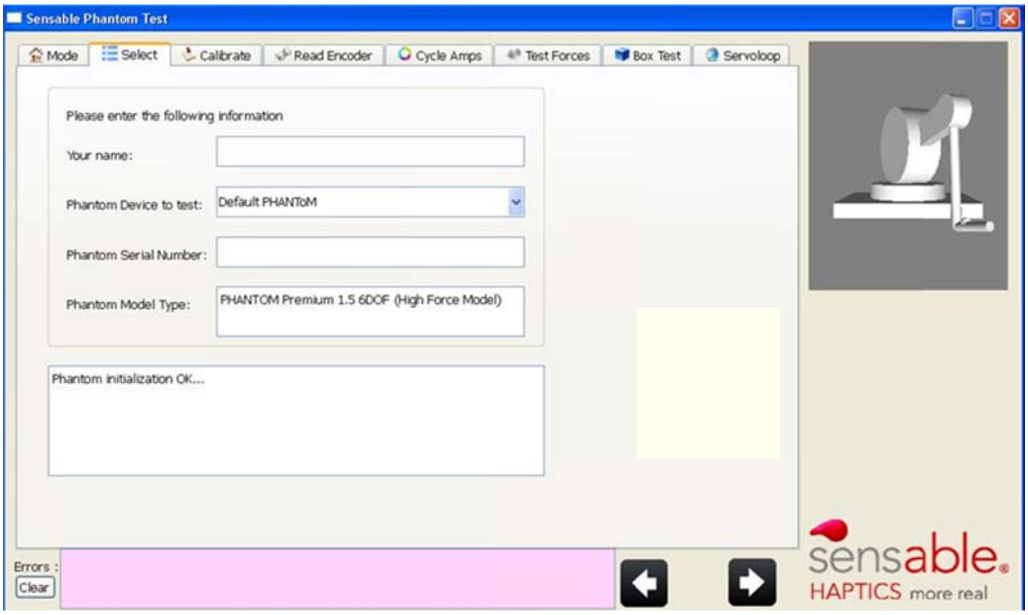


Figure 1.5 : Sockets and switches of the Phantom Premium 1.5 HF 6 DOF.

After completing the required installations and cable connections, the Phantom Test software can be used to verify the installations and connections.

The Phantom Test software also provides a calibration interface and some simple applications to check the communication between the PC and the haptic device. The interface of the Phantom Test software is shown in Figure 1.6.

The device specifications of the Phantom Premium 1.5 High Force 6DOF are given in Table 1.3.



Please enter the following information

Your name:

Phantom Device to test: Default PHANToM

Phantom Serial Number:

Phantom Model Type: PHANTOM Premium 1.5 6DOF (High Force Model)

Phantom initialization OK...

Figure 1.6 : Phantom Test Interface.

Table 1.3 : Device Specifications[9].

	Phantom Premium 1.5 High Force 6DOF

Workspace	Translational= 381 W x 267 H x 191 D mm
	Rotational= Yaw: 297 degrees
	Pitch: 260 degrees
	Roll: 335 degrees

Footprint	330 W x 254 D mm
Range of Motion	Lower arm pivoting at elbow
Nominal Position Resolution	0.0007 mm Rotational= Yaw&Pitch: 0.0023 degrees Roll: 0.008 degrees
Backdrive Friction	0.2 N
Maximum exertable force (nominal position)	37.5 N Rotational= Yaw&Pitch: 188 mNm Roll: 170 mNm
Continuous exertable force (nominal position)	6.2 N Rotational= Yaw&Pitch: 515 mNm Roll: 48 mNm
Stiffness	3.5 N mm^{-1}
Inertia (apperant mass at tip) -without encoder gimbal	<159 g
Force Feedback	x, y, z
Position Sensing	x, y, z (roll, pitch, yaw upon special request)
Interface	Parellel Port
Supported Platforms	Intel-based PCs
GHOST® SDK Compatibility	Upon special request
OpenHaptics™ Toolkit Compatibility	Yes

The workspace of the Phantom 1.5 High Force 6DOF haptic device is physically limited by using mechanical stops. The device must be operated in this workspace.

GHOST® SDK and OpenHaptics™ Toolkit are two Software Development Kits, which are generated to use in haptics applications.

2. HAPTICS

Haptics term is derived from Greek word “haptesthai” and means “pertaining to the sense of touch”. The haptics is the science of incorporating the tactile and/or kinesthetic sensations into the human-computer interaction.

Haptic interfaces are the haptic devices which stimulate the sense of the interaction between the user and the virtual environment. Haptic rendering algorithms compute the force feedback toward the user for interacting with the virtual objects and the haptic interface applies the computed force feedback to the user.

In 1970s and 1980s, significant research studies in robotics began to focus on manipulation and perception by touch and initially researches concerned with building autonomous robots[11].

In the early 1990s, haptic applications were begun to be combined with the computer graphics technology. Knoll demonstrated haptic interaction with simple virtual objects in 1960s. However, haptic interactions with complex computer-simulated objects were possible in the early 2000s and are still developing.

The spectrum and trend for Hapto-Audio-Virtual-Environment (HAVE) applications is shown in Figure 2.1.

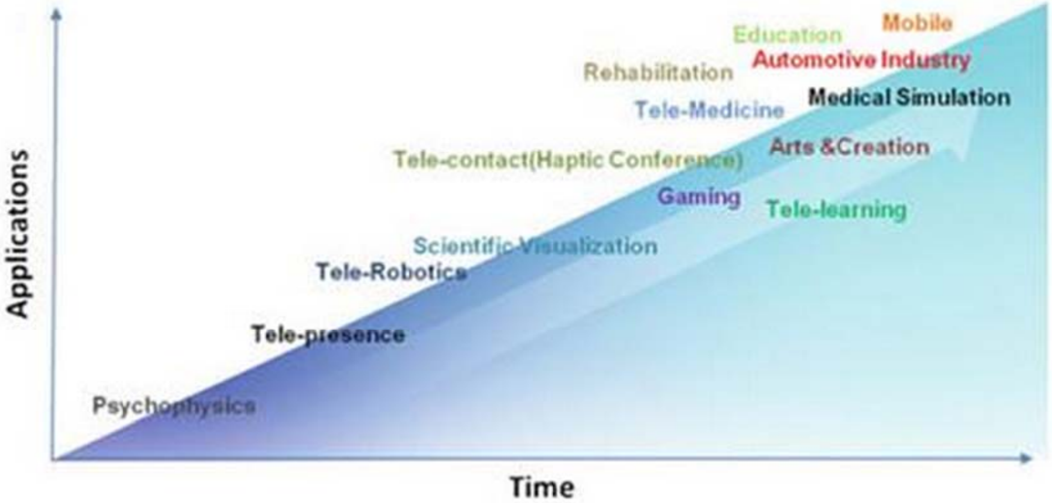
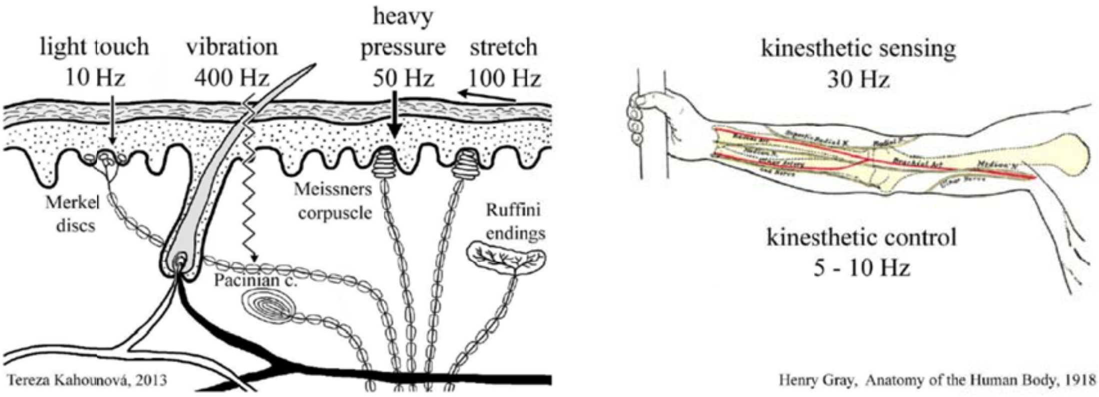


Figure 2.1 : The spectrum and trend for HAVE[11].

The haptics has a broad range of application areas and multidisciplinary researches can be achieved by combining different disciplines. In reference [6], the haptics is divided into three areas: human haptics, machine haptics and computer haptics.

Human haptics: For developing the optimal haptic applications, understanding the mechanical, sensory, motor and cognitive subsystems of human tactile perception is important. Therefore, the human hapics concerns with these topics.

Generally, the haptic perception is divided into two sensations: the tactile sensing and the kinesthetic sensing. The tactile sensing is achieved by the cultaneous mechanoreceptors such as Meissner corpuscle, Pacinian corpuscle, Merkel disks and ruffini endings. Different cultaneous mechanoreceptors have their own sensing bandwidth. The kinesthetic sensing provides the information about position and movement of the parts of the body such as the muscles, tendons and joints. The kinesthetic sensing bandwidth and the kinesthetic control bandwidth are aproximately 30 Hz and 5-10 Hz respectively[12]. Different types of the cutaneous mechanoreceptors with their own approximate sensing frequency are shown on the left side and the kinesthetic sensing and the kinesthetic control bandwidth are shown on the right side in Figure 2.2.



Tereza Kahounová, 2013

Henry Gray, Anatomy of the Human Body, 1918

Figure 2.2 : Approximate sensing frequencies of the cultaneous mechanoreceptors, kinesthetic and the range of the kinesthetic control[12].

Machine haptics: The machine haptics concerns with design, construction and use of machines for replacing or scaling the human touch.

Computer haptics: The objective of the computer haptics is generating the algorithms using appropriate softwares for rendering the touch and feel of the virtual objects and the graphical interfaces.

2.1 Virtual Reality (VR)

Virtual reality or immersive virtual reality is a computer-generated media and enables the user to interact with the generated virtual environment. Combining human sensations with computer and machine skills can spread to many fields of industry and science. Therefore, the VR has a wide range of application areas.

Fundamental block diagram of a virtual reality application incorporating visual, auditory and haptic feedback is shown in Figure 2.3. The simulation engine computes the behavior of the virtual environment. The visual, auditory and haptic rendering algorithms compute the graphic, sound and force/torque feedback toward the human operator. Then, the human operator perceives video, audio and haptic feedback via visual displays, audio and haptic devices.

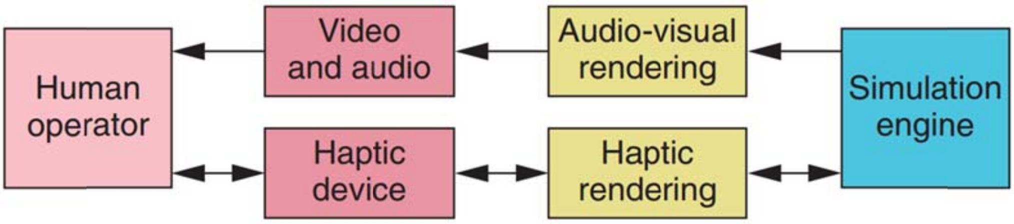


Figure 2.3 : Basic architecture for a virtual reality application incorporating visual, auditory and haptic feedback[11].

2.2 Haptic Rendering

The haptic rendering is the process of user interaction with the virtual environment. It allows the user to touch, feel and manipulate the virtual objects via the haptic interfaces.

The haptic rendering is splitted into three main blocks as previously proposed in [11]. These three blocks include collision-detection, force-response and control algorithms.

The collision-detection algorithms give the collisions and penetrations between a proxy and a virtual object by using position information.

The force-response algorithms compute the ideal interaction force between the proxy and the virtual object.

The control algorithms compute the closest force to the ideal interaction force considering the haptic interface device capabilities.

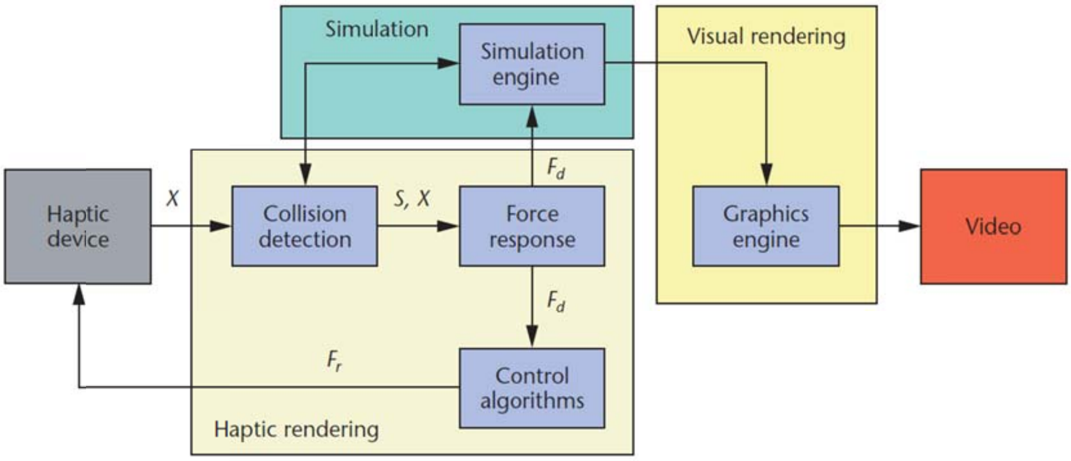


Figure 2.4 : Haptic rendering block diagram[11].

2.3 OpenHaptics SDK(Software Development Kit)

The OpenHaptics toolkit is a Software Development Kit including APIs, programming tools, PHANTOM® Device Drivers (PDD), documentation and source code examples. It is provided by SensAble technologies. OpenHaptics® Toolkit version 3.1 is used in this study. The APIs of the OpenHaptics Toolkit 3.1 are QuickHaptics micro API, Haptic Device API (HDAPI) and Haptic Library API (HLAPI).

The QuickHaptics micro API enables fast and easy programming through the wide range of default parameters. It requires minimal amount of codes to set up haptic/graphic scenes.

The HDAPI provides the low-level interaction with the haptic device and direct force/torque rendering.

The HLAPI provides high-level haptic rendering and enables the programmer to use it with the OpenGL® API. Haptic applications including advanced computer graphics processes can be achieved with the HLAPI.

The general scheme of an OpenHaptics application is shown in Figure 2.5. HHD is the haptic device handle and it needs to be initialized and configured by the HDAPI. HHLRC is the HL haptic rendering context. The HHLRC uses the HHD to interface with the haptic device. The graphics and the servo loop runs at 30 Hz and 1 kHz respectively.

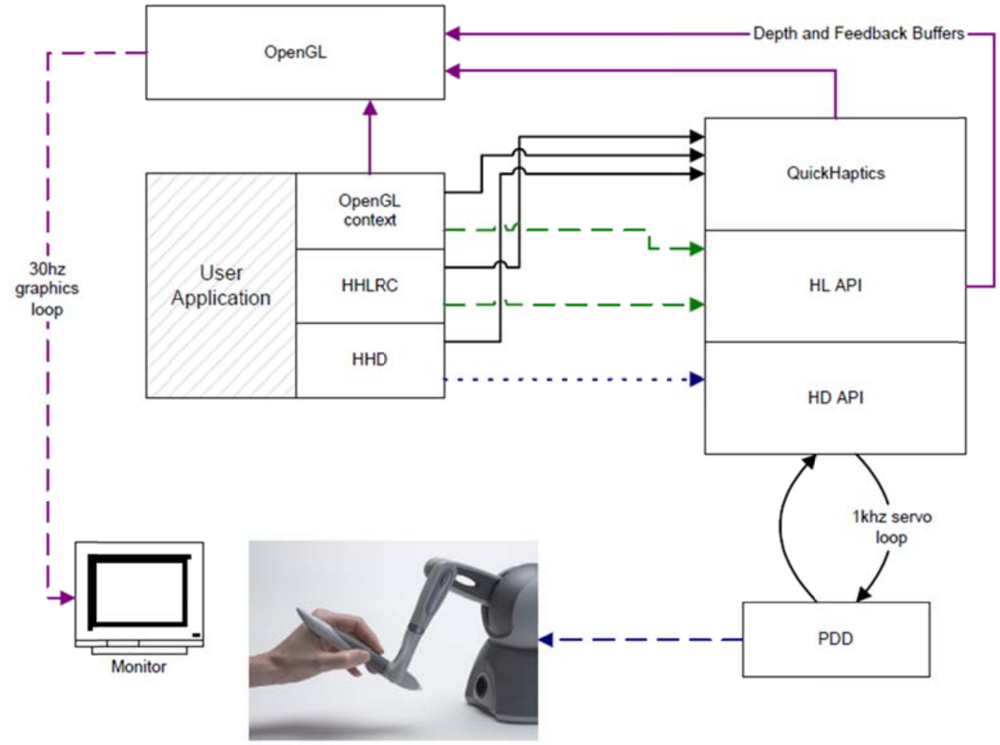


Figure 2.5 : The OpenHaptics overview[8].

2.3.1 OpenGL(Open Graphics Library) API

OpenGL is a cross-language and cross-platform application programming interface which produces 2D and 3D computer graphics.

The OpenGL rendering pipeline is given in Figure 2.6. The blue boxes are the programmable operations. Basically, vertex shader performs the vertex attributes from vertex arrays. These vertex attributes describe the boundaries of the primitives that are tessellated in the tesellation stage.

The geometry shader is a primitive assembly step and user-defined functions can be utilized to process the incoming primitives in this stage.

In the clipping stage, primitives which are located inside the viewing volume are splitted from outside primitives.

Rasterization refers the conversion of the geometric data to a sequence of fragments and this data is used to compute the final data for pixels in the output framebuffer.

Fragments are rectangular arrays and their color and depth properties are processed in the fragment shader stage. Finally, the interface window of the application displays the framebuffer.

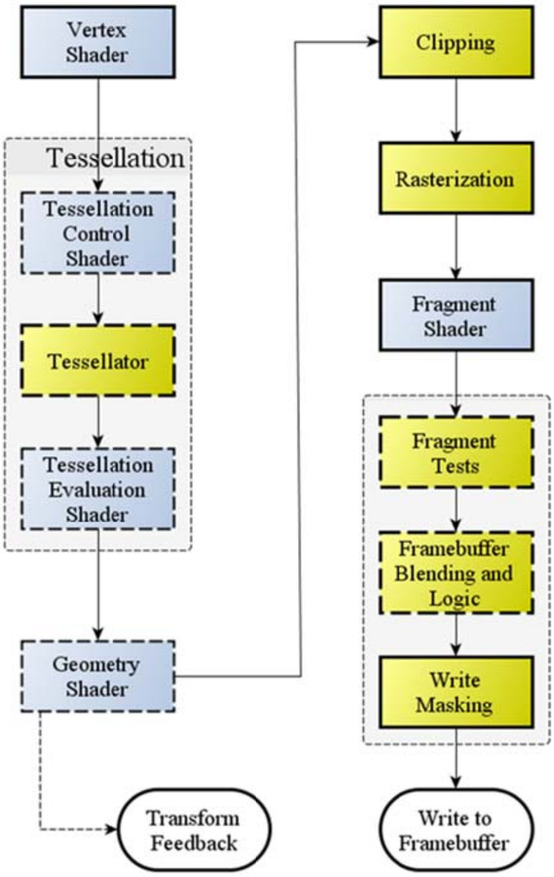


Figure 2.6 : Diagram of the OpenGL rendering pipeline[13].

2.3.2 QuickHaptics micro API

The QuickHaptics micro API is used with C/C++ programing language. The QuickHaptics micro API classes and properties are given in Appendix A. The fundamental four classes of the QuickHaptics micro API are DeviceSpace, QHRenderer, Shape and Cursor class.

The DeviceSpace defines force parameters and user interaction with the haptic workspace of a specific Phantom haptic device.

The QHRenderer class draws the scene both graphically and haptically. QHWin32 and QHGLUT are two classes inherited from the QHRenderer class and they are used to create windows. The QHWin32 and the QHGLUT are generated to use with Win 32 API and OpenGL Utility Toolkit.

In this study, the QHGLUT class is used to take the advantage of the utilities of the OpenGL. The Openhaptics SDK has its own commands to generate a 3D space. However, these commands are based on the OpenGL API.

The OpenHaptics uses the OpenGL to define the geometries and to implement the haptic rendering algorithms of the defined geometries. In the imaging stage, the projection of the OpenGL 3D world space is displayed on the computer screen. This image on the computer screen is a 2D representation of a 3D frustum.

The default camera location of the QuickHaptics is shown in Figure 2.7. Each shape in the world space has a bounding box that encloses itself. All of the bounding boxes generate a global bounding box that contains all of the shapes inside it.

The default viewing direction of the camera is along the z-axis and the default viewing angle is 45° as shown in the Figure 2.7.

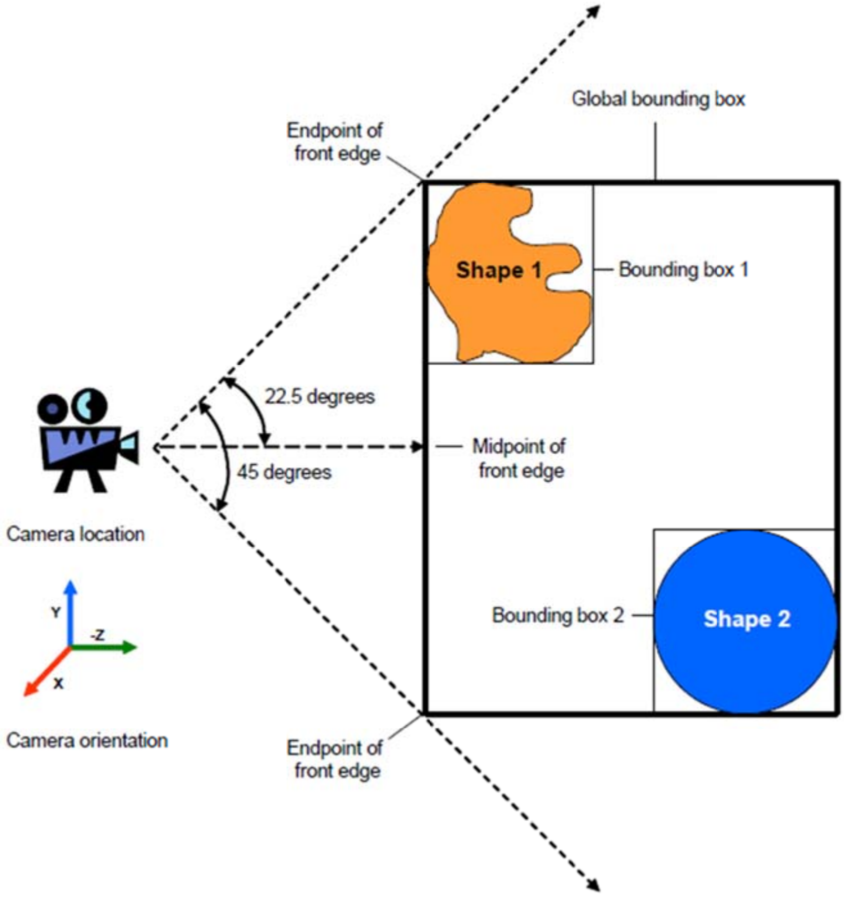


Figure 2.7 : Default QuickHaptics camera location[8].

The world space has two clipping planes. These planes are the near and far clipping plane and they are shown in Figure 2.8. Here, the camera sees only the volume between the near and the far clipping plane of the world space. Therefore, the near and the far plane must be set to encompass the virtual environment which is designed for the application.

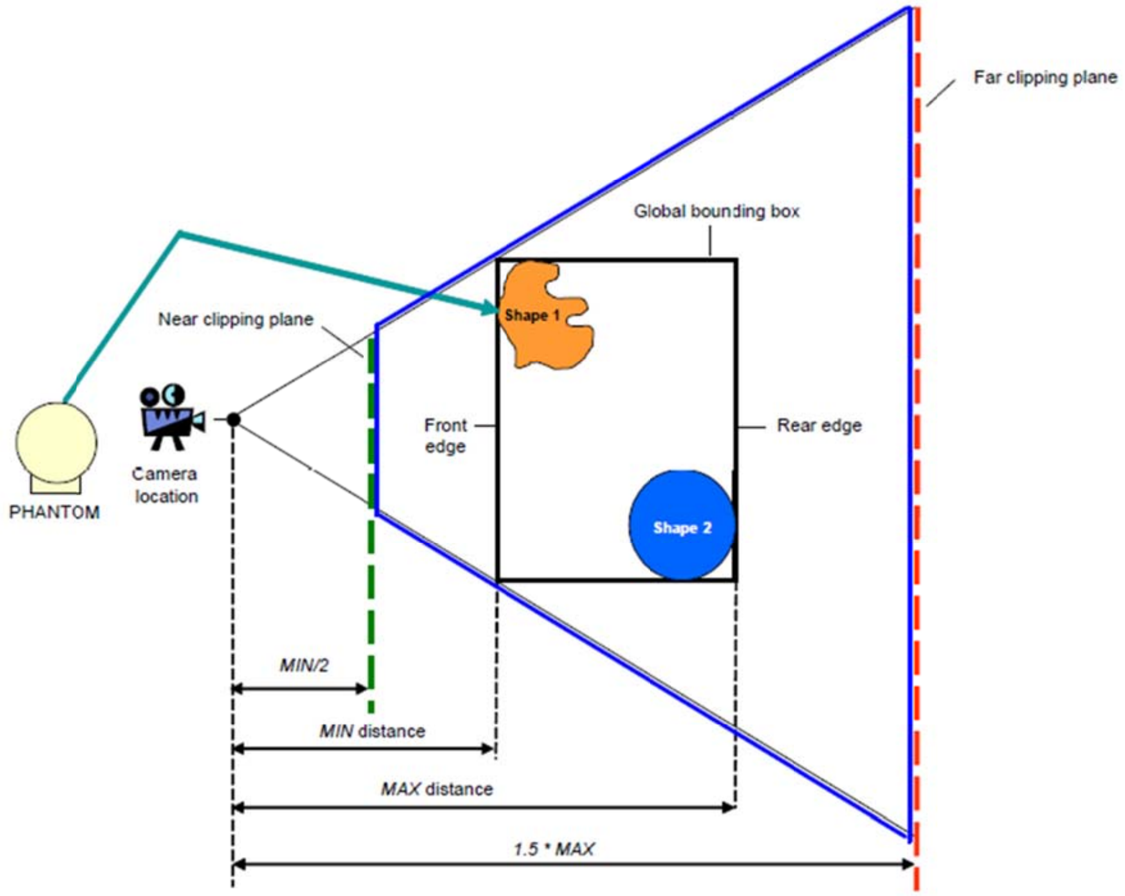


Figure 2.8 : Default clipping planes for world space[8].

The Shape class is the base class for the geometry operations. The inherited classes of the Shape class are Line, Plane, Cone, Cylinder, Sphere, Box, TriMesh and Text class. Properties like color, texture, position can be applied to these primitives by the shape class.

The TriMesh primitive is a 3D model and includes .stl, .obj, .3ds and .pyl file formats. These are the some of the standart 3D model file formats. A lot of 3D modeling softwares can produce these file formats. If another file format is used to design a 3D model, they can be converted to these file formats.

The Cursor class defines the interaction point in the world space and this is called “haptic interface point”[8]. The user controls the haptic interface point inside the workspace of the haptic device and the haptic interface point controls the proxy in the virtual environment.

The position information of the haptic interface point is obtained by using the OpenHaptics commands. The OpenHaptics reads the encoders of the motors and computes the position value.

The haptic interface point of the Phantom Premium 1.5 6DOF High Force model is shown in Figure 2.9. The proxy in the virtual environment is controlled by this haptic interface point.

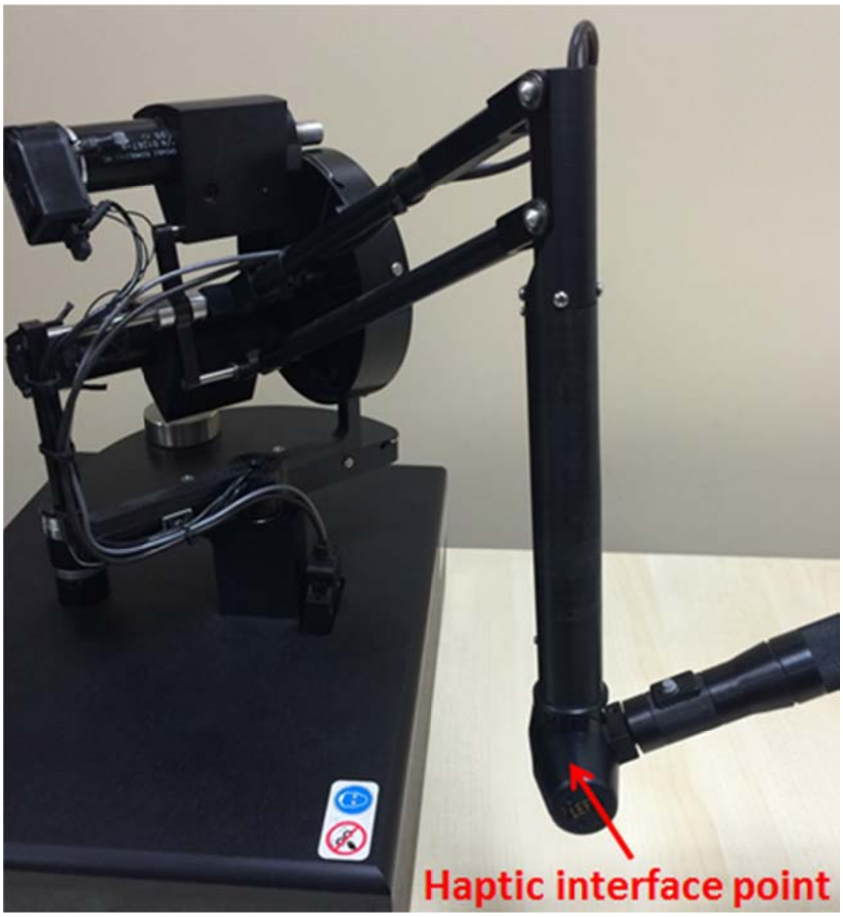


Figure 2.9 : Haptic interface point.

The default cursor is a blue cone and a TriMesh model can be used instead of it. Since, computing the cursor position in the world space requires information of the interaction with all of the components, it uses all of the classes of the QuickHaptics micro API.

The cursor class uses all the other classes to get the required information. The cursor class association is shown in Figure 2.10.

The location of the haptic interface point is obtained from the DeviceSpace class. The world space information (transformation from device space to the world space) is obtained from the QHRenderer class. The information of the virtual objects for cursor interaction is obtained from the shape class. The cursor information is also provided by the shape class. Thus, the cusor class gets the information from all other classes.

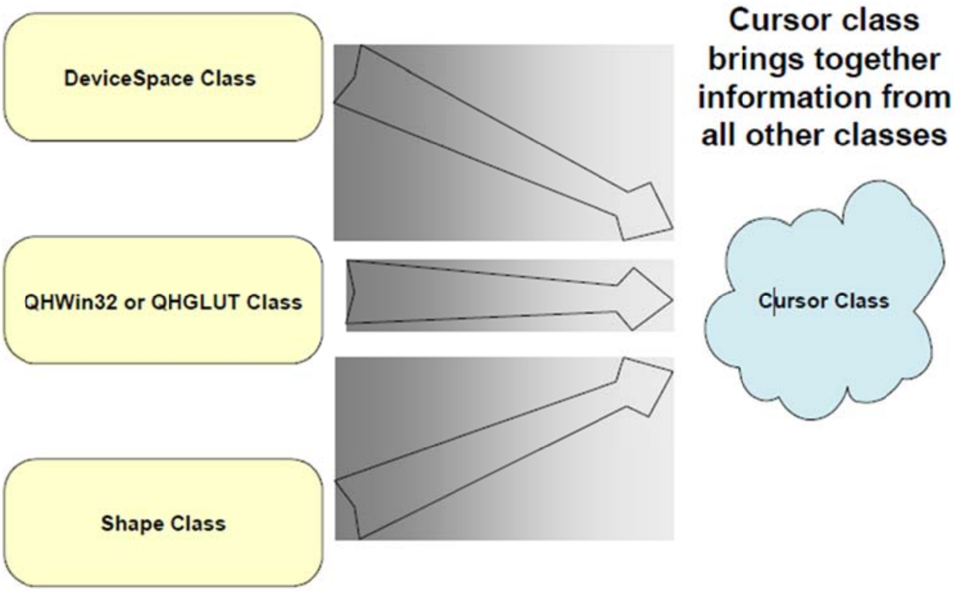

DeviceSpace Class
QHWin32 or QHGLUT Class
Shape Class
Cursor class brings together information from all other classes
Cursor Class

Figure 2.10 : The Cursor class association[8].

The QuickHaptics micro API program flow is shown in Figure 2.1. First, a window needs to be defined. Then, the flow order can be changed. If multiple haptic devices are used in a single application, the DeviceSpace must be defined for each device before the shapes are defined.

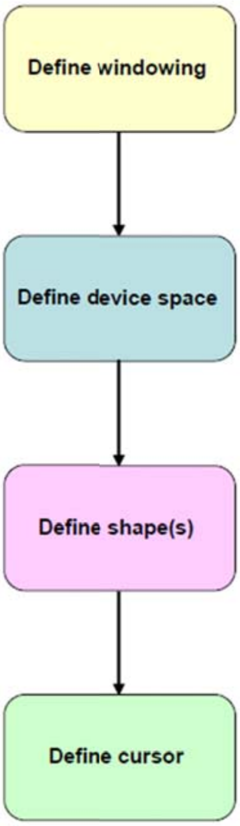


Figure 2.11 : QuickHaptics micro API program flow[8].

2.3.3 HDAPI

The HDAPI provides low-level haptic rendering. Control and haptics algorithms can be applied to the haptic applications with the HDAPI by using the motors and the encoders directly. Therefore, it has an important role in haptics researches.

The HDAPI can be used to query the device capabilities and parameters during the application.

The servo loop is a control loop for computing force algorithms and sending the forces to the haptic device. The servo loop can be adjusted by the HDAPI and needs 1 kHz or higher haptic update rate for a stable force feedback. To obtain this high update rate, the servo loop should be executed in a particular high-priority thread.

There are two fundamental components in the HDAPI involving the device and the scheduler. The device component provides the communication process with the haptic device. The device processes comprise device initialization, device safety and device state.

The device initialization is about the communication settings with the haptic device. The device safety includes the device protection routines and controls the safety parameters of the haptic device like maximum velocities, maximum forces and motor tempratures. The device state provides the settings and queries about the haptic device.

The scheduler enables the user to access the servo loop with routines. Thus, forces, which will be applied to the haptic device can be sent to the servo loop and the device state informations can be obtained.

A typical program flowchart of the HDAPI is given in Figure 2.12. The main operations of the HDAPI are realized in the servo loop.

A typical application starts by initializing the haptic device. Then, force output is enabled and the scheduler is started. Afterwards, the haptic frame is begun in servo loop. In the servo loop, the haptic rendering algorithms are run and then, the haptic frame is stopped. These haptic frames are repeated until the haptic rendering algorithm is stopped.

After the servo loop is stopped, finally, the scheduler is stopped and the haptic device is disabled.

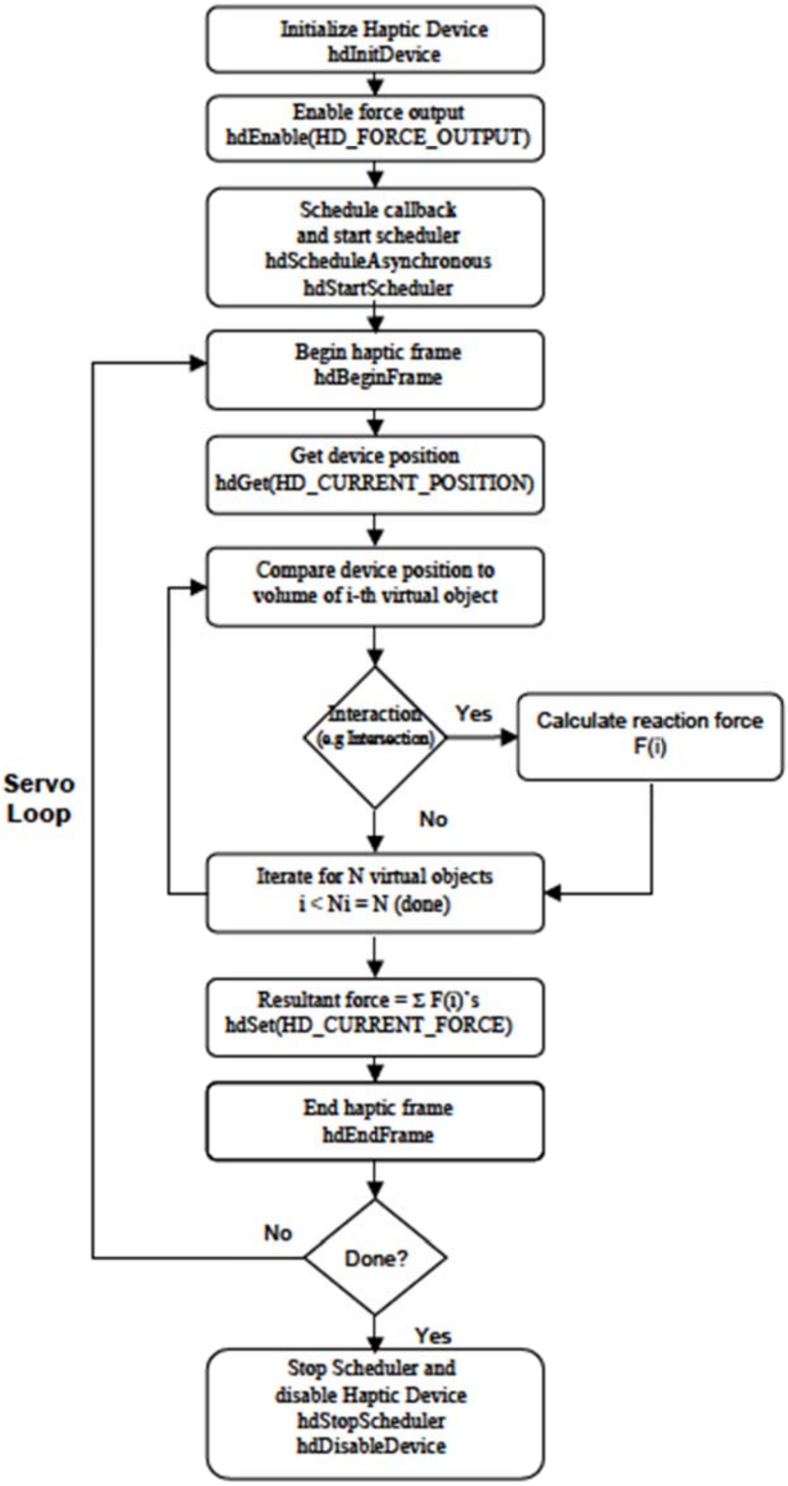


Figure 2.12 : HDAPI program flowchart[8].

2.3.4 HLAPI

The HLAPI is used for graphic rendering. It is a high-level API, which uses the OpenGL API. The HLAPI provides the graphic scene properties like background color and texture, camera properties and location and arrangement of the matrix stacks. The geometries can be imported to the world space by the HLAPI. Their material properties, colors, textures and motions like translation, rotation and scaling can be achieved by the HLAPI. The HLAPI enables the user to set event callback functions such as collision detection, the stylus button queries and implementations as well.

The OpneGL commands can be used in the HLAPI. To capture a geometry from the OpenGL, there are two possible ways: depth buffer shapes and feedback buffer shapes.

In the depth buffer shapes method, the OpenGL renders the geometries to the depth buffer and the HLAPI gets the image from the depth buffer and applying the force algorithm, calculates the forces, which will be sent to the haptic device.

In the feedback buffer shapes method, the geometry is rendered by the OpenGL is captured in the OpenGL feedback buffer and the HLAPI gets the geometry out of the feedback buffer and applying the force algorithm, calculates the forces, which will be sent to the haptic device.

Proxy method is used for haptic rendering. The proxy is called “SCP” or “god-object” and follows the haptic interface point considering the mapping between the haptic device and the world space (graphic space). If the proxy touches a geometry it can’t penetrate the surface of the shape, whereas, the haptic interface point (haptic device position) penetrates the surface of the shape.

The HLAPI calculates the contact forces, which will be sent to the haptic device by assuming the existence of a virtual spring-damper mechanism between the proxy and the surface of the shape. The spring-damper mechanism, which is placed under the virtual object surface by the HLAPI is illustrated in Figure 2.13.

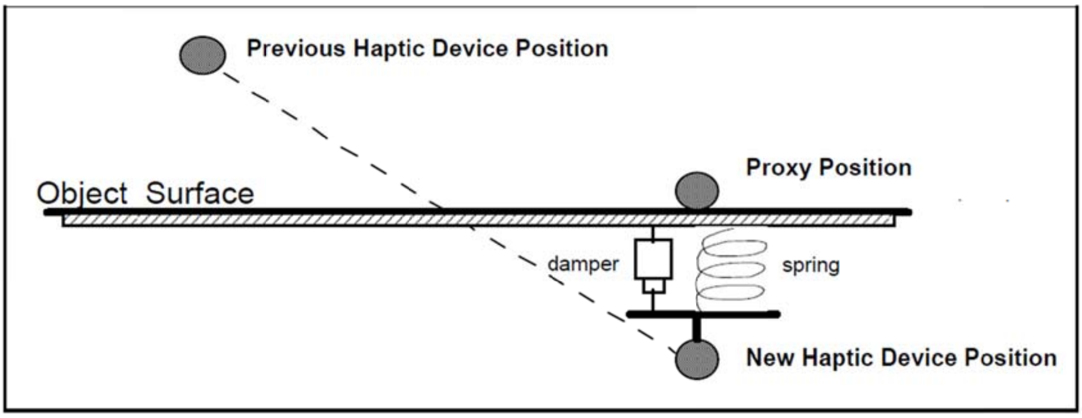


Figure 2.13 : The proxy position[8].

A typical program flowchart of the HLAPI is given in Figure 2.14. The program starts with the OpenGL setup and the HLAPI initialization. Then, the mapping between the haptic device and the graphic workspace is carried out.

In the next step, process events such as contact detection and stylus button operations are performed. The depth or the feedback buffer method is used to capture the geometry in the haptics rendering. After graphics and haptics rendering operations, a 3D cursor is rendered. Then, the loop continues with the process events step again.

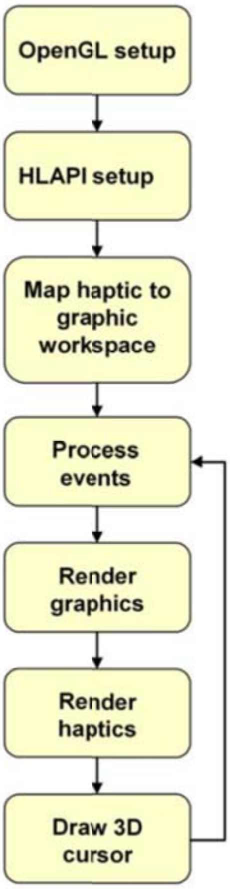


Figure 2.14 : The HLAPI program flowchart [8].

In the HLAPI, all the haptic rendering operations are realized inside a haptic frame. Therefore, before these operations, a haptic frame must be started and end of the operations, the haptic frame must be terminated.

The haptic frames inside the HLAPI program are given in Figure 2.15. At the beginning of the haptic frame, the current haptic rendering state is obtained from haptic thread and at the end of the haptic frame, haptic rendering operations are processed and the proxy position is updated.

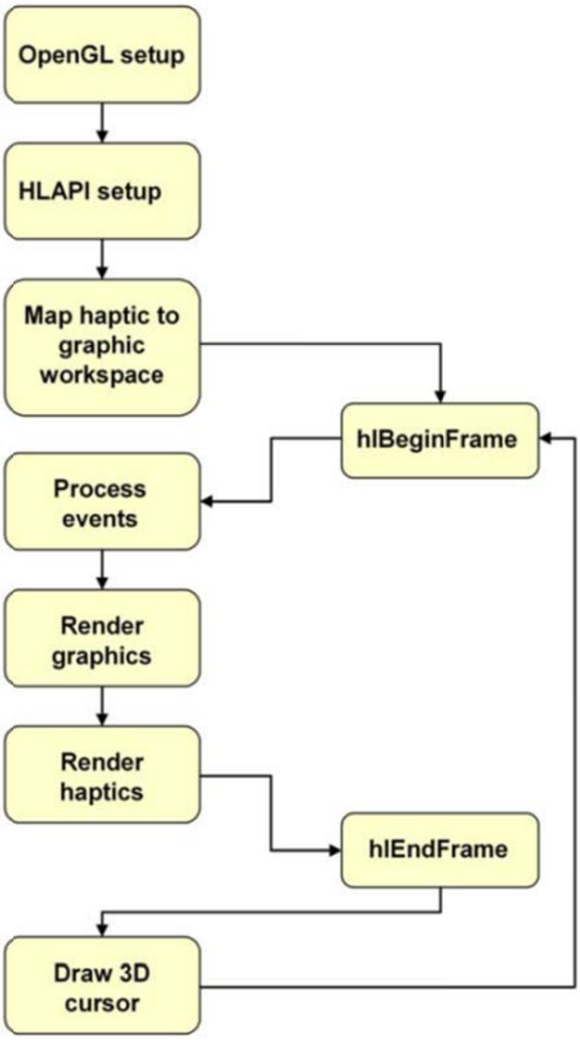


Figure 2.15 : The haptic frames inside the HLAPI program flowchart [8].

2.3.5 Multithreading

In computer programming, a thread of execution is a routine that can be accessed independently by an operating system scheduler. There can be either a single thread or multiple threads in a computer program.

The multithreading provides the concurrent execution of multiple threads. Due to this parellel processing, an operation does not need to wait the termination of an another operation. There are several methods to process an application parellely in computer programming. Multithreads, which processes parellely are used in this study.

A basic scheme of a single thread and multiple threads are shown in Figure 2.16.

The files, data and code are common for all the threads. However, each thread has its own registers and stack.

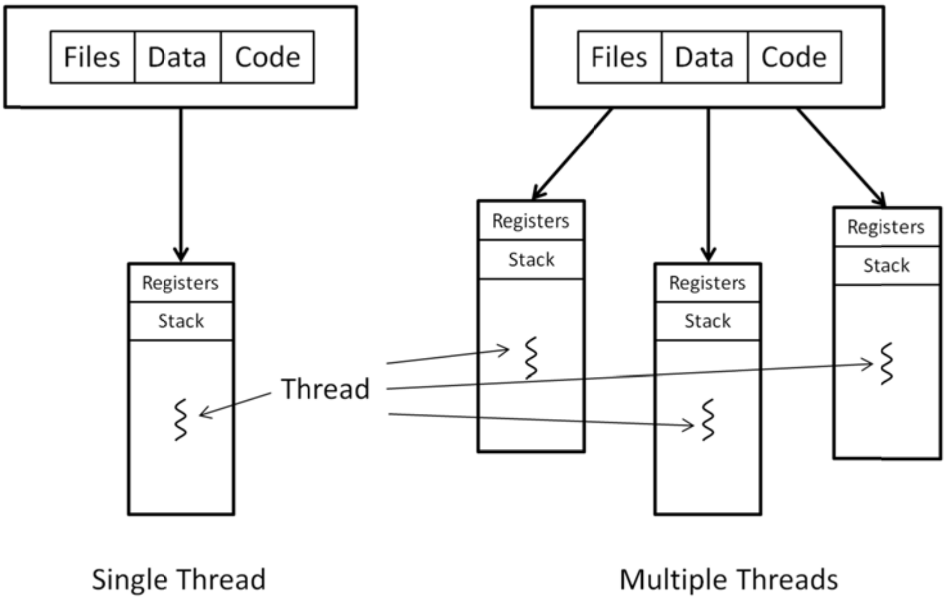


Figure 2.16 : Thread structures.

Multithreading can be utilized for different purposes in haptics applications. In generally, there is a main application thread called "client thread" and two additional threads in a typical haptics application. The additional threads are the graphics thread and servo thread. The servo loop needs to run approximately at 1000 Hz to obtain a stable force feedback. Whereas, the graphics loop needs to run approximately at 30 Hz for human eye perception.

The one of the important topics for a multithreading application is the thread safety. For example, since the servo thread runs at a high rate, sharing the data between servo and other threads can cause to time delays for haptic rendering. To avoid this situation, asynchronous calls can be used. The synchronous and asynchronous call are the two operations of the scheduler. The synchronous call returns the data after its process is finished and the application thread waits its completion. The asynchronous call returns instantly. When a process was started between two threads, that information can be logged until it is queried. Then, the information can be queried using synchronous call. Thus, the application is thread-safe.

2.3.6 Mapping the device space to world space

In generally, the device space (the haptic device workspace) and the world space (graphics space) have different dimensions from each other. Therefore, an appropriate mapping must be done between these spaces. The device space is the workspace of the haptic device. According to the application, all of this workspace or a predefined sub-workspace can be used in the application. The world space is a virtual space and it can be defined depending on the dimensions of the application.

There are two matrix stacks for mapping operation in the HLAPI. These matrices are HL_VIEWTOUCH_MATRIX and HL_TOUCHWORKSPACE_MATRIX and 4x4. The mapping between the world coordinates and the device workspace coordinates is shown in Figure 2.17. The world coordinates are the coordinates of the graphics scene. The world-view matrix is the transformation matrix, which transforms the world to camera coordinate frame. The view coordinates are the camera coordinates. The view-touch matrix is the transformation matrix, which transforms the haptic workspace to view coordinates and includes the rotations and translations of the haptic device independent of the workspace mapping. The touch coordinates are the parent coordinate system of the workspace. The touch-workspace matrix is the transformation, which transforms the workspace to the view coordinates.

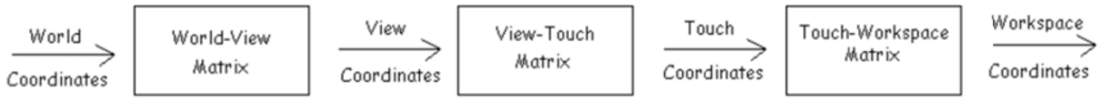


Figure 2.17 : The haptic device workspace coordinates to world coordinates mapping[8].

The HLAPI provides both uniform and non-uniform mapping methods. The non-uniform mapping is a mapping method, which includes the different scales for different axes. The QuickHaptics micro API uses a non-uniform mapping by default.

2.3.7 Calibration

The position of the haptic interface point is calculated according to the motor encoder data. However, some of the encoder data can be missed because of some reasons like overvelocity. This causes incorrect calculation of the position of the haptic interface point. Therefore, the haptic device needs to be calibrated, before the application start. The calibration data can also be corrupted during the application and should be checked during the application. Hardware reset of the encoders, inkwell calibration and auto calibration methods are the general calibration methods.

The hardware reset of the encoders method can be used for all PHANTOM haptic devices. The links are placed at the orthogonal position as shown in Figure 2.18. Then, the encoders are reseted.

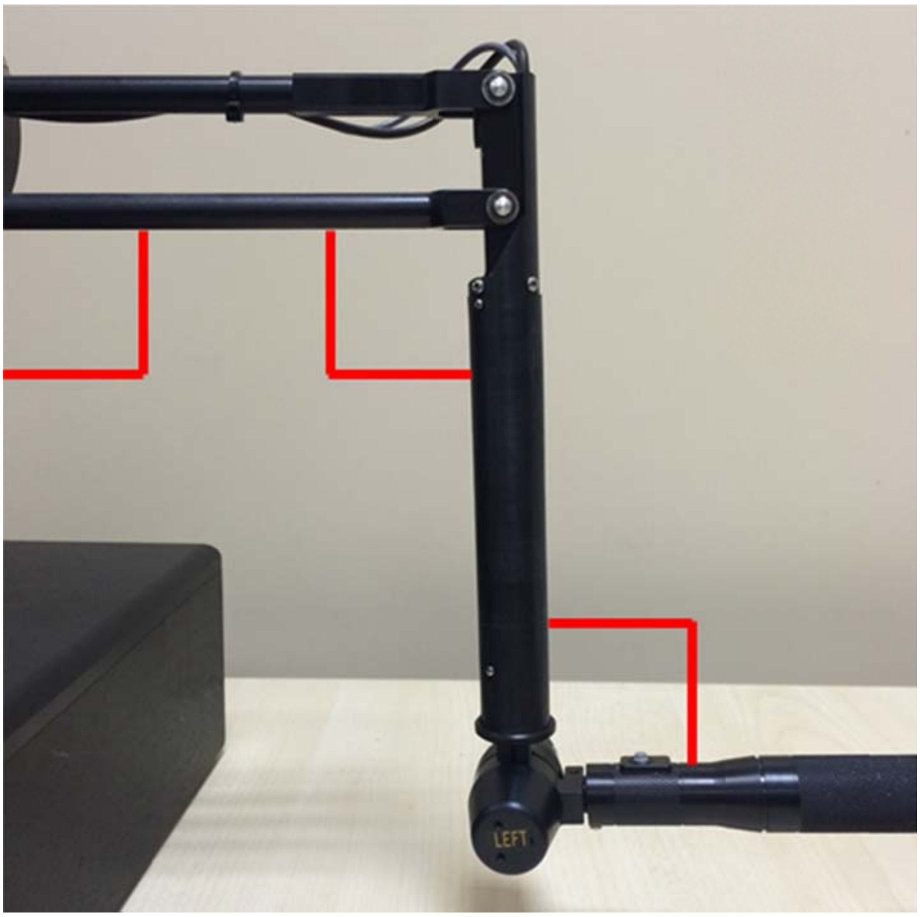


Figure 2.18 : Calibration position.

The inkwell calibration can be applied only the devices having an inkwell calibration fixture. This fixture constrains the translation and rotation of the haptic interface point. Then, the device is calibrated.

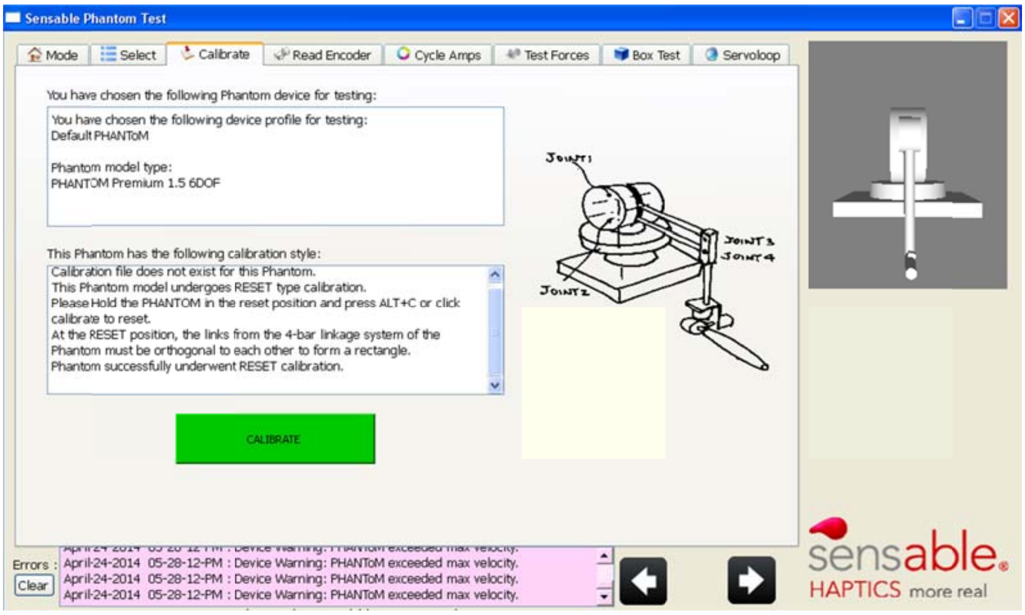
In the auto calibration method, the device calibration is checked periodically. When the device moves, the calibration is updated.

The OpenHaptics SDK provides the PHANTOM Test program and it is installed with the PDD (PHANTOM Device Drivers).

The PHANTOM Test program has a calibration interface visualizing the device movements and uses the the hardware reset of the encoders. Before running an application, it can be used for the initial calibration of the device.

The Phantom Test software also gives some informations and warnings about the haptic device such as the velocity limit or the motor temperatures.

The PHANTOM Test interface is shown in Figure 2.19.



You have chosen the following Phantom device for testing:

You have chosen the following device profile for testing:
Default PHANToM

Phantom model type:
PHANTOM Premium 1.5 6DOF

This Phantom has the following calibration style:

Calibration file does not exist for this Phantom.
This Phantom model undergoes RESET type calibration.
Please Hold the PHANTOM in the reset position and press ALT+C or click calibrate to reset.
At the RESET position, the links from the 4-bar linkage system of the Phantom must be orthogonal to each other to form a rectangle.
Phantom successfully underwent RESET calibration.

```
April-24-2014  05-28-12-PM : Device Warning: PHANToM exceeded max velocity.
April-24-2014  05-28-12-PM : Device Warning: PHANToM exceeded max velocity.
April-24-2014  05-28-12-PM : Device Warning: PHANToM exceeded max velocity.
```

Figure 2.19 : Calibration interface.

3. MODELING OF 3 DOF STAUBLI RX160

In this chapter, the geometric, kinematic, static and dynamic models of the 3 DOF Staubli RX160 manipulator are derived.

3.1 Geometric Model

The geometric model consists of the forward and inverse geometric model. The homogeneous transformation matrices are the fundamental computation tools for the geometric model. By defining homogeneous transformation matrices, the computations of the forward and inverse geometric model are achieved.

3.1.1 Homogeneous transformation matrices

In robotics, transformation between frames is a crucial topic since, computing the location, position and orientation of robot links relative to each other or specific frames are the basic calculations of robotics.

The transformation matrix expresses a frame R_j according to a frame R_i describing translations and rotations between these two frames.

(4X4) Homogeneous transformation matrix structure T_i^j is used in this study. T_i^j represents translations and rotations of a frame R_i with respect to a frame R_j. The matrix form is stated in Equation (3.1).

$$T_i^j = \left[s_i^j n_i^j a_i^j P_i^j\right] = \begin{bmatrix} s_x & n_x & a_x & P_x \\ s_y & n_y & a_y & P_y \\ s_z & n_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & A_i^j & & P_i^j \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(3.1)}$$

$A_i^j(s_i^j,\ n_i^j$ and $a_i^j)$ expresses the (3X3) rotation matrix and P_i^j expresses the translation vector. Since, A_i^j is orthogonal:

$$(A_i^j)^{-1} = (A_i^j)^T = A_i^j \quad \textbf{(3.2)}$$

However,

$$(T_i^j)^{-1} = T_j^i \tag{3.3}$$

Illustration of T_i^j is stated in Figure 3.1.

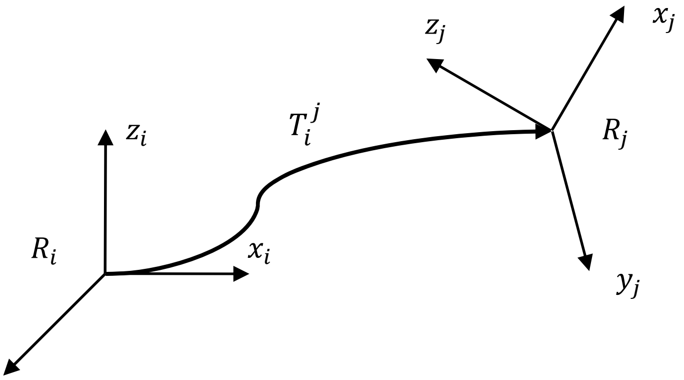


Figure 3.1 : Transformation between frame R_i and frame R_j .

The transformation matrix including only translation is expressed as Trans(a,b,c), where a,b and c denotes the translation along x,y and z axes.

$$T_i^j = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

The translation matrix including only rotation about principle axes (x,y,z) by an angle θ is expressed as Rot(x,θ), Rot(y,θ) and Rot(z,θ). Whereas, rot(x,θ), rot(y,θ) and rot(z,θ) denotes the (3X3) orientation matrices about x, y and z axes by an angle θ.

Equation (3.5), (3.6) and (3.7) are shown Rot(x,θ), Rot(y,θ) and Rot(z,θ) matrices.

$$T_i^j = \mathrm{Rot}(\mathrm{x}, \theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\theta & -S\theta & 0 \\ 0 & S\theta & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & & & 0 \\ & \mathrm{rot}(\mathrm{x}, \theta) & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

$$T_i^j = \mathrm{Rot}(y,\theta) = \begin{bmatrix} C\theta & 0 & S\theta & 0 \\ 0 & 1 & 0 & 0 \\ -S\theta & 0 & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & \mathrm{rot}(y,\theta) & & 0 \\ & & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

$$T_i^j = \mathrm{Rot}(z,\theta) = \begin{bmatrix} C\theta & -S\theta & 0 & 0 \\ S\theta & C\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & \mathrm{rot}(z,\theta) & & 0 \\ & & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

where Cθ and Sθ denotes cos(θ) and sin(θ) respectively.

3.1.2 Forward geometric model

Robot operations are described in the task space. However, robots are controlled in joint space. The forward geometric model is calculated for defining transformation equations from the joint space to the task space.

The end-effector position for applied rotation angles to links relative to the base frame is determined by multiplying its coordinate vector by the transformation matrix.

In Equation (3.8), P^j is a vector, which is defined in a frame R_j and it is transformed to a vector P^i. P^i expresses the P^j according to the frame R_i by multiplying P^j by T_i^j.

$$P^i = T_i^j P^j \tag{3.8}$$

The transformation matrices of robots, which have serial links are calculated by multiplying coordinate frame transformations of joints consecutively. The homogeneous transformation matrices from frame 0 to frame n are stated in Equation (3.9). The orientation matrix A_i^j has also this property as shown in Equation (3.10).

$$T_0^n = T_0^1(q_1)T_1^2(q_2) \dots T_{n-1}^n(q_n) \tag{3.9}$$

where q is the joint varible.

$$A_i^j = A_0^1 A_1^2 \dots A_{n-1}^n \tag{3.10}$$

Front, left, top and perspective view of Staubli RX160 are shown in Figure 3.2. The world frame and link coordinate frames of Staubli RX160 are placed as shown in Figure 3.2 (a), (b) and (c) as well. Distances between frames are stated in Figure 3.2 (c) and all the dimensions are in meter. x_0, y_0 and z_0 are the axes of the world frame and x_4, y_4 and z_4 are the axes of the end-effector frame. All of the frame origins are placed on a same plane on the x_0 and z_0 axes in order to make the solution less complicated. This placement of the coordinate frames also gives the same results with Denavit-Hartenberg convention.

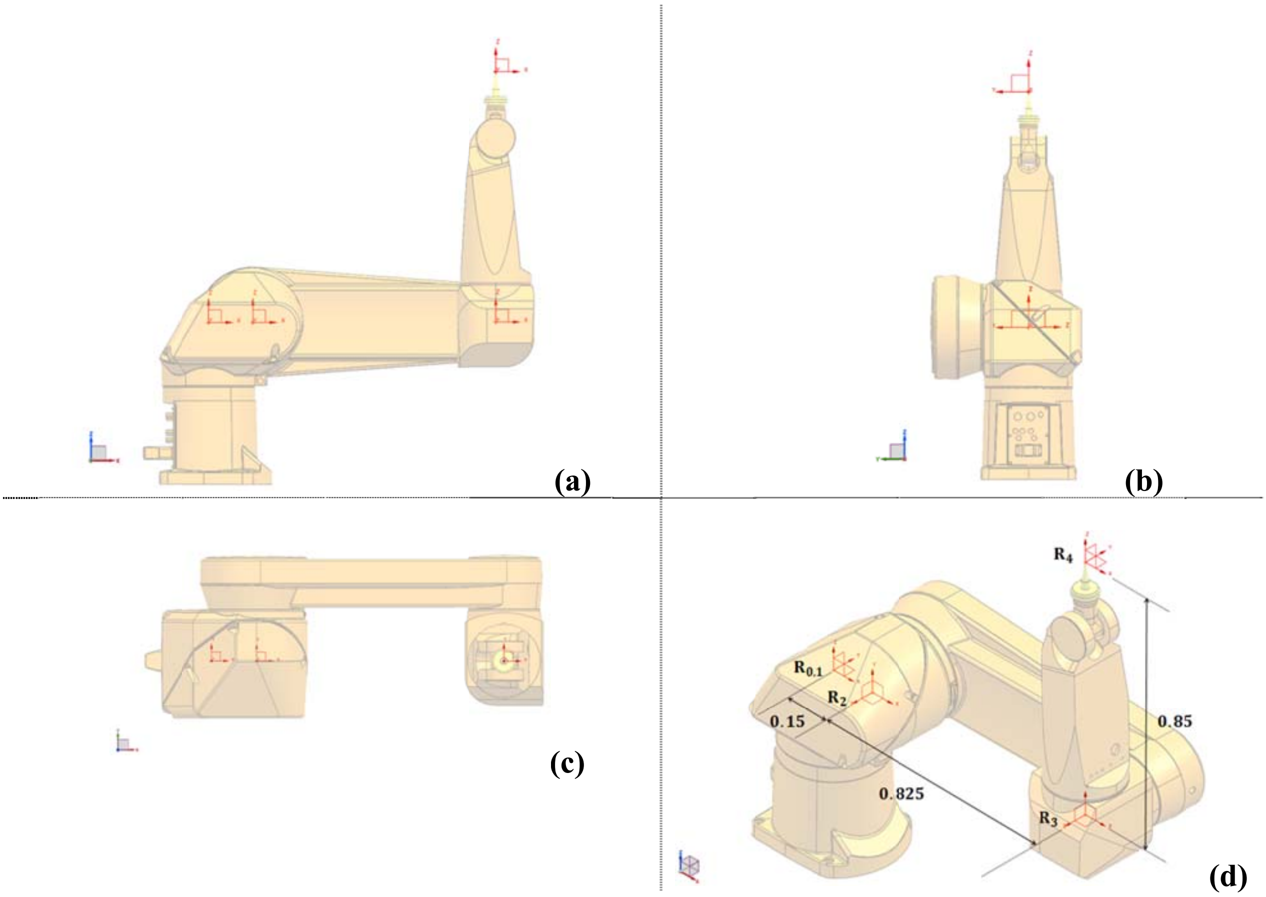


Figure 3.2 : Robot Views: (a)Front View. (b)Left View. (c)Top View. (d)Perspective View.

The coordinate frames on links are stated in Figure 3.3 explicitly. Figure 3.2 and Figure 3.3 illustrate the zero position of the robot, which is used in this study.

The origin of the fourth frame is the end-effector point of the virtual Staubli manipulator. The haptic interface point controls this point in the virtual environment.

The position, velocity and acceleration inputs are provided from the information of the haptic interface point motion in the workspace of the haptic device to the dynamic model of the 3 DOF virtual Staubli manipulator. Then, the dynamic model computes the resulting forces for the origin of the fourth frame using the position, velocity and acceleration inputs.

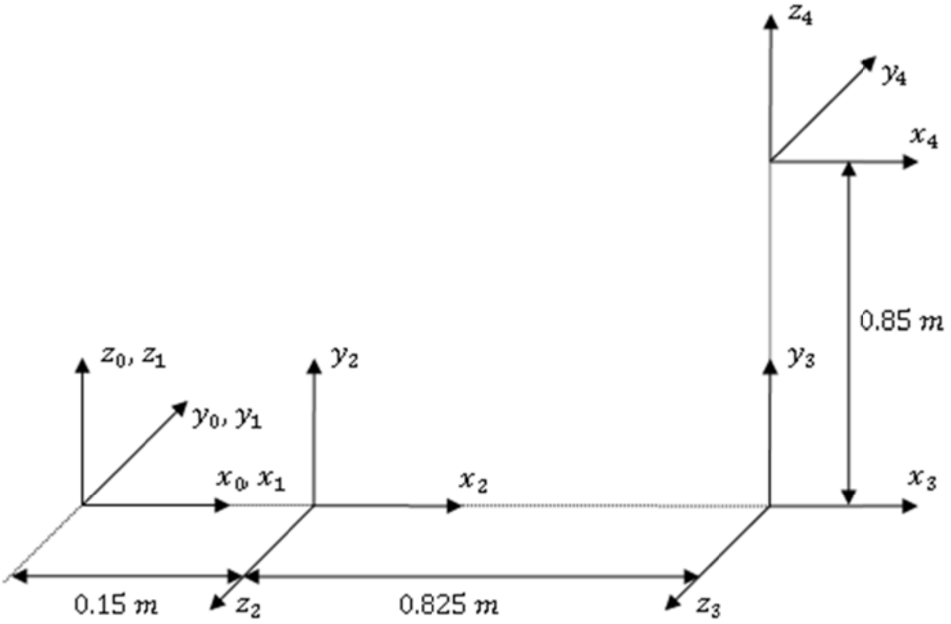


Figure 3.3 : Link coordinate frames of 3 DOF Staubli RX 160 robot.

The homogeneous transformation matices between R_0-R_1, R_1-R_2, R_2-R_3 and R_3-R_4 are calculated in Equation (3.11), (3.12), (3.13) and (3.14) respectively.

$$T_0^1(q_1) = I_4 \tag{3.11}$$

where I_4 is (4x4) identit matrix.

$$T_1^2(q_2) = \text{Trans}(\text{x}, 0.15)\text{Rot}(\text{x}, 90) \tag{3.12}$$

$$T_2^3(q_3) = \text{Trans}(\text{x}, 0.825) \tag{3.13}$$

$$T_3^4(q_4) = \text{Trans}(\text{y}, 0.850)\text{Rot}(\text{x}, -90) \tag{3.14}$$

The Staubli RX160 manipulator has serial links, therefore the homogeneous transformation matrix from world frame to fourth frame is calculated by multiplying transformation matrices as Equation (3.15):

$$T_0^4 = T_0^1(q_1)T_1^2(q_2)T_2^3(q_3)T_3^4(q_4) \qquad \textbf{(3.15)}$$

The direct geometric model of the 3DOF Staubli RX 160 is obtained as Equation (3.16).

$$T_0^4 = \left[\begin{matrix} S1S4 - C4(C1S2S3 - C1C2C3) & S4(C1S2S3 - C1C2C3) - C4S1 \\ C1S4 - C4(S1S2S3 - C2C3S1) & C1C4 + S4(S1S2S3 - C2C3S1) \\ S23C4 & S23S4 \\ 0 & 0 \end{matrix}\right.$$

$$\left.\begin{matrix} -S23C1 & C1(0.825C2 - 0.825S23 + 0.15) \\ -S23S1 & S1(0.825C2 - 0.85S23 + 0.15 \\ C23 & 0.85C23 + 0.825S2 \\ 0 & 1 \end{matrix}\right] \qquad \textbf{(3.16)}$$

where S23 denotes $\sin(\theta_2 + \theta_3)$.

3.1.3 Inverse geometric model

The inverse geometric model calculates the required joint variables to obtain the specified location of the end-effector. These calculations are complex and their complexity depends on the geometry of the robot. The inverse geometric model can have multiple solutions as well. There are both numerical and analytical computing methods to achieve the solution.

In this study, an analytical solution of 3 DOF Staubli RX160 is calculated using Paul method [Paul 81]. Paul method [Paul 1981] solves the transformation equation premultiplying left and right side of the equation by T_j^{j-1}, for j increasing from 1 to n-1 consecutively. For each equation

$$T_1^0(q_1)T_0^n = T_1^2(q_2) \ldots T_{n-1}^n(q_n)$$

$$T_2^1(q_2)T_1^0(q_1)T_0^n = T_2^3(q_3) \ldots T_{n-1}^n(q_n) \qquad \textbf{(3.17)}$$

$$T_{n-1}^n(q_n) \ldots T_1^0(q_1)T_0^n = T_{n-1}^n(q_n)$$

All joint variables are obtained by equating left and right sides of Equation (3.17).

In generally, necessary equations to obtain the joint variables are must be determined intuitively. Then, appropriate solution methods are applied to the equations.

The equation types which is encountered in Paul method and solutions of these types of equations are given in [14]. The equations which are encountered in this study are given in Table 3.1. Then, solutions of type-2, type-3 and type-6 system of equations are given to derive the inverse geometric model of 3-DOF Staubli RX160.

Table 3.1 : Types of equations encountered in Paul method[14].

Type 2	$\mathrm{X}S\theta_i + \mathrm{Y}C\theta_i = Z$
Type 3	$\mathrm{X1}S\theta_i + \mathrm{Y1}C\theta_i = Z1$ $\mathrm{X2}S\theta_i + \mathrm{Y2}C\theta_i = Z2$
Type 6	$\mathrm{W1}S\theta_j = \mathrm{X1}C\theta_i + \mathrm{Y1}S\theta_i + Z1$ $\mathrm{W2}C\theta_j = \mathrm{X2}S\theta_i + \mathrm{Y2}C\theta_i + Z2$

Solutions of types of equations which are given in Table 3.1 are as follows[14]:

Solution of type-2 system of equation:

X, Yand Z are not zero in this study. Therefore, type-2 equation is written as:

$$\mathrm{Y}C\theta_i = Z - XS\theta_i \tag{3.18}$$

Squaring the Equation (3.18), Equation (3.19) is obtained:

$$Y^2C^2\theta_i = Y^2(1 - S^2\theta_i) = Z^2 - 2ZXS\theta_i + X^2S^2\theta_i \tag{3.19}$$

By solving a second degree equation in θ_i, an equation is written in θ_i and Equation (3.20) is obtained:

$$\begin{cases} S\theta_i = \dfrac{XZ + \varepsilon Y\sqrt{X^2 + Y^2 - Z^2}}{X^2 + Y^2} \\ C\theta_i = \dfrac{YZ - \varepsilon X\sqrt{X^2 + Y^2 - Z^2}}{X^2 + Y^2} \end{cases} \tag{3.20}$$

with ε=±1. If $X^2 + Y^2 \leq Z^2$, there is no solution. Otherwise,

$$\theta_i = atan2(S\theta_i, C\theta_i) \tag{3.21}$$

Solution of type-3 system of equation:

If $\mathrm{X1Y2} - \mathrm{X2Y1} \neq 0$, two solutions of type-3 system of equation are independent. However, it is not for the equation, which is encountered in this study. Thus, one of these equations are solved as a type-2 equation. Since $\mathrm{X1Y2} - \mathrm{X2Y1} = 0$, solution of type-2 equation is reduced to Equation (3.21):

$$\begin{cases} X1S\theta_i = Z1 \\ Y2C\theta_i = Z2 \end{cases} \quad \textbf{(3.22)}$$

Thus, solution is as follows:

$$\theta_i = atan2(\frac{Z1}{X1}, \frac{Z2}{Y2}) \quad \textbf{(3.23)}$$

Solution of type-6 system of equation:

For Z1≠0 and/or Z2≠0, by squaring the both equations and adding them, a type-2 system of equation is obtained in θ_j:

$$\mathrm{B1}S\,\theta_i + B2C\theta_i = B3 \quad \textbf{(3.24)}$$

with:

$$\mathrm{B1} = 2(Z1Y + Z2X) \quad \textbf{(3.25)}$$

$$\mathrm{B2} = 2(Z1X - Z2Y) \quad \textbf{(3.26)}$$

$$\mathrm{B3} = 2(W^2 - X^2 - Y^2 - Z1^2 - Z2^2) \quad \textbf{(3.27)}$$

with θ_i known, θ_j is obtained by solving type-3 system of equation.

The homogeneous transformation matrix T_0^4 of 3 DOF Staubli RX160 has already been derived in Equation (3.16). Considering Equation (3.16), inverse geometric model of 3 DOF Staubli RX160 is obtained by applying Paul method to the derived forward geometric model. P_x, P_y and P_z are assumed as the x, y and z coordinates of the end-effector position of the virtual Staubli RX160 relative to the world frame. Premultiplying the transformation matrix T_0^4 by the origin coordinates (0,0,0) of the R_4, P_x, P_y and P_z are obtained as Equation (3.28):

$$\begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = T_0^1(q_1)T_1^2(q_2)T_2^3(q_3)T_3^4(q_4)\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.28}$$

By applying Paul method to the model, Equation (3.30) and (3.34) are derived:

$$T_1^0(q_1)\begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = T_1^2(q_2)T_2^3(q_3)T_3^4(q_4)\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.29}$$

$$\begin{bmatrix} P_x\cos(\theta_1) + P_y\sin(\theta_1) \\ P_y\cos(\theta_1) - P_x\sin(\theta_1) \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} 0.825\cos(\theta_2) - \sin(\theta_2 + \theta_3) + 0.15 \\ 0 \\ 0.85\sin(\theta_2 + \theta_3) + 0.825\sin(\theta_2) \\ 1 \end{bmatrix} \tag{3.30}$$

By equating the second raws of the matrices in the Equation (3.30), the following results are obtained:

$$\theta_1 = atan2(P_y, P_x) \tag{3.31}$$

$$\theta_1' = \theta_1 + \pi \tag{3.32}$$

In order to find θ_2 and θ_3, left and right side of the Equation (3.29) are premultiplied by ${}^2T_1(q_2)$:

$$T_2^1(q_2)T_1^0(q_1)\begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = T_2^3(q_3)T_3^4(q_4)\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.33}$$

$$\begin{bmatrix} \cos(\theta_2)\left(P_x\cos(\theta_1) + P_y\cos(\theta_1) - 0.15\right) + P_z\sin(\theta_2) \\ -\sin(\theta_2)\left(P_x\cos(\theta_1) + P_y\sin(\theta_1) - 0.15\right) + P_z\cos(\theta_2) \\ \sin(\theta_1)P_x - \cos(\theta_1)P_y \\ 1 \end{bmatrix} = \begin{bmatrix} 0.825 - 0.85\sin(\theta_3) \\ 0.85\cos(\theta_3) \\ 0 \\ 1 \end{bmatrix} \tag{3.34}$$

First and second raw of the Equation (3.34) must be solved to obtain θ_2 and θ_3.

$$\mathrm{W} = -0.85 \tag{3.35}$$

$$\mathrm{X} = P_x \cos(\theta_1) + P_y \cos(\theta_1) - 0.15 \tag{3.36}$$

$$\mathrm{Y} = P_z \tag{3.37}$$

$$\mathrm{Z1} = -0.825 \tag{3.38}$$

$$\mathrm{Z2} = 0 \tag{3.39}$$

$$\mathrm{B1} = 2(-0.825P_z) \tag{3.40}$$

$$\mathrm{B2} = 2(-0.825((P_x \cos(\theta_1) + P_y \cos(\theta_1) - 0.15)) \tag{3.41}$$

$$\begin{aligned}\mathrm{B3} = 2\,((-0.85)^2 - \left(P_x \cos(\theta_1) + P_y \cos(\theta_1) - 0.15\right)^2 \\ - (P_z)^2 - (-0.825)^2\end{aligned} \tag{3.42}$$

$$\begin{cases} S2 = \dfrac{B1B3 + \varepsilon B2\sqrt{B1^2 + B2^2 - B3^2}}{B1^2 + B2^2} \\ C2 = \dfrac{B2B3 - \varepsilon B1\sqrt{B1^2 + B2^2 - B3^2}}{B1^2 + B2^2} \end{cases} \tag{3.43}$$

The solution for θ_2 is obtained in Equation (3.44):

$$\theta_2 = atan2(S2, C2) \tag{3.44}$$

Thus, θ_2 is known and θ_3 is obtained by solving a type-3 system of equation.

$$\theta_3 = atan2(S3, C3) \tag{3.45}$$

$$\begin{cases} S3 = \dfrac{-\left(P_x\, cos(\theta_1) + P_y\, cos(\theta_1) - 0.15\right)C2 - P_zS2 - 0.825}{0.85} \\ C3 = \dfrac{P_zC2 - \left(P_x\, cos(\theta_1) + P_y\, cos(\theta_1) - 0.15\right)S2}{0.85} \end{cases} \tag{3.46}$$

3.2 Kinematic Model

Kinematics deals with pure motion without considering the effects of forces, moments or masses to the motion. Kinematic model consists of forward and inverse kinematic model.

Jacobian matrix of 3 DOF Staubli manipulator is derived to obtain the forward and the inverse kinematic models in this chapter.

3.2.1 Jacobian matrix

In robotics, the Jacobian matrix can be used for multiple objectives. In this study, Jacobian matrix is applied in the following calculations:

- To obtain the end-effector force, which represents the calculated joint torques,
- To obtain angular velocities of the joints, which represent the linear velocity of the end-effector,
- To obtain the angular accelerations of the joints, which represent the linear acceleration of the end-effector.

The Jacobian matrix is calculated by differentiating the forward geometric model. The forward geometric model is leaded as Equation (3.47) as well.

$$X = f(q) \tag{3.47}$$

where X is the position and q is the joint variable for this study. This notation is also used in the following calculations due to its simplicity. Differentiation is achieved by using partial derivatives of the model according to joint variables as Equation (3.48)

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial q_1} & \cdots & \frac{\partial f_1}{\partial q_j} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_i}{\partial q_1} & \cdots & \frac{\partial f_i}{\partial q_j} \end{bmatrix} \tag{3.48}$$

Since, 3 DOF of Staubli RX160 is modeled, reduced (3X3) Jacobian matrix is calculated. Therefore, j and i are 3 in Equation (3.48). Thus, jacobian matrix for the model is obtained as Equation (3.49):

$$J = \begin{bmatrix} -S1(0.825C2 - 0.85S23 + 0.15) & -C1(0.85C23 + 0.825S2) & -0.85C23C1 \\ C1(0.825C2 - 0.85S23 + 0.15) & -S1(0.85C23 + 0.825S2) & -0.85C23S1) \\ 0 & 0.825C2 - 0.85S23 & -0.85S23) \end{bmatrix} \quad \textbf{(3.49)}$$

Inverse of the Jacobian matrix is used in robotic computations. However, the Jacobian matrix can not be invented when the robot is in singular configurations and determinant of the Jacobian matrix is zero in these configurations. Independent rows of a jacobian matrix is indicated the number of independently controllable DOF of a robot. At a singular configuration, two or more rows are dependent, in other words, rank of the jacobian is less than DOF. There are several methods to avoid the singularities, however, it is not the objective of this study. To avoid the singularities, the approximate workspace is determined as shown Figure 3.4.

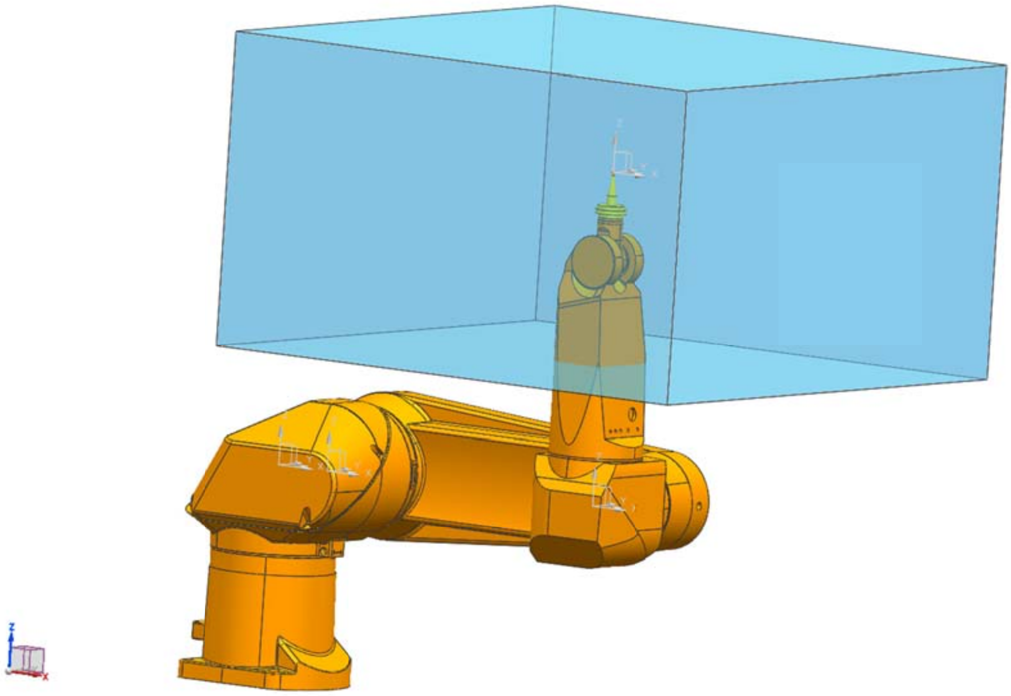


Figure 3.4 : Workspace of this study.

3.2.2 Forward kinematic model

The forward kinematic model calculates the linear velocity of the end-effector $\dot{X}$ in terms of the angular velocities of the joints $\dot{q}$. Premultiplying the angular velocities of the joints by Jacobian, end-effector velocity is obtained as in the equation (x.x).

$$\dot{X} = J\dot{q} \tag{3.50}$$

3.2.3 Inverse kinematic model

Unlike the forward kinematic model, the inverse kinematic model calculates the angular velocities of the joints $\dot{q}$ in terms of the linear velocity of the end-effector $\dot{X}$. Premultiplying the linear velocity of the end-effector by inverse of the Jacobian matrix, angular velocities of the joints are obtained.

$$\dot{q} = J^{-1}\dot{X} \tag{3.51}$$

3.3 Static Model

The static model provides the joint torques or forces exerted to the environment by the end-effector. For revolute joints, it gives the joint torques and for prismatic joints, it gives the joint forces. For 3 DOF Staubli RX160 all the joints are revolute. Therefore, joint torques are calculated with the static model.

Vector pairs, which consist of forces and moments exerted on a rigid body are called wrenches. Wrenches are represented by screws and a screw is a six-dimensional vector composed of a pair of three-dimensional vectors. Representation of a wrench is shown in Equation (3.52).

$$W = \begin{bmatrix} f \\ m \end{bmatrix} = \begin{bmatrix} f_x \; f_y \; f_z \; m_x \; m_y \; m_z \end{bmatrix}^T \tag{3.52}$$

where W is wrench, f is force and m is moment.

Since, 3 DOF Staubli RX160 is considered in this study, in other words, orientation of the Staubli RX160 is not used, wrench is reduced as the jacobian matrix. Thus, wrench is taken into account as follows:

$$W = [f] = \left[f_x\, f_y\, f_z\,\right]^T \quad \textbf{(3.53)}$$

An external wrench is mapped into the joint torques as in Equation (3.54).

$$\Gamma = J^T W \quad \textbf{(3.54)}$$

By considering Equation (3.54), calculated joint torques can be mapped into the end-effector force as in Equation (3.55). Inverse of the transpose of the Jacobian matrix must be available for the actual configuration to obtain a solution.

$$W = J^{-T}\Gamma \quad \textbf{(3.55)}$$

3.4 Dynamic Model

The dynamic model represents the relation between the applied forces/torques and the resulting motion of a rigid body. The forward dynamic model expresses the joint accelerations in terms of the joint positions, velocities and torques. Equation (3.56) defines the forward dynamic model.

$$\ddot{q} = g(q, \dot{q}, \Gamma, f_e) \quad \textbf{(3.56)}$$

where q is the joint positions, $\dot{q}$ is the joint velocities, $\ddot{q}$ is the joint accelerations, Γ is the joint torques and f_e is the forces and moments exerted by the robot on the environment.

The inverse dynamic model expresses the joint torques and forces in terms of the joint positions, velocities and accelerations. The inverse dynamic model is often called the dynamic model[14]. In this study, inverse dynamic model is called as the dynamic model as well and is given in Equation (3.57).

$$\Gamma = f(q, \dot{q}, \ddot{q}, f_e) \quad \textbf{(3.57)}$$

3.4.1 Robot dynamic parameters

In this chapter, robot parameters, which are used in the computations of dynamic model are stated.

3.4.1.1 Inertial parameters

Moment of inertia is a mass property of a rigid body and indicates the required torque value for rotating the rigid body about an axis with a specified angular acceleration. (3X3) inertia matrix is called inertia tensor.

Inertial parameters of each link of the Staubli RX160 are provided by Staubli as the form of the Equation (3.58). The inertia tensor is specified for the x,y,z axes and origin of rotation is the center of mass for the given inertia tensor.

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} \tag{3.58}$$

Center of mass coordinates of each link of Staubli RX160 are also provided by Staubli. All of the robot links are imported in a solid model software and masses, inertial tensors and center of mass coordinates provided by Staubli are defined in the software for each link. Since, the last 4 links of Staubli RX160 are assumed as a single link, the mass of the last 4 links are summed and center of mass coordinates are calculated. Inertial tensor of each link is defined in part module of the software. Then, inertia tensor for the last 4 links is measured in the assembly module of the software. Thus, The inertia tensor for the last 4 links as a single link is computed via the software.

3.4.1.2 Friction

A significant proportion of friction is occurred in powertrains of robot manipulators. Sliding friction is constituded a resistance to the motion on the powertrain surfaces. Therefore, sliding friction effects are investigated and added to the dynamic model. In reference [15], a suitable friction model for Staubli RX160 has already been obtained. This friction model and identified parameters in [15] are used for the friction model, which is given in Equation (3.62).

Equation (3.59) gives the frictional torque, which includes Coulomb, viscous and static frictions. Figure 3.5 shows the relationship between friction torque and joint velocity[15]. There are sharp transitions in the Figure 3.5.

$$\tau^{(f)} = \begin{Bmatrix} \left|\tau^{(f)}\right| \leq \tau^{(f,s)}; & if\ \dot{q} = 0 \\ sign(\dot{q})\tau^{(f)} + c^{(v)}\dot{q}; & if\ \dot{q} \neq 0 \end{Bmatrix} \tag{3.59}$$

where $\tau^{(f,s)}$ is the static friction torque and $c^{(v)}$ is the viscous friction parameter.

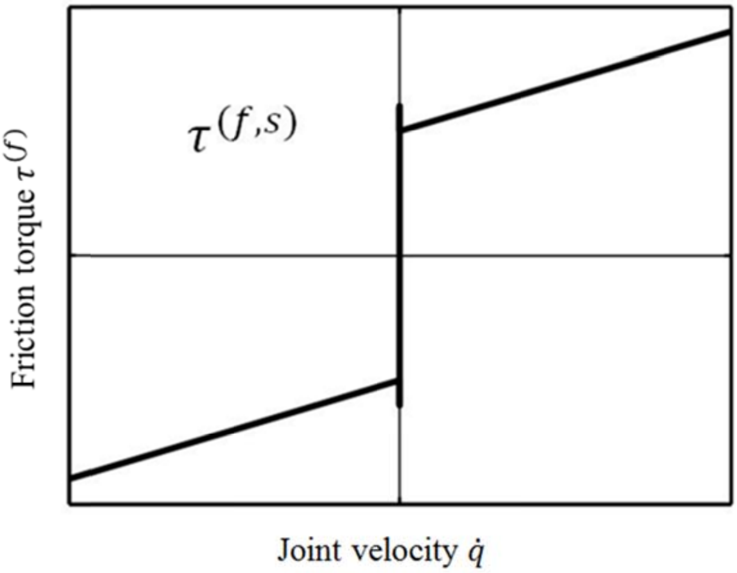


Figure 3.5 : Coulomb, viscous and static friction model[15].

Stribeck(1902) showed that the friction curve is a continuous function dependent on the joint velocity. Figure 3.6 shows the Stribeck effect to the friction.

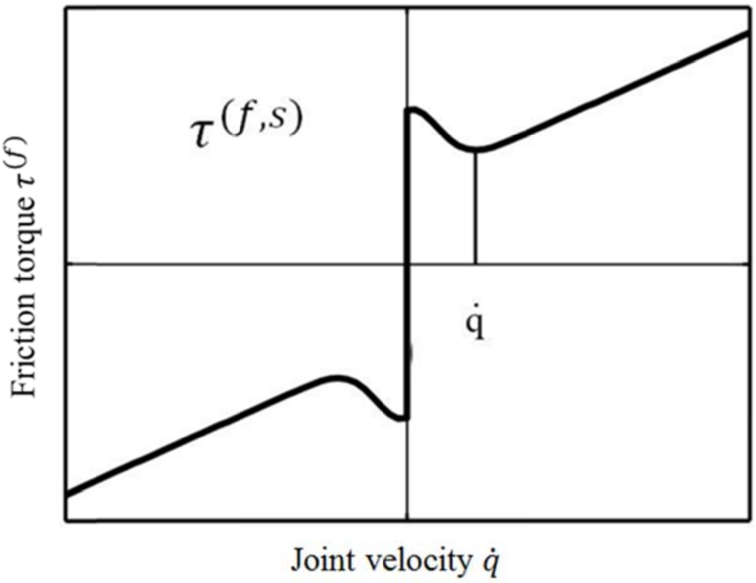


Figure 3.6 : Stribeck effect of friction model[15].

Bo and Pavelescu(1982) presented the Equation (3.60) to express the Stribeck effect[15].

$$\tau^{(f)} = sign(\dot{q})[\tau^{(f,C)} + \left(\tau^{(f,s)} - \tau^{(f,C)}\right)e^{-|\dot{q}/\dot{q}^{(s)}|^{\delta(s)}}] \quad \textbf{(3.60)}$$

where $\dot{q}^{(s)}$ is the Stribeck velocity, $\delta(s)$ is the empirical parameter dependent on the material ranging between 0.5 and 1 and $\tau^{(f,C)}$ is the Coloumb friction torque.

Armstrong-Helovury added the viscous parameter $c^{(v)}\dot{q}$ to the Equation (3.60) and Equation (3.61) is created. Equation (3.61) is widely used to obtain sliding friction torques of manipulators[15].

$$\tau^{(f)} = sign(\dot{q})[\tau^{(f,C)} + \left(\tau^{(f,s)} - \tau^{(f,C)}\right)e^{-|\dot{q}/\dot{q}^{(s)}|^{\delta(s)}} + c^{(v)}\dot{q}] \quad \textbf{(3.61)}$$

However, it is complicated to apply the Equation (3.61) to all of the friction components in the powertrains. Therefore, the Equation (3.61) is applied to the significant components in terms of friction. The significant terms of the Staubli RX160 manipulator in terms of sliding friction model include effect of the asperity of rolling bearings ${\tau_j}^{(a,BL)}$ and viscous effects.

Finally, the Equation (3.62) is obtained for the Staubli RX160 manipulator as the friction torque equation[15].

$$\tau^{(f)} = {\tau_j}^{(a,BL)} e^{\left(-\dot{q}/\dot{q}^{(s)}\right)^{\delta_j(a)}} + {c_j}^{(v)} \dot{q}^{(1-\delta_j(v))} \quad \textbf{(3.62)}$$

3.4.1.3 Balancing system

The Staubli RX160 manipulator is equipped with a spring balance system located inside the joint 2. The springs compensate the weights situated after joint 2. The balancing system is shown in Figure 3.7.

The spring torque is changed by position of the joint 2. The equation of the spring torque is provided by Staubli and it is added to the dynamic model in this study. Thus, the spring effect is also computed during the application.

The majority of the computed force values occur on the z axis for low velocity and acceleration values due to the earth gravity. By adding the spring model to the dynamic model, the maximum force values are reduced in these cases. Thus, the force values are close to each other on all the coordinate axes. Since, the uniform force mapping is realized in this study, the spring model helps to obtain more regular force values after force mapping.

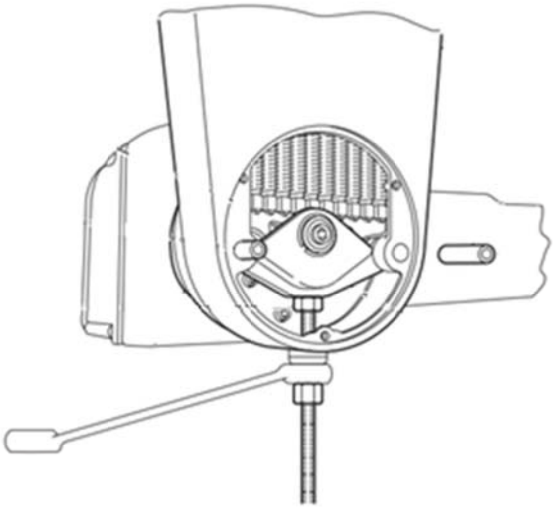


Figure 3.7 : Staubli RX160 balancing system[7].

3.4.2 Newton-Euler formulation

The Newton-Euler equations solve the forces and moments (wrench) acting on the center-of-mass of a rigid body. All the joints of the Staubli RX160 manipulator are revolute, therefore, all the equations are derived for revolute joints. Applying the D'Alembert's Principle to the robot links and omitting the viscous damping terms, Equation (3.63) and Equation (3.64) are obtained.

$$F_i = \frac{d(m_i \hat{v}_i)}{dt} = m_i \dot{\hat{v}}_i \tag{3.63}$$

where F_i is the total external force vector exerted on link i, m_i is the mass of link i and $\hat{v}_i$ is the linear acceleration vector of the center of mass of link i in base coordinates(x_0, y_0, z_0).

$$N_i = \frac{d(J_i w_i)}{dt} = J_i \dot{w}_i + w_i \times (J_i \times w_i) \tag{3.64}$$

where N_i is the total external moment vector exerted on link i, J_i is the inertia matrix of link i about its center of mass in base coordinates, w_i is the angular velocity vector with reference to the base coordinates $\dot{w}_i$ is the angular acceleration vector of link i

with reference to the base coordinates. Equations (3.65) and (3.66) represent these vectors.

$$w_{i+1} = w_i + z_i \dot{q}_i \qquad \textbf{(3.65)}$$

$$\dot{w}_{i+1} = \dot{w}_i + z_i \ddot{q}_i + \dot{w}_i \times (z_i \dot{q}_i) \qquad \textbf{(3.66)}$$

where z_i is the unit vector of the R_i on z axis.

$\hat{v}_{i+1}$ and $\dot{\hat{v}}_i$ are given in Equation (3.67) and (3.68) respectively.

$$\hat{v}_{i+1} = w_i \times \hat{s}_i + v_i \qquad \textbf{(3.67)}$$

$$\dot{\hat{v}}_i = \dot{w}_i \times \hat{s}_i + w_i \times (w_i \times \hat{s}_i) + \dot{v}_i \qquad \textbf{(3.68)}$$

v_{i+1} and $\dot{v}_{i+1}$ are the linear velocity and linear acceleration vector of joint $i+1$ respectively and given in Equation (3.69) and (3.70).

$$v_{i+1} = w_i \times {p_{i+1}}^{*} + v_i \qquad \textbf{(3.69)}$$

$$\dot{v}_{i+1} = \dot{w}_{i+1} \times {p_{i+1}}^{*} + w_{i+1} \times (w_{i+1} \times {p_{i+1}}^{*}) + \dot{v}_i \qquad \textbf{(3.70)}$$

The forces and moments exerted to the link i by link $i-1$ and $i+1$ are added to the total force and torque in Equation (3.71) and (3.72).

$$f_i = f_{i+1} + F_i \qquad \textbf{(3.71)}$$

$$n_i = n_{i+1} + {p_i}^{*} f_{i+1} + ({p_i}^{*} + \hat{s}_i) F_i + N_i \qquad \textbf{(3.72)}$$

Representation of f_i, n_i, $\hat{s}_i$ and ${p_{i+1}}^{*}$ are stated in Figure 3.8.

Equation (3.73) computes the torque projections on the joints for a robot motion. These torque projections are the resulting scalar torque values of the robot joints for the motion.

$$\tau_i = n_i^T z_{i-1} + b_i \dot{q}_i \qquad \textbf{(3.73)}$$

where b_i is the visous damping coefficient.

The equations of motion, which are derived in the Newton-Euler formulation section are relative to the base frame. However, the inertia matrices change by the rotations of links. Therefore, computations of dynamic model are complicated for solving the equations in one specific frame.

The dynamic model parameters are moved to their own joint spaces to simplify the computations. To obtain an efficient algoritm, computations are done in the joint spaces. For this purpose, all the vectors and matrices in the dynamic model are moved into their own joint spaces.

The orientation matrices of the 3 DOF Staubli RX160 manipulator, which are given in Equation (3.74) to (3.77) are used to move the vectors and the matrices to their own joint spaces.

$$A_0^1 = \mathrm{rot}(\mathrm{z}, \theta_1) \tag{3.74}$$

$$A_1^2 = rot(x, 90)rot(z, \theta_2) \tag{3.75}$$

$$A_2^3 = rot(z, \theta_3) \tag{3.76}$$

$$A_3^4 = rot(x, -90) \tag{3.77}$$

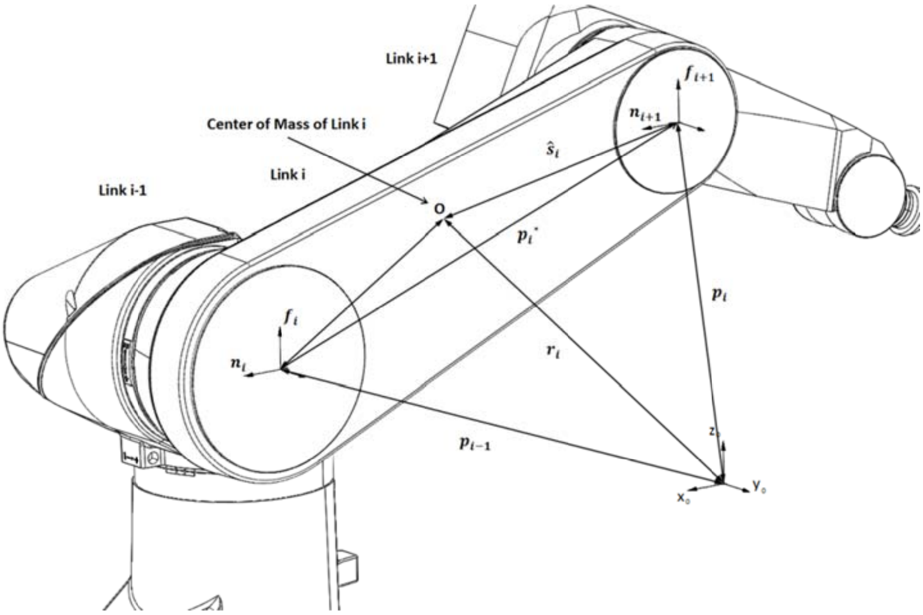


Figure 3.8 : Force, torque and position vctors on Staubli RX160 manipulator.

By using the properties of the orientation matrix A_j^i, which are given in Equation (3.2), computations of the dynamic model are obtained as in Equation (3.78) to (3.88).

$$A_i^0 F_i = m_i A_i^0 \dot{\hat{v}}_i \tag{3.78}$$

$$A_i^0 N_i = \left(A_i^0 J_i A_0^i\right)(A_i^0 \dot{w}_i) + (A_i^0 w_i) \times [\left(A_i^0 J_i A_0^i\right)(A_i^0 w_i)] \tag{3.79}$$

$$A_{i+1}^0 w_{i+1} = A_{i+1}^i (w_i + z_0 \dot{q}_i) \tag{3.80}$$

$$A_i^0 \dot{\hat{v}}_i = (A_i^0 \dot{w}_i) \times (A_i^0 \hat{s}_i) + (A_i^0 w_i) \times [(A_i^0 w_i) \times (A_i^0 \hat{s}_i) + A_i^0 \dot{v}_i \tag{3.83}$$

$$A_{i+1}^0 v_{i+1} = (A_{i+1}^0 w_i) \times (A_{i+1}^0 p_{i+1}{}^*) + A_{i+1}^i (A_i^0 v_i) \tag{3.84}$$

$$\begin{aligned} A_{i+1}^0 \dot{v}_{i+1} = {} & (A_{i+1}^0 \dot{w}_{i+1}) \times (A_{i+1}^0 p_{i+1}{}^*) + (A_{i+1}^0 w_{i+1}) \\ & \times [(A_{i+1}^0 w_{i+1}) \times (A_{i+1}^0 p_{i+1}{}^*)] + A_{i+1}^i (A_i^0 \dot{v}_i) \end{aligned} \tag{3.85}$$

$$A_i^0 f_i = A_i^{i+1} (A_{i+1}^0 f_{i+1}) + A_i^0 F_i \tag{3.86}$$

$$\begin{aligned} A_i^0 n_i = {} & A_i^{i+1} [A_{i+1}^0 n_{i+1} + (A_{i+1}^0 p_i{}^*) \times (A_{i+1}^0 f_{i+1})] \\ & + (A_i^0 p_i{}^* + A_i^0 \hat{s}_i)(A_i^0 F_i) + A_i^0 N_i \end{aligned} \tag{3.87}$$

$$\tau_i = (A_i^0 n_i)^T (A_{i+1}^i z_0) + \tau_i \tag{3.88}$$

3.4.2.1 Newton-Euler recursive computation

Newton-Euler approach is an efficient method to compute the dynamic model for real time applications. Newton-Euler recursive computation is proposed by Luh, Walker and Paul in 1980[16]. It is called Luh-Paul-Walker algorithm as well. The algorithm consists of two recursive computations as forward recursion and backward recursion.

The variables, which are used in Luh-Paul-Walker algorithm are stated as[16]:

(1) Constants are determined: n=number of the joints, $w_0 = \dot{w}_0 = 0$ and $v_0 = 0$, since the robot is fixed to the ground, however, using $\dot{v}_0 = \begin{bmatrix} 0 \\ 0 \\ -9{,}8 m/s^2 \end{bmatrix}$, effect of the gravity is taken to the account.

(2) Joint variables are $q, \dot{q} \ are \ \ddot{q}$ for $i = 0,1,2, \dots, n$.

(3) Link-wise variables are $i,\ F_i, N_i, f_i, n_i, \tau_i$.

Steps of Luh-Paul-Walker algorithm are stated as[16]:

Step 0: Set $i = 1$.

Step 1: Compute $A_i^0 w_i, A_i^0 \dot{w}_i, A_i^0 v_i, A_i^0 \dot{v}_i$.

Step 2: Compute $A_i^0 \hat{v}_i$.

Step 3: Compute $A_i^0 F_i, A_i^0 N_i$.

Step 4: If $i = n$ continue. Otherwise set $i = i + 1$ and return to Step 1.

Step 5: Compute $A_i^0 f_i \ and\ A_i^0 n_i$ with f_{n+1} and n_{n+1} being the required forces and moments, respectively, for the hand (i.e. ink n) to carry the load.

Step 6: $i = 1$ stop. Otherwise set $i = i - 1$ and return to Step 5.

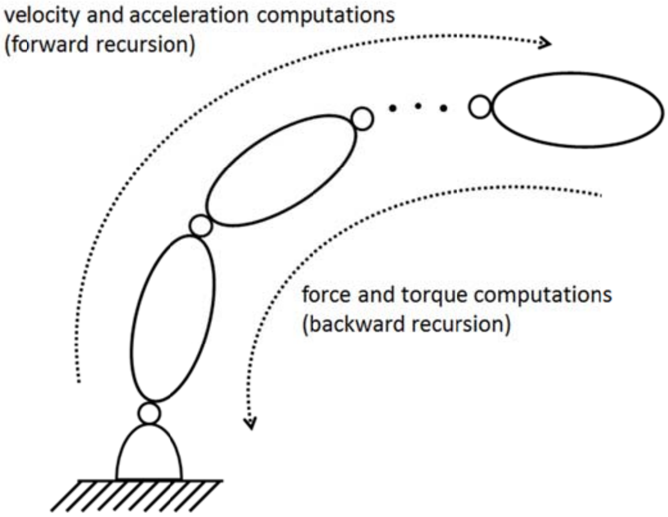

velocity and acceleration computations
(forward recursion)
force and torque computations
(backward recursion)

Figure 3.9 : Forward and backward recursion shema.

4. VISUAL INTERFACE

A visual interface is designed for the interaction between the haptic device and the virtual Staubli RX160. The visual interface is a window, which displays the 3D space rendered by the OpenGL. This is the world space for the virtual Staubli RX160.

4.1 Virtual Model of Staubli RX160

The .3ds file of Staubli RX160 robot is imported to the world space. This is the solid model of the Staubli RX160 and the physical dimensions are the ones of the the real robot. There are seven separate components of Staubli RX160. These components are tool flange, wrist, forearm, elbow, arm, shoulder and base.

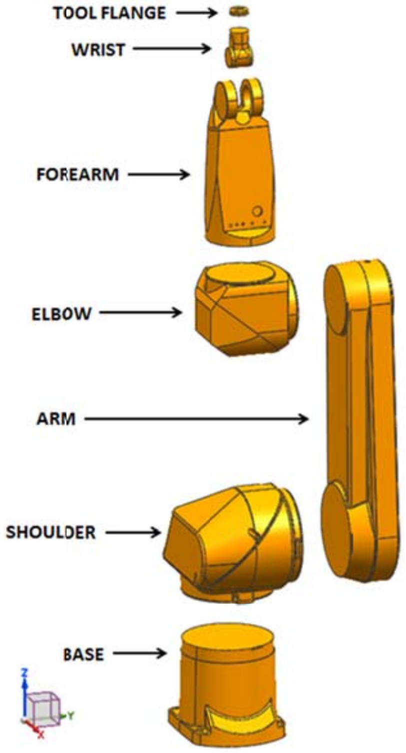


Figure 4.1 : Links of the Staubli RX 160 manipulator.

A tool is designed in NX Unigraphics solid modelling software to manipulate the virtual model. The mass of the tool is calculated using NX Unigraphics and The tool is shown in Figure 4.2.

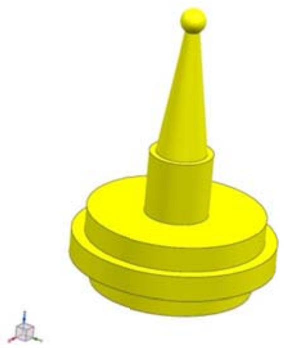


Figure 4.2 : End-effector of the Staubli RX160 manipulator.

The .3ds files include the coordinate information of the parts. Therefore, all the parts of the Staubli RX160 imported to their places, which is shown in Figure 4.3. The tool is placed to top of the flange tool. There is only one coordinate system in the world space. All the links and the origin of the coordinate system of the world space (the origin of the coordinate system of the OpenGL space) are shown in Figure 4.3.

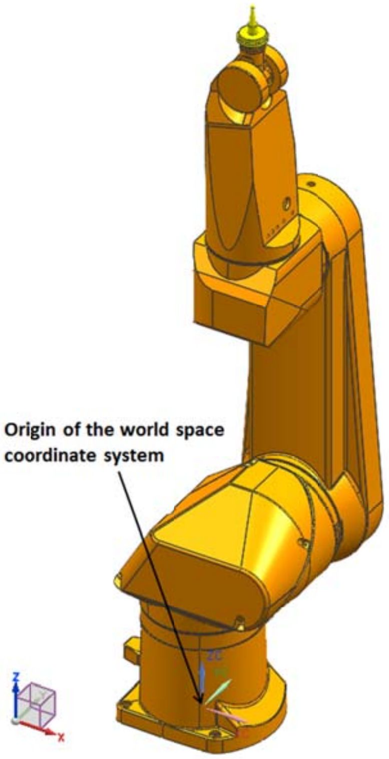


Figure 4.3 : Virtual model of the Staubli RX 160 manipulator.

4.2 Motion Algorithm of the Staubli RX160

A proxy is used to change the end effector position by using the haptic device position information. All the parts of the virtual model must be moved by considering the hierarchy of the 3 DOF Staubli RX160.

4.2.1 Hierarchy modeling

The hierarchy models are used to animate the objects by considering the interactions with each other in the virtual reality. The hierarchy is created considering the constraints of the objects. Tree model is a common method to generate the hierarchy models. The tree model includes parent and child nodes. The child node movement is dependent on its parent movement. The Staubli RX160 is an articulated robot and its links are connected to each other serially. Therefore, each link has a one child and one parent node except the base and the tool flange. The base is the root node and the tool flange is the leaf node. The root nodes don’t have any parent node and the leaf nodes don’t have any child nodes. Tree-structured hierarchy of the virtual Staubli RX160 is shown in Figure 4.4. Each block in Figure 4.4 represents a node for the tree model. The base is the root node and the last block, which includes the last 4 links and the tool is the leaf node. Since, the first 3DOF of Staubli RX160 is modelled in this study, the elbow, the forearm, the wrist, the tool flange and the tool assumed as one link and showed in a one single block in the Figure 4.4. If one link rotates, all the links between that link and the end-effector also rotates according to the hierarchy. For example if the shoulder rotates, the shoulder, arm, elbow, forearm, wrist, tool flange, and tool rotate as a single link or if the arm rotates, the arm, elbow, forearm, wrist, tool flange, and the tool rotate as a single link.

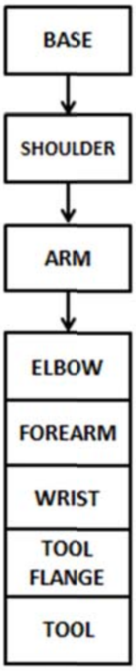


Figure 4.4 : Tree-structured hierarchy model of the Staubli RX160 manipulator.

4.2.2 Proxy rendering

The HLAPI provides the both direct and default proxy rendering. The default proxy rendering is disabled to translate and rotate the proxy by user defined motion algorithms. In this study, the proxy rendering is achieved by the user commands, which update the proxy position by the movements of the haptic interface point.

The default proxy is a 3D blue cone and also it is called cursor. It starts to move from the origin of the world space. However, the robot is controlled at the end-effector or origin of the 4th frame in this study. Therefore, the cursor is moved to the origin of the 4th frame. Thus, the proxy starts to motion from the zero position of the Staubli RX160, which is stated in Figure 3.2.

4.2.3 Motion algorithm of the virtual Staubli RX160

All the links and the tool are located at their initial position into the world space and an algorithm is developed to obtain the robot motion. In this study, the proxy moves by the movement of the haptic interface point. Therefore, the position information is fed to the proxy from the haptic device. The angle values of the Staubli RX160 are calculated by the inverse geometric model of the 3 DOF Staubli RX160, which is derived in chapter 2 to obtain the proxy position.

The motion algorithm has 5 basic steps:

Step 1: Calculate the distance between the origin of the world space and the rotation point of the link:

$$D_i = T(\theta_i)\,{}_o^i \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \qquad \textbf{(4.1)}$$

where, i is the link number and the D_i is the distance between the origin of the world space and the rotation point of link i. By multiplying the transformation matrix $T(\theta_i)\,{}_o^i$ by the origin position (0, 0, 0) of the ith link, the D_i is obtained.

Step 2: Move the link to the origin of the world space using the distance obtained in step 1.

Step 3: Calculate the rotation axis vector of the link i corresponding to the coordinate system of the world space.

$$RV_i = R(\theta_i)\,{}^{i}_{o}\begin{bmatrix}0\\0\\1\\1\end{bmatrix} \tag{4.2}$$

where the RV_i is the rotation axis vector of the link i corresponding to the coordinate system of the world space. z-axis is the rotation axis for all the links. Therefore, by multiplying the rotation matrix $R(\theta_i)\,{}^{i}_{o}$ by the unit z-axis vector (0, 0, 1) the RV_i is obtained.

Step 4: Rotate the link.

Step 5: Move the link back using negative of the distance calculated in step 1.

These 5 steps are done for each DOF recursively. Therefore, to move the end-effector from a position to a new position, the 5 steps are done 3 times. The 1st and 2nd steps include the mathematical calculations and the 3rd, 4th and 5th steps are achieved by the OpenHaptics commands.

The illustration of the algorithm for a position change of the end-effector is shown in Figure 4.5.

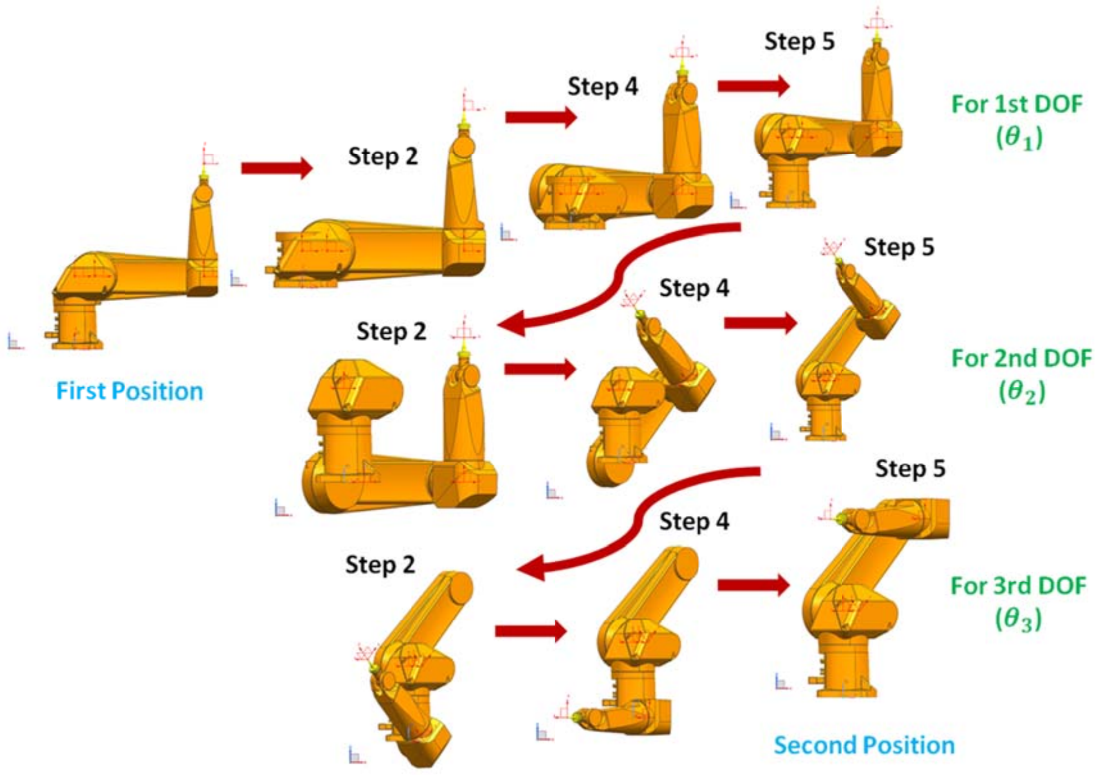


Figure 4.5 : The illustration of the motion algorithm for a position change of the end-effector.

The algorithm is iterated in a graphics loop. Since, the graphics thread runs at 30 Hz, the algorithm is iterated 30 times per second. The graphics operations are rendered at the end of the loop. Therefore, only the configurations for the end-effector positions are drawn in the scene. The visual interface window including the Staubli manipulator at its zero position for this study is shown in Figure 4.6.

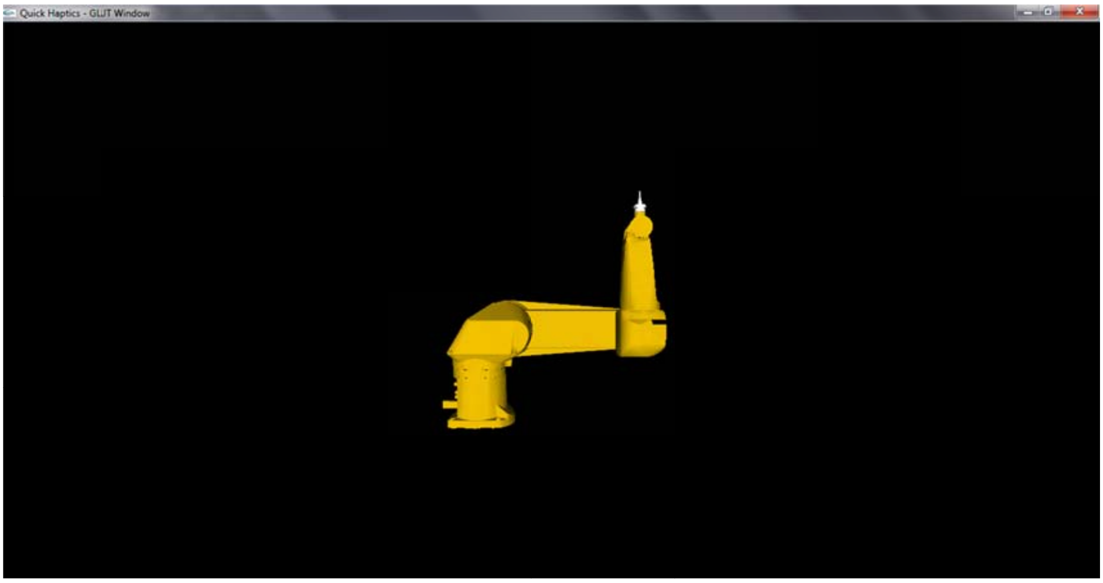


Figure 4.6 : The visual interface window.

5. APPLICATION AND EXPERIMENTS

In this chapter, application setup and experiments are described and the experimental results are given.

5.1 Application Setup

The application setup consists of a PC and a Phantom Premium 1.5 HF 6 DOF haptic device. The communication between the haptic device and the PC are achieved via the parellel communication by the parallel ports of the PC and the haptic device. The application is developed using the OpenHaptics SDK, which is based on the C/C++ programming language in Microsoft Visual Studio IDE (Integrated Development Environment).

At first, the haptic device is calibrated. Before running the application, the haptic device is positioned at its zero position. When the application is run, the interface and MS-DOS windows appear. The movement of the virtual Staubli manipulator is visualized in the graphical interface and the desired information about the application is printed in the MS-DOS window on-line. Thus, the user can follow both the movement of the virtual Staubli manipulator and the desired information such as position, velocity, acceleration inputs and resulting forces. User defined informations about the application can be printed in the MS-DOS window as well.

Initially, the virtual model of the Staubli RX 160 manipulator is its zero position and it moves by the movement of the haptic interface point. The software computes graphics and haptic rendering algorithms and applies the resulting visual and force feedback toward the user in real time.

During the experiment, the software records the position, velocity, acceleration input and the resulting forces. After the application is stopped, the software writes the recorded data to a file.

The application setup is shown in Figure 5.1.

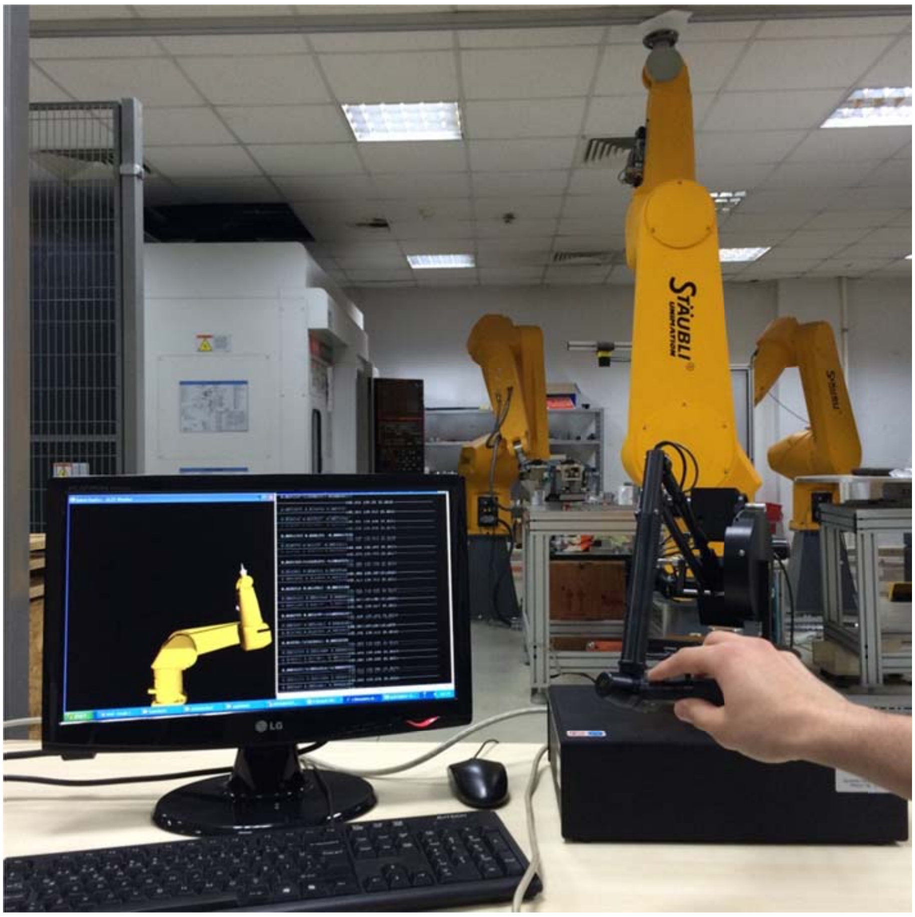


Figure 5.1 : Application Setup.

5.2 Experiments

The main objective of the experiments is investigating the stability of the system in diverse conditions. Some particular conditions are determined and the experiments are realized for these conditions.

The general scheme of the application is shown in Figure 5.2. There are two main loops in the scheme running at 30 Hz and 1000 Hz.

The human eye can perceive approximately 30 frames per second and the human tactile perception can feel the impacts continuous at 1000 Hz. Therefore, the graphic rendering loop runs at 30 Hz and the haptic rendering loop runs at 1000 Hz.

The end-effector of the Staubli RX160 manipulator follows the haptic interface point in the application. The human operator provides the position input to the system. Then, the graphic and haptic rendering loops process the position input at particular frequencies. Finally, resulting visual feedback at 30 Hz and resulting force at 1000 Hz applied toward the human operator.

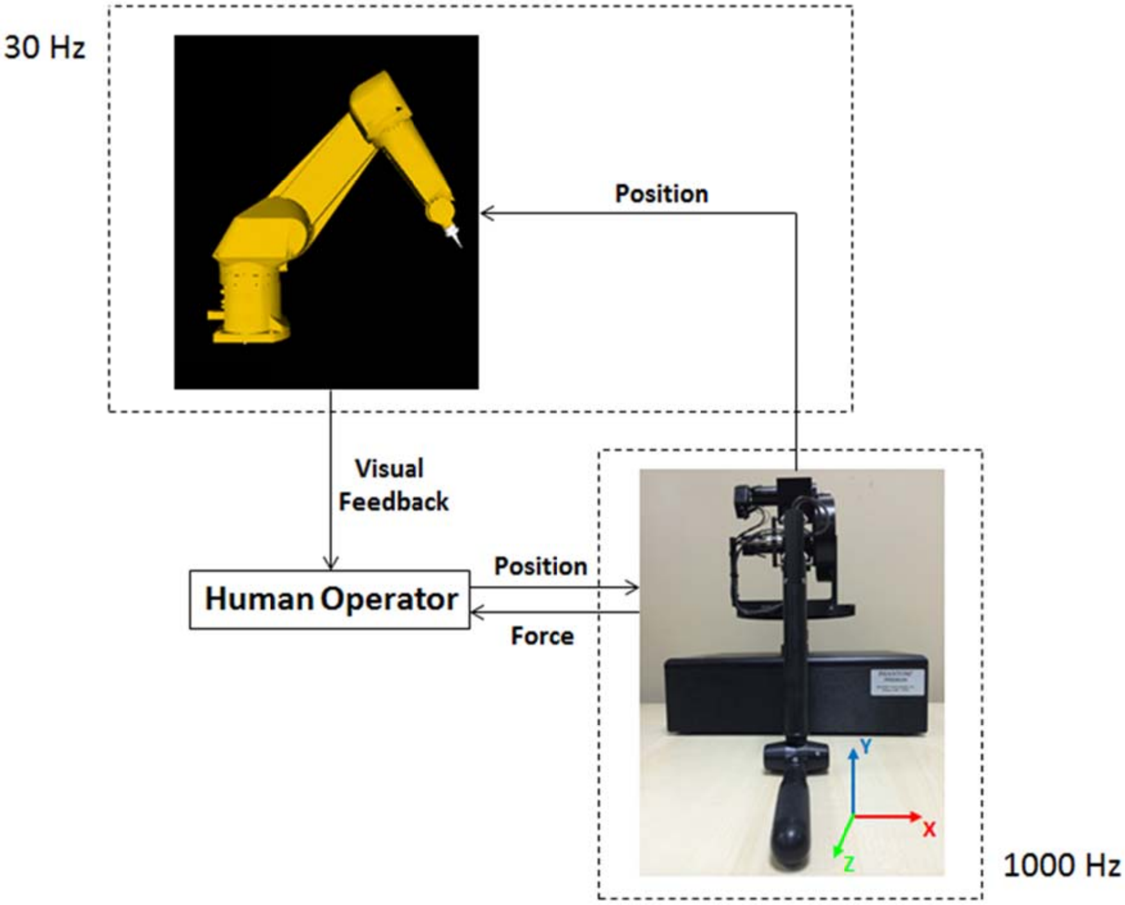


Figure 5.2 : General scheme of the application.

Block diagram of the application is shown in Figure 5.3.

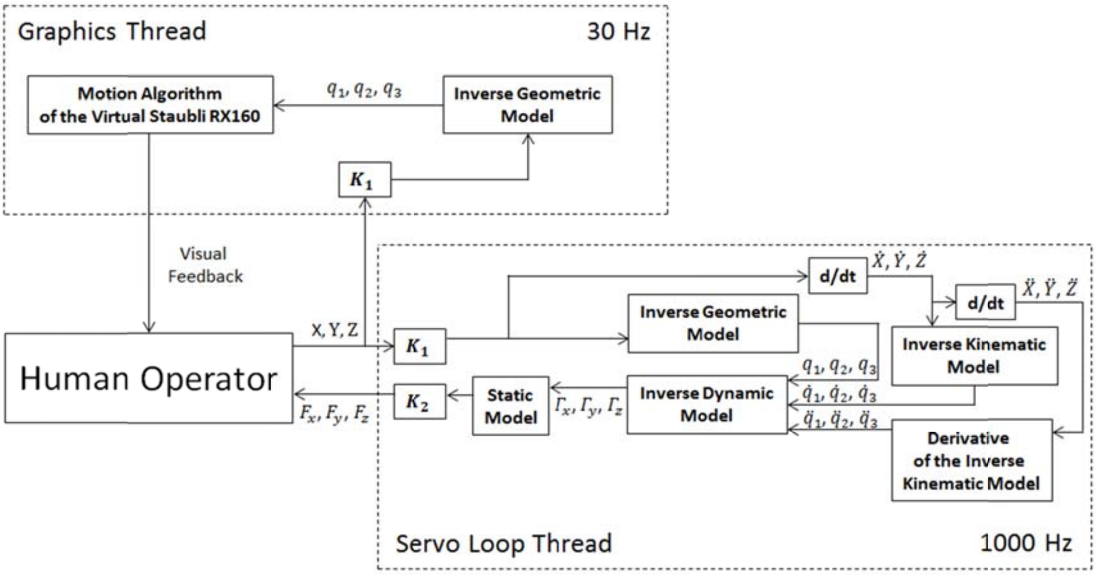


Figure 5.3 : Block diagram of the application.

The inverse dynamic model block is the main process of the servo loop thread. The inverse dynamic model of the 3 DOF Staubli RX160 including the joint frictions and the spring system is derived as Equation (5.1).

$$\Gamma = f(q, \dot{q}, \ddot{q}) \quad \textbf{(5.1)}$$

q, $\dot{q}$ and $\ddot{q}$ are the inputs and Γ is the output of the inverse dynamic model.

X, Y and Z are the position information of the haptic interface point and it is obtained from the encoders of the Phantom haptic device using the HDAPI. $\dot{X}, \dot{Y}$ and $\dot{Z}$ are the linear velocity information of the haptic interface point and are obtained by taking derivative of the X, Y and Z numerically. $\ddot{X}, \ddot{Y}$ and $\ddot{Z}$ are the linear acceleration information of the haptic interface point and are obtained by taking derivative of the $\dot{X}, \dot{Y}$ and $\dot{Z}$ numerically.

q is the joint positions of the Staubli manipulator for its desired end-effector position. The end-effector position of the virtual Staubli manipulator depends on the X, Y and Z. q is computed using inverse geometric model, which is derived in chapter 3. $\dot{q}$ is the joint velocity vector of the virtual Staubli manipulator and it is computed using inverse kinematic model, which is derived in chapter 3. $\ddot{q}$ is the joint acceleration vector of the virtual Staubli manipulator. By taking derivative of the forward kinematics model, Equation (5.2) is obtained.

$$\ddot{X} = \dot{J}\dot{q} + J\ddot{q} \quad \textbf{(5.2)}$$

Then, $\ddot{q}$ is computed using Equation (5.3).

$$\ddot{q} = \frac{\ddot{X} - \dot{J}\dot{q}}{J} \quad \textbf{(5.3)}$$

K_1 is the position gain and scales the haptic interface point movement to the end-effector movement of the Staubli maniulator. K_2 is the force gain and scales the computed forces to the Phantom haptic device. By using the static model equations, computed torques are mapped to the end-effector force.

The motion algorithm of the 3 DOF Staubli RX160 is in the graphics thread. It is given in chapter 4 and provides the visual feedback toward the user.

The experiments are realized for two paths. The first path is shown in Figure 5.4. In this path, the haptic interface point is moved along the positive and negative directions along x, y and z axes respectively.

The position, velocity, acceleration and force projection graphs are plotted to compare their effects to each other clearly. In the appendix, magnitude graphs of the position, velocity, acceleration and force values are given for the same 12 experiments.

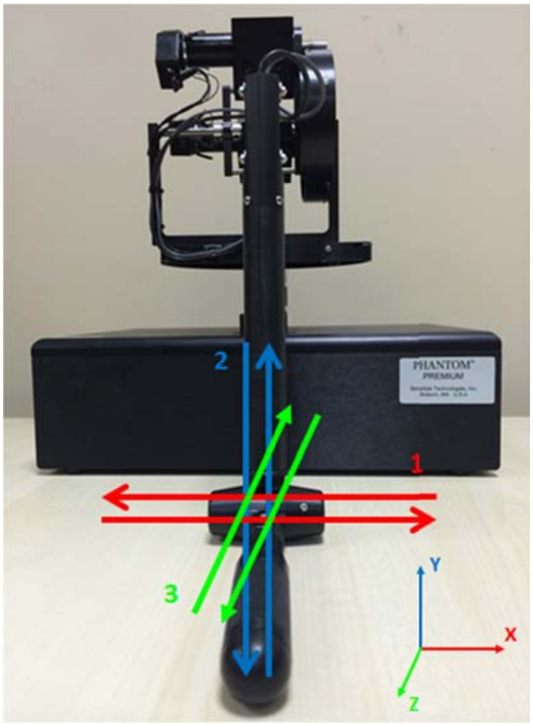


Figure 5.4 : First experiment path.

By moving the haptic interface point to negative and positive directions along the coordinate axes, sinusoidal oscillations are created on the position projections. By using this path, the oscillations are obtained on all the axes and the projection signals can be investigated seperately.

Since, the haptic rendering algorithm runs at 1000 Hz, the position information is sampled from the encoders per 1 millisecond. Then, the velocity, acceleration and force values are computed per 1 millisecond as well. Therefore, the system is a discrete time system and the software records the position, velocity, acceleration and force informations per 1 millisecond.

The first 6 experiments are realized for first path and the resulting graphs are given below.

Experiment 1: Position gain $K_1 = 1$, $K_2 = 0{,}01$.

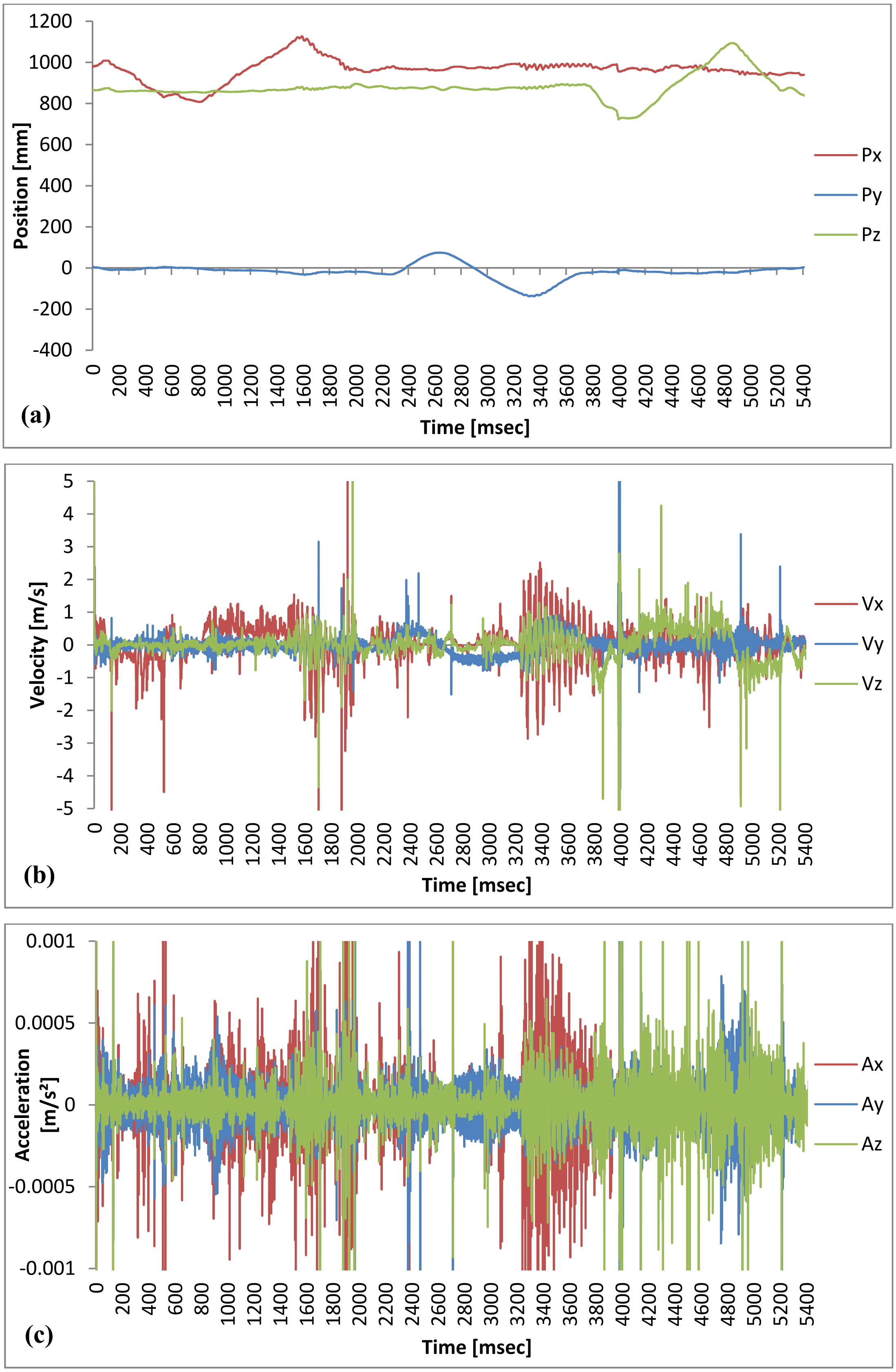


Figure 5.5 : Experimental results for first path, $K_1 = 1$ and $K_2 = 0{,}01$: (a)Position. (b)Velocity. (c)Acceleration.

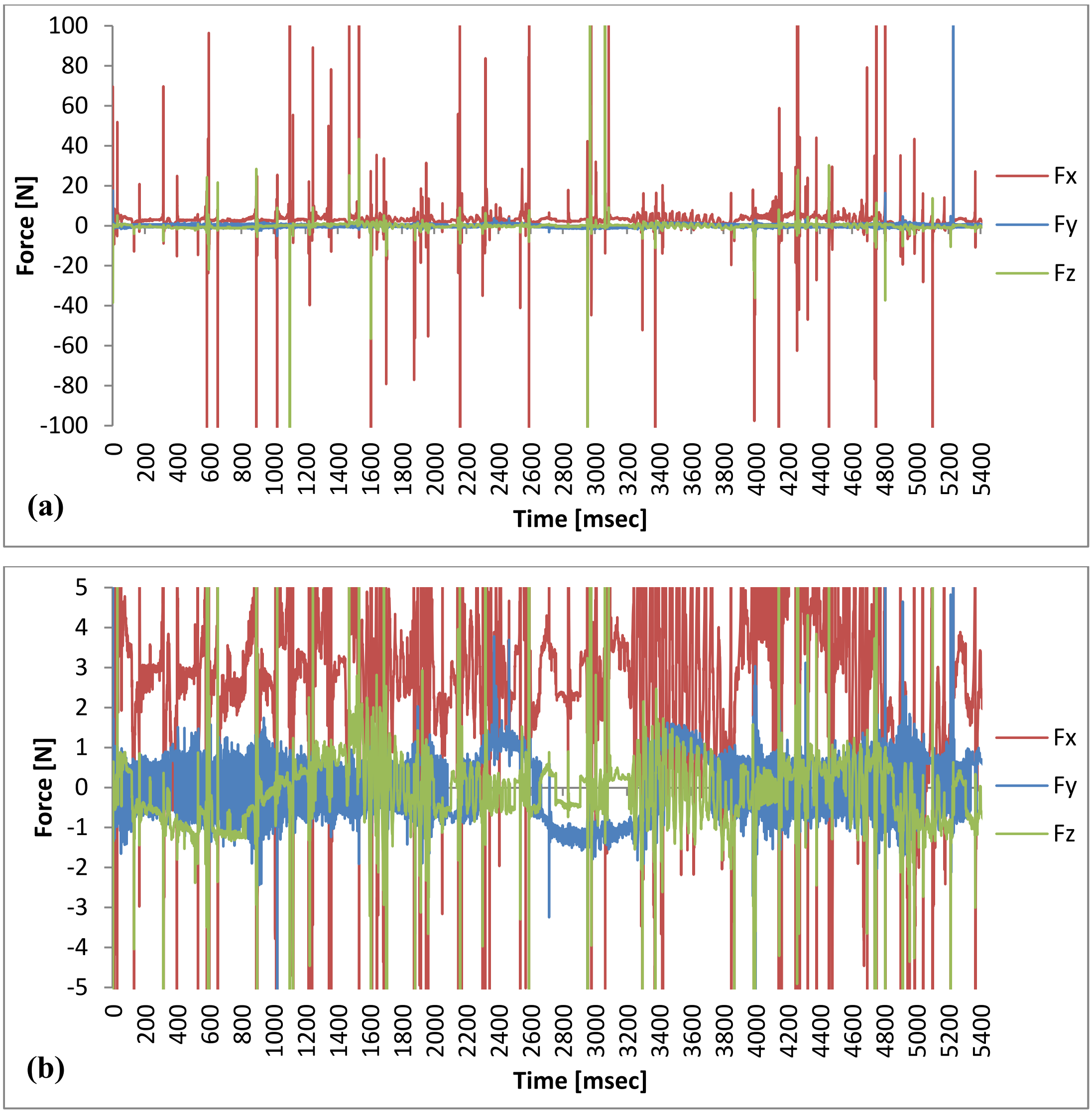


Figure 5.6 : Force graphs for first path, $K_1 = 1$ and $K_2 = 0{,}01$: (a)Computed virtual end-effector force. b)Limited force applied to the haptic device.

Figure 5.5 shows the resulting position, velocity and acceleration values of the experiment 1. There is a small amount of noise on position curves and this shows the instability of the system. Since, the velocity and the acceleration values are obtained by numerical derivation, some peak values are occured on the velocity and acceleration curves. Figure 5.6 shows the resulting force values. In first force graph, numerical calculations caused some peak values. Second graph shows the force values inside the force limits of the application. If the resulting force values are more then 5 N or less than -5 N, 0 N force applied to the user. There are quite a lot of peak values in the first force graph and in second graph, many force values are outside of the force limits. These fluctuations force the user to move the haptic interface point irregularly and this cause instability. Therefore, experiment 1 is quite unstable.

Experiment 2: $K_1 = 1$, 0,001.

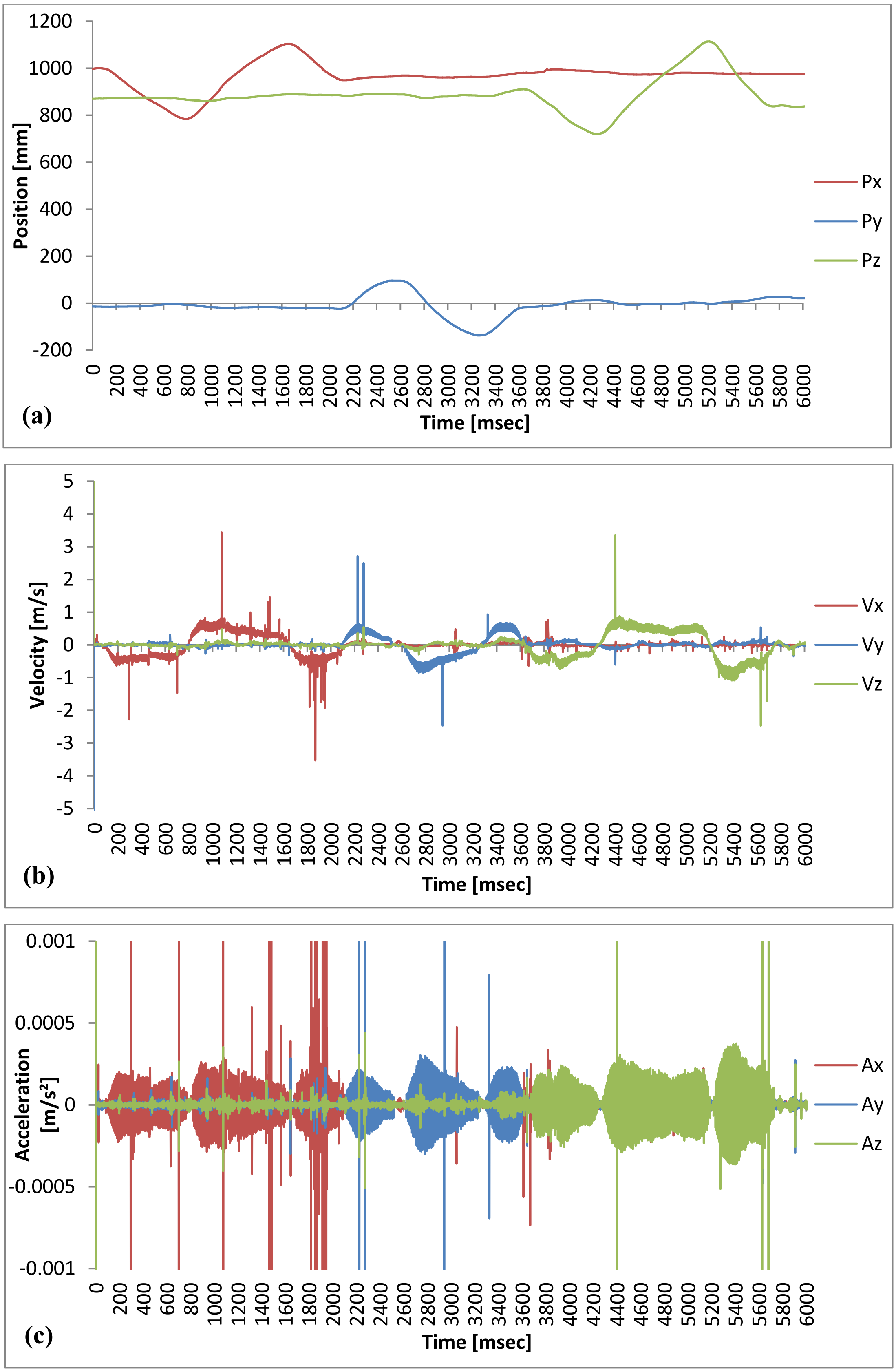


Figure 5.7 : Experimental results for first path, $K_1 = 1$ and $K_2 = 0{,}001$: (a)Position. (b)Velocity. (c)Acceleration.

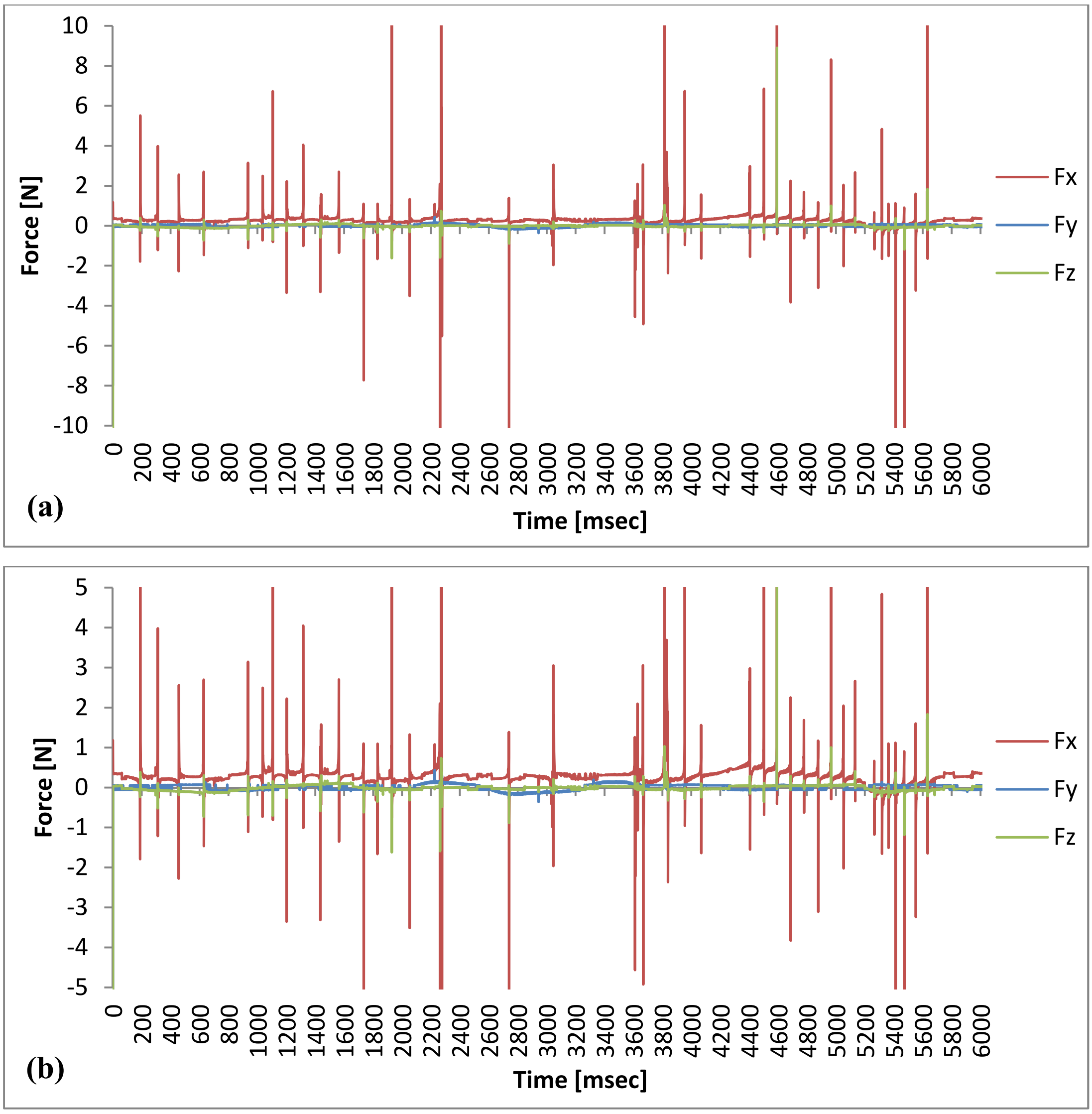


Figure 5.8 : Force graphs for first path, $K_1 = 1$ and $K_2 = 0{,}001$: (a)Computed end-effector force. b)Limited force applied to the haptic device.

In experiment 2, only the force gain K_2 is reduced. The instability is improved as can be shown in position graph in Figure 5.7 and force graphs in Figure 5.8. The position curves are smoother than experiment 1. However, decreasing the force gain cause to reduce the sense of reality.

By improving the stability, the acceleration graph becomes more regular. Thus, acceleration effects on the computed forces can be observed from the acceleration and computed force graphs clearly.

The haptic interface point have to move inside the haptic device velocity limits and the position gain is 1 in this experiment. Therefore, low acceleration values occured in the system.

Experiment 3: $K_1 = 1$, $K_2 = 0{,}0001$.

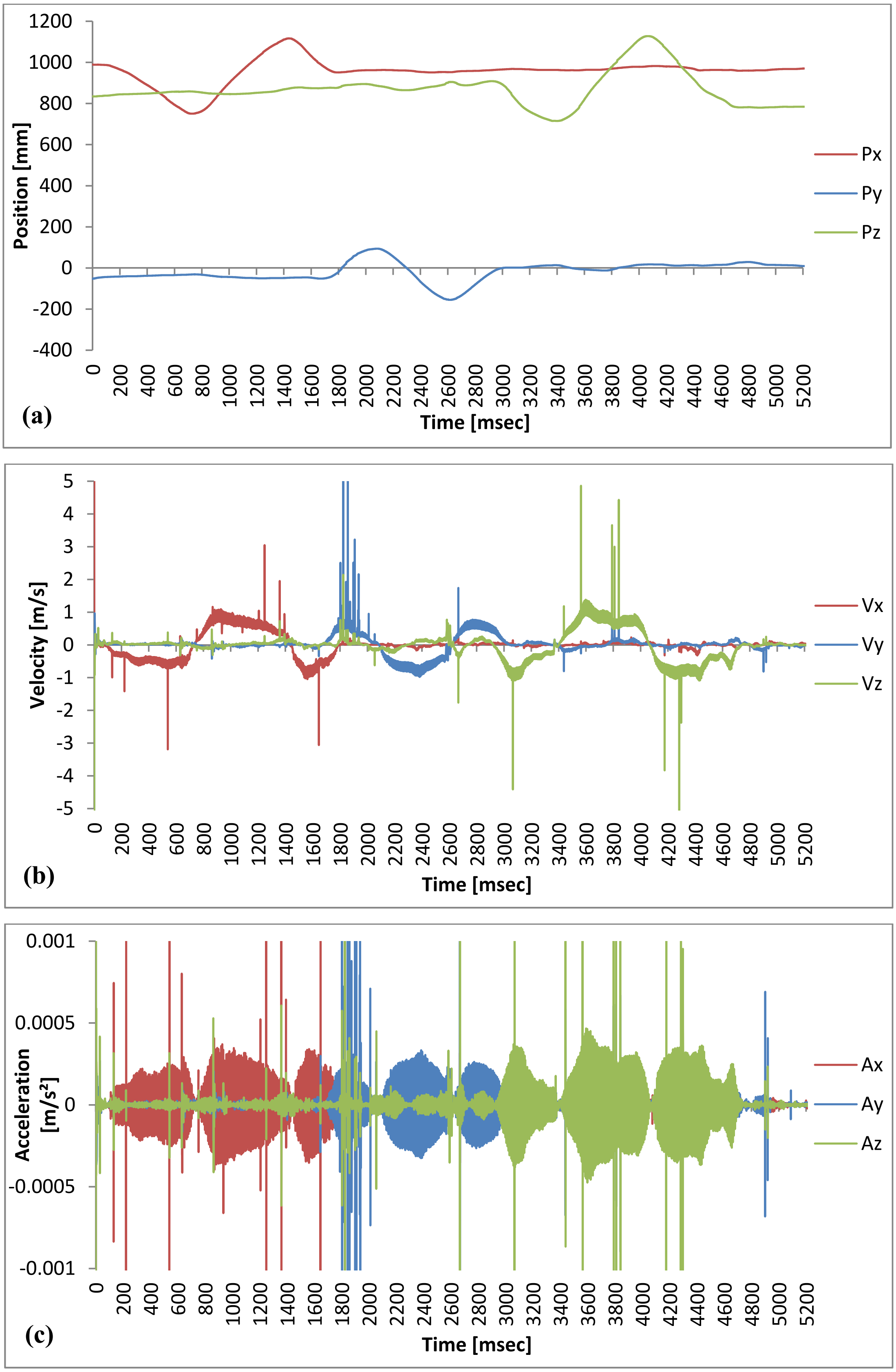


Figure 5.9 : Experimental results for first path, $K_1 = 1$ and $K_2 = 0{,}0001$: (a)Position. (b)Velocity. (c)Acceleration.

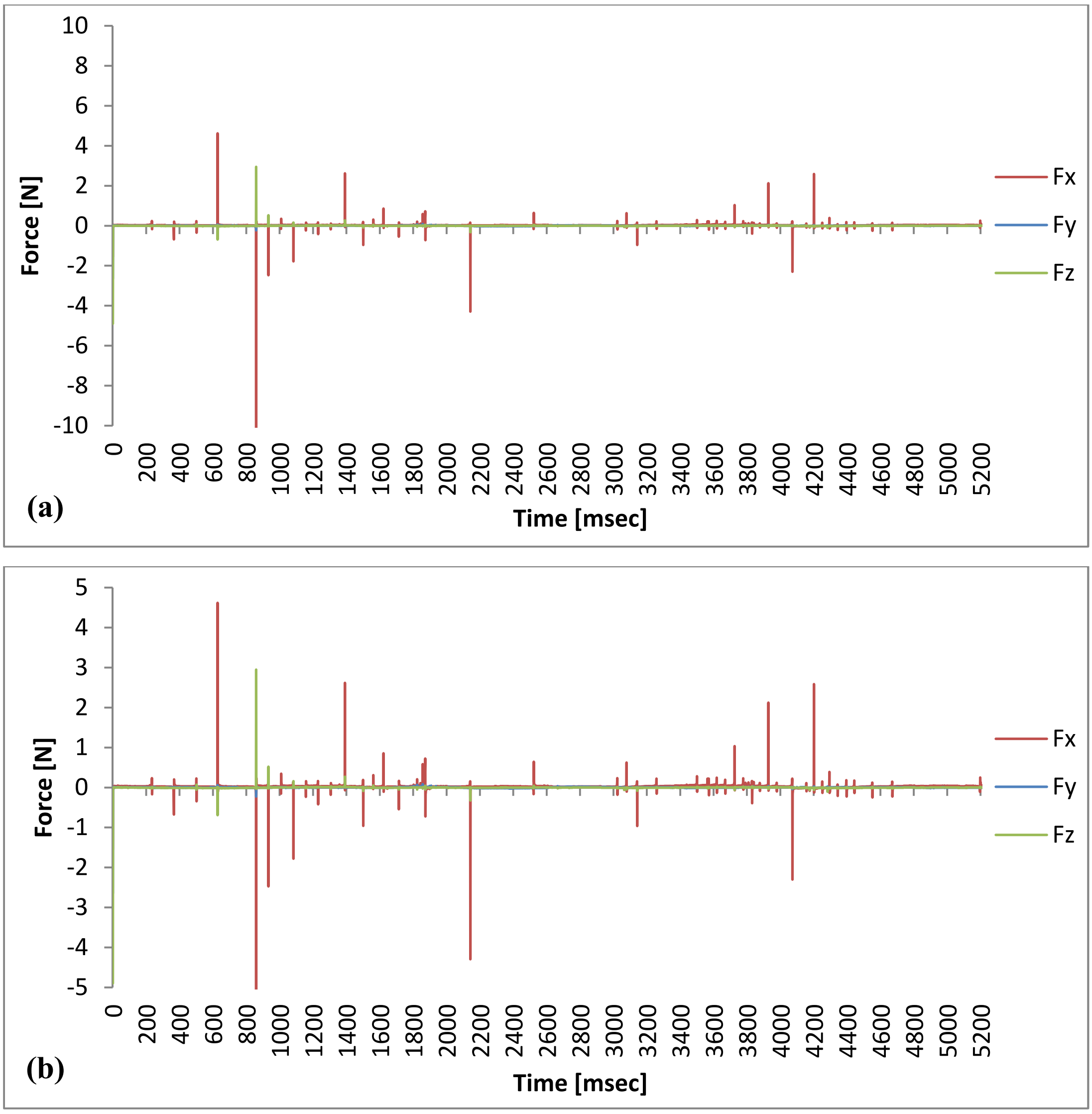


Figure 5.10 : Force graphs for first path, $K_1 = 1$ and $K_2 = 0{,}0001$: (a)Computed end-effector force. b)Limited force applied to the haptic device.

In experiment 3, only the force gain K_2 is decreased again. The system is more stable than experiment 1 and 2 as can be shown in position graph in Figure 5.9 and force graphs in Figure 5.10. However, force magnitudes and the sense of reality are quite reduced.

The numerical computations caused to peak values in the force graphs. Since, the force gain is very low, these peak forces act as force kicking. Because, after applying the force gain to the peak forces, some of them are within the force limits of the application. There are a small number of these forces as can be seen in Figure 5.10. Therefore, they are felt during the experiment, however, they don’t effect the stability of the system significantly.

Experiment 4: $K_1 = 10$, $K_2 = 0{,}01$.

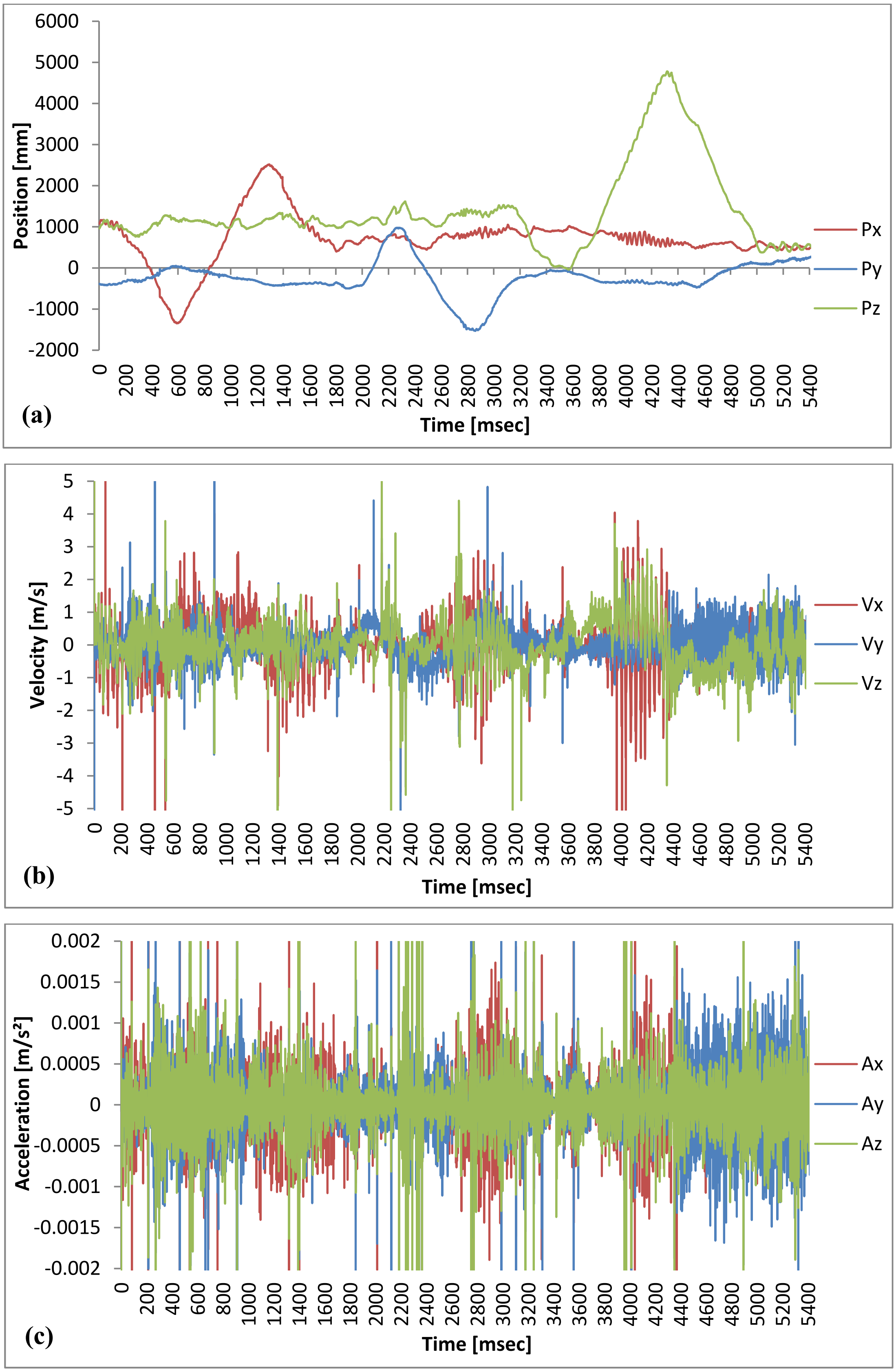


Figure 5.11 : Experimental results for first path, $K_1 = 10$ and $K_2 = 0{,}01$: (a)Position. (b)Velocity. (c)Acceleration.

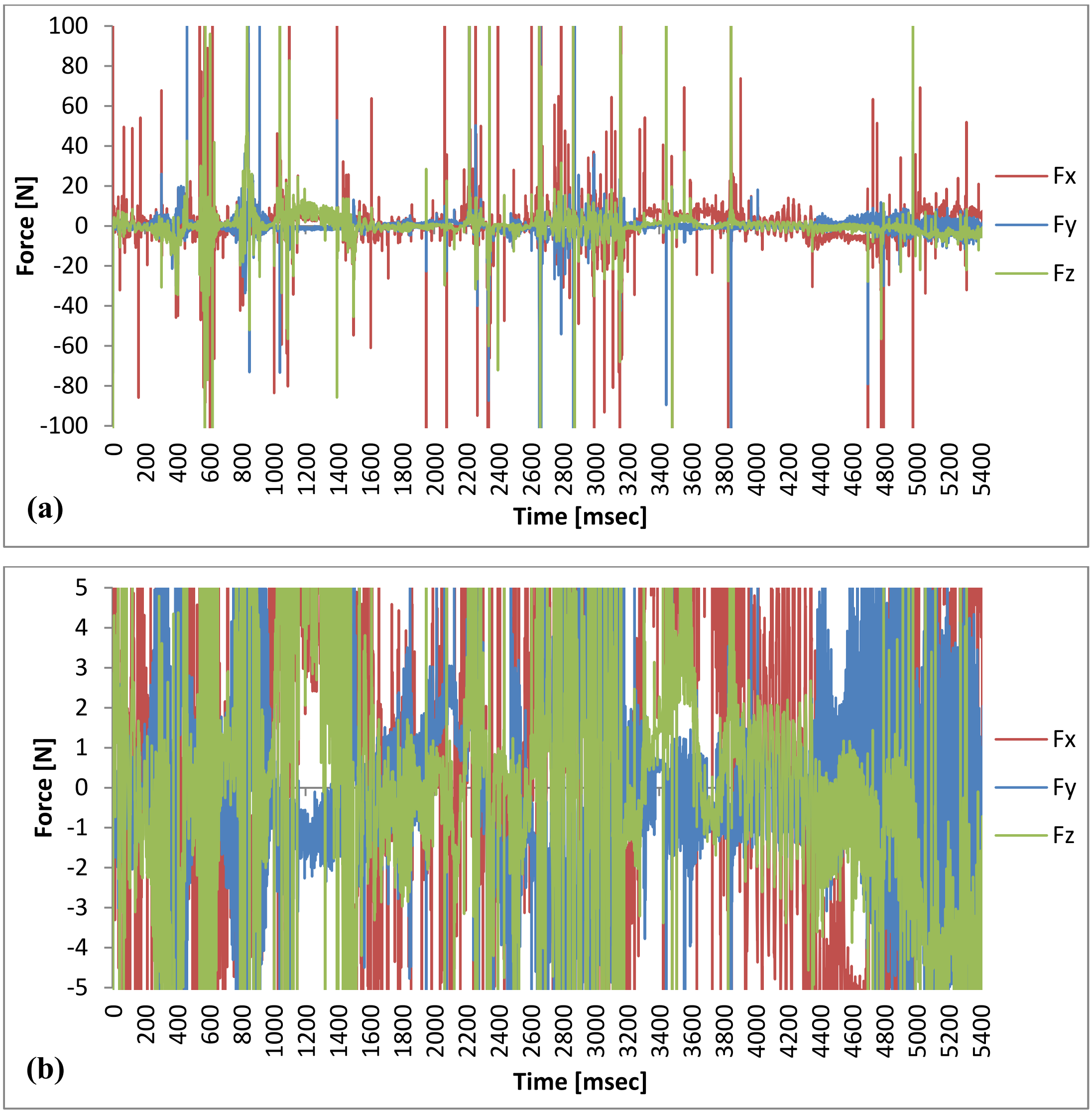


Figure 5.12 : Force graphs for first path, $K_1 = 10$ and $K_2 = 0{,}01$: (a)Computed end-effector force. b)Limited force applied to the haptic device.

In experiment 4, the position gain K_1 is increased for high force gain. Thus, the end-effector of the Staubli manipulator moves 10 times more than the haptic interface point on each axis. Therefore, the velocity and acceleration values of the end effector is increased as can shown in Figure 5.11. Thus, the resulting forces are increased in Figure 5.12.

The position, velocity and acceleration signals are quite noisy. Therewith, the force feedback is quite noisy as well.

Under these conditions, the system is not stable. Consequently, the sense of reality is lost.

Experiment 5: $K_1 = 10$, $K_2 = 0{,}001$.

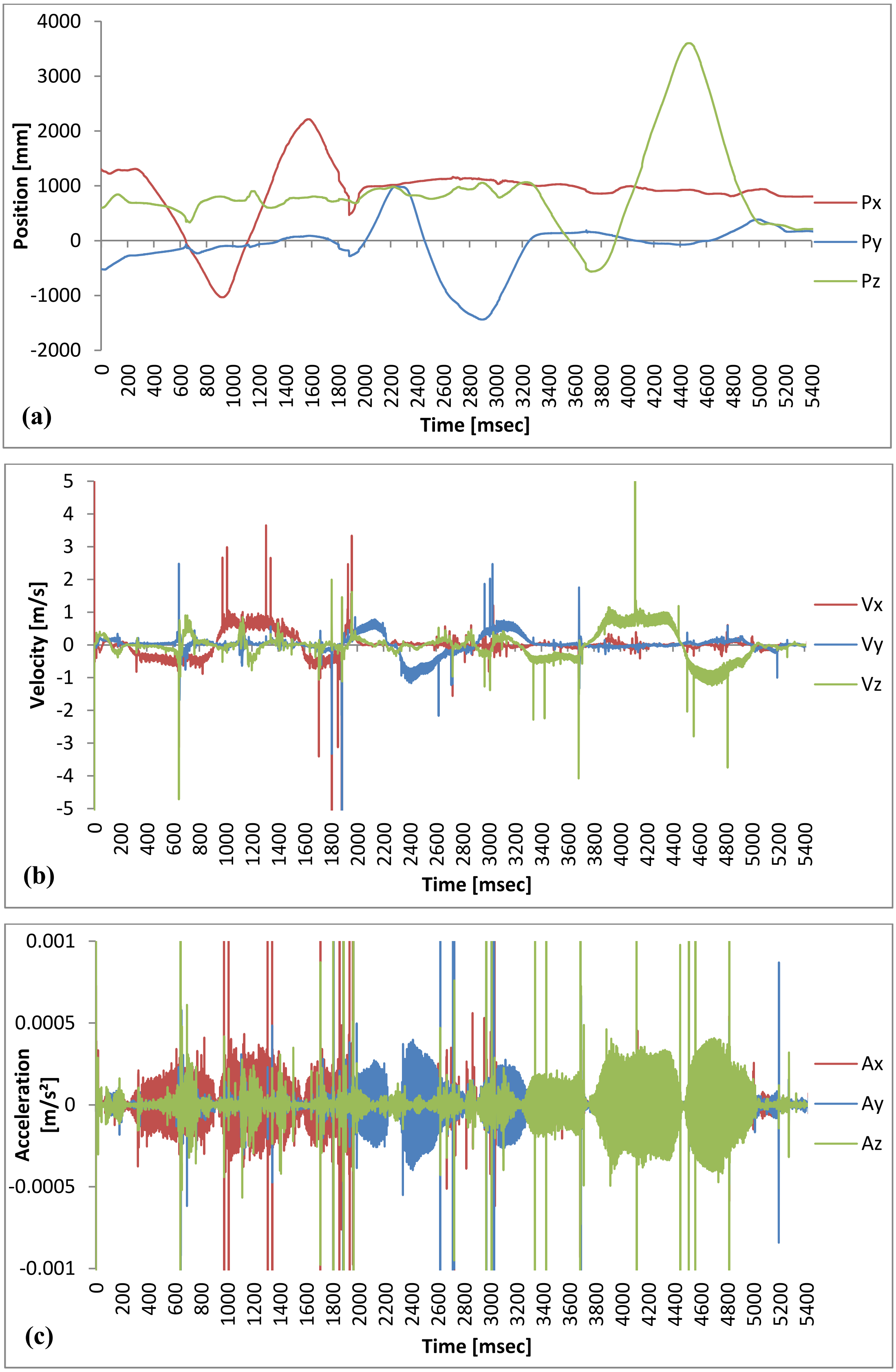
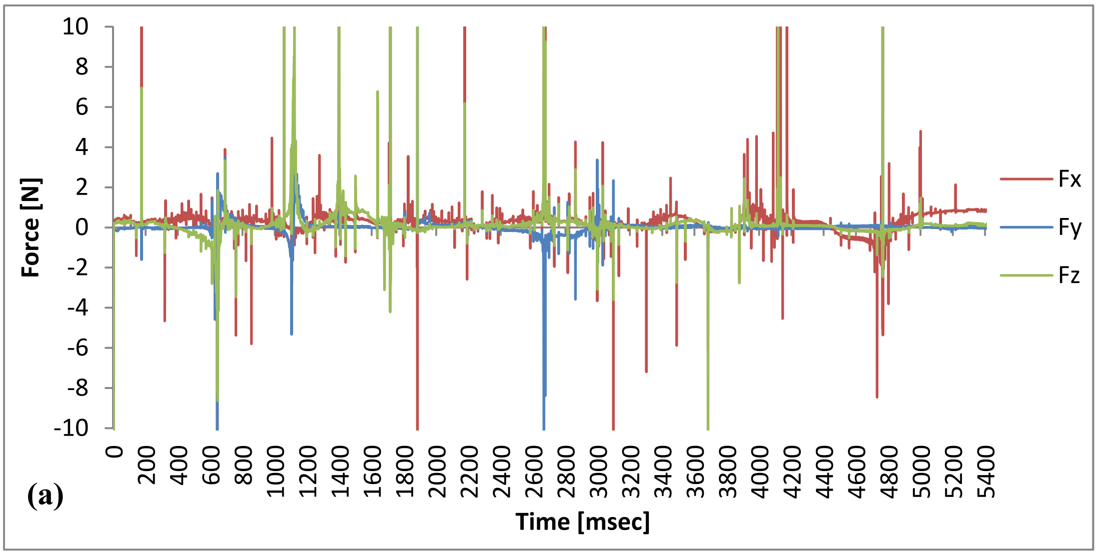


Figure 5.13 : Experimental results for first path, $K_1 = 10$ and $K_2 = 0{,}001$: (a)Position. (b)Velocity. (c)Acceleration.



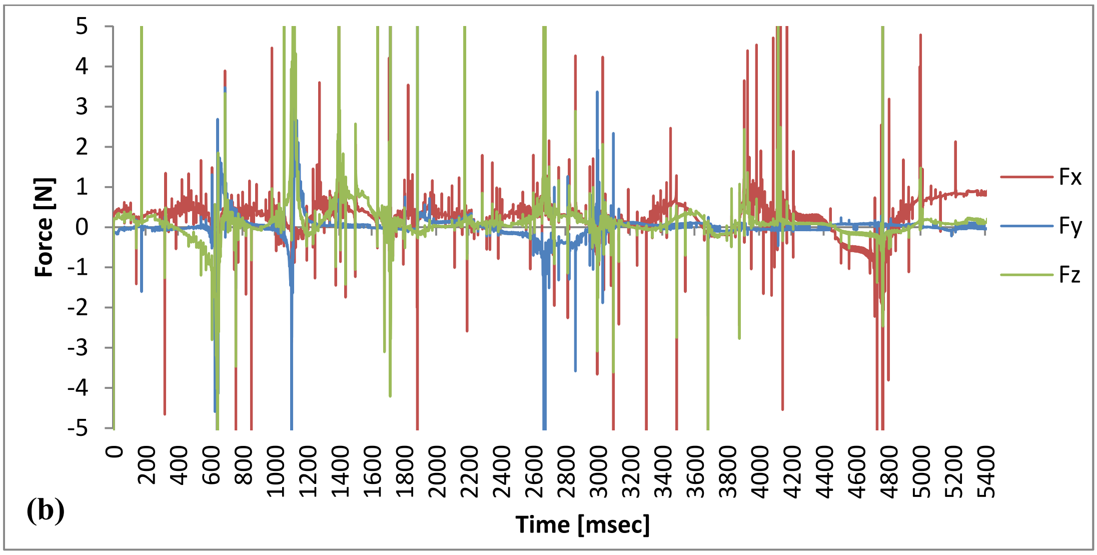


Figure 5.14 : Force graphs for first path, $K_1 = 10$ and $K_2 = 0{,}001$: (a)Computed end-effector force. b)Limited force applied to the haptic device.

By decreasing the force gain, the resulting force values are occurred inside the force limits in experiment 5. However, the system is still quite unstable as can be seen in position graph in the Figure 5.13.

The noises in the velocity and the acceleration graphs are reduced. However, they are not adequate to compute smooth force values. Therefore, the computed force values fluctuate.

In Figure 5.14, the force values inside the force limits are high and abrupt force changes caused a significant amount of force kicking. Consequently, irregular force values occur in the system.

Experiment 6: $K_1 = 10$, $K_2 = 0{,}0001$.

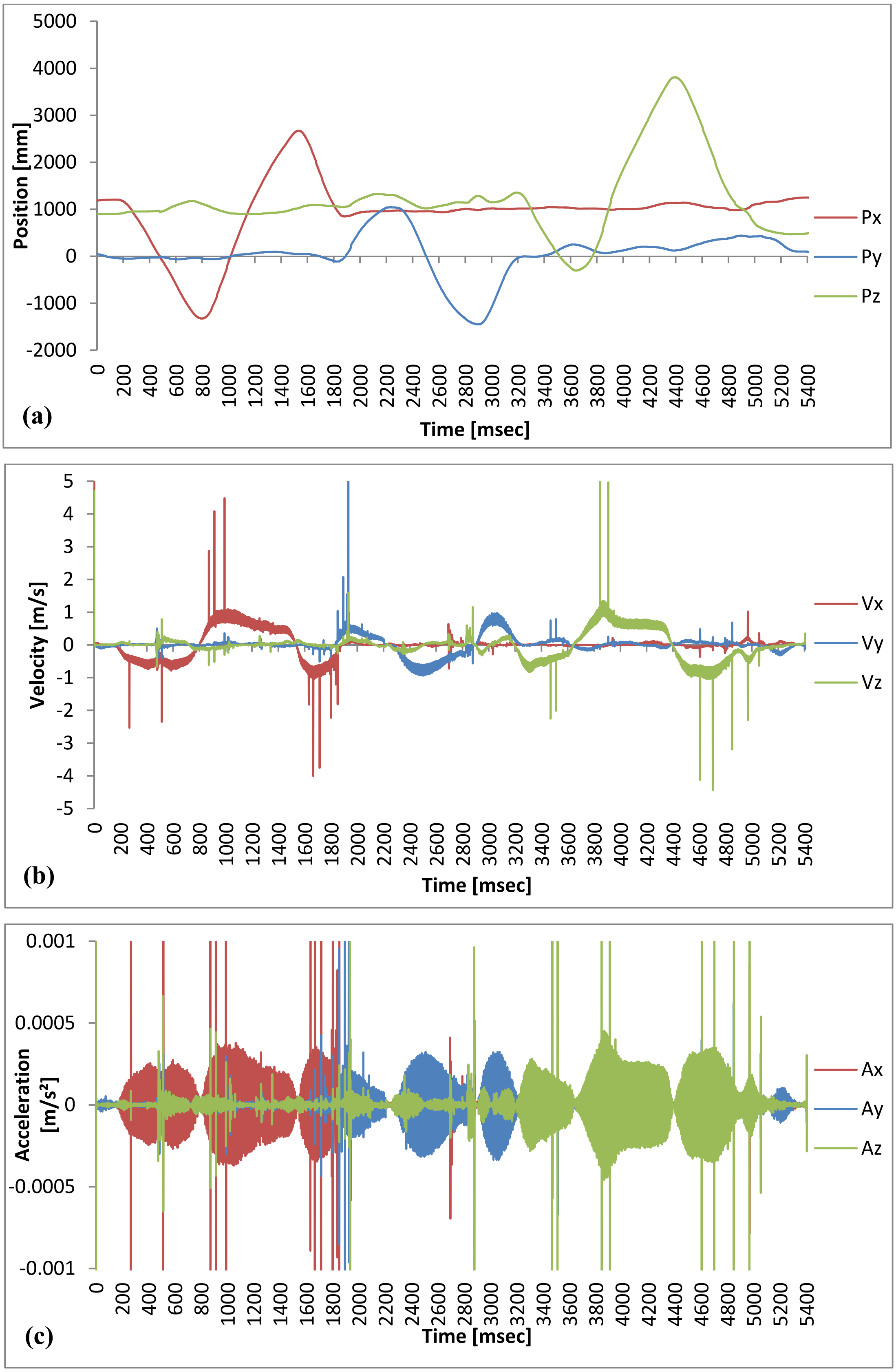


Figure 5.15 : Experimental results for first path, $K_1 = 10$ and $K_2 = 0{,}0001$: (a)Position. (b)Velocity. (c)Acceleration.

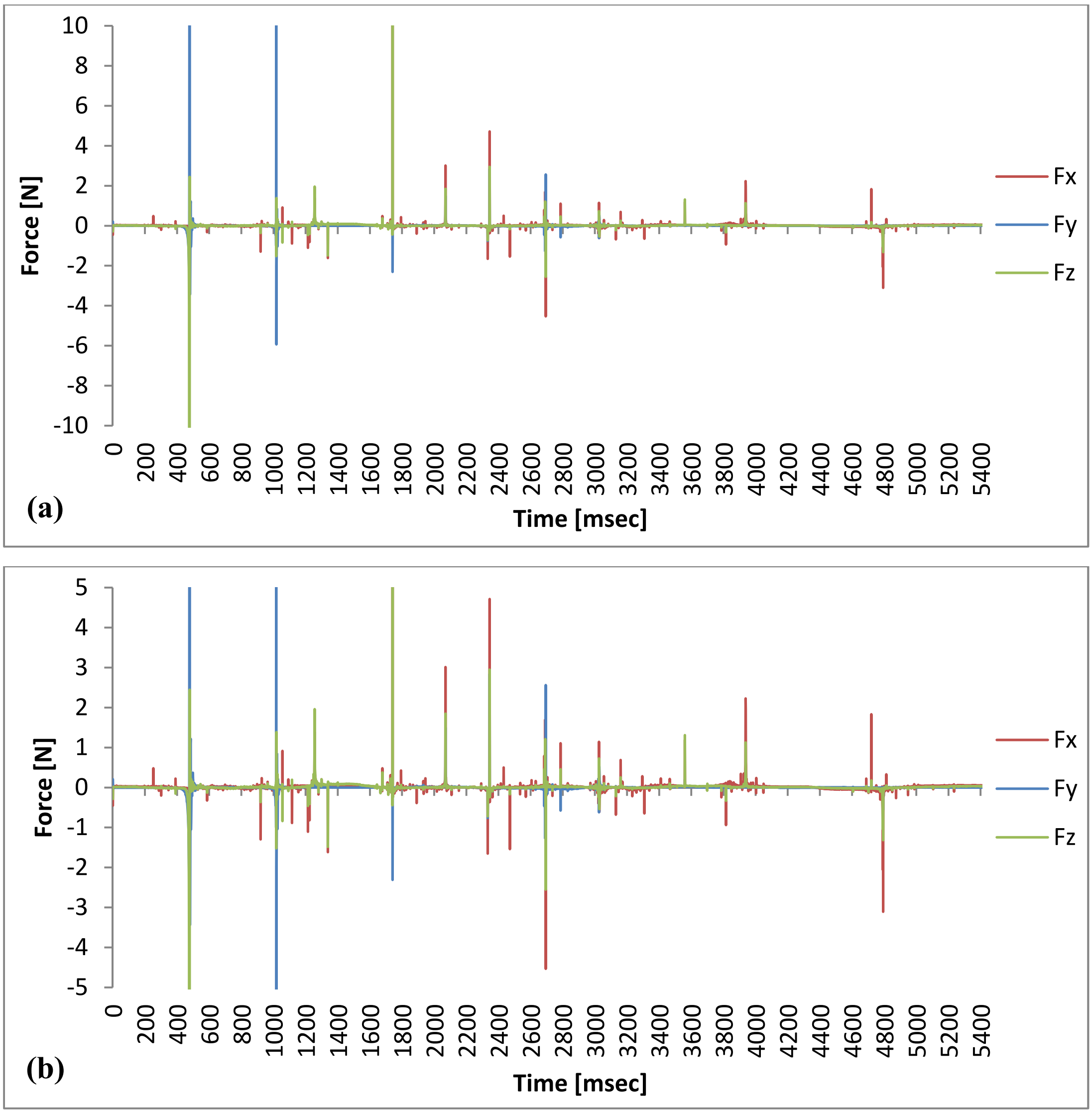


Figure 5.16 : Force graphs for first path, $K_1 = 10$ and $K_2 = 0{,}0001$: (a)Computed end-effector force. b)Limited force applied to the haptic device.

In experiment 6, the force gain is decreased. The system is stable relative to the experiment 4 and 5 as can be seen in the position curves in the Figure 5.15.

The velocity and acceleration values are smoother than experiment 4 and 5. However, they still need to be smoother than experiment 6. The numerical differentiation caused these irregularities in the velocity and the acceleration graphs. They effect the computed force values as well.

Numerical computations caused to occur peak force values as can be seen in Figure 5.16 and they can be felt during the experiment.

Since, the position gain is low, the computed force values are also quite low in this experiment. Therefore, the sense of reality decreased under these conditions.

The second path is a circular path as shown in Figure 5.17. In this path, the haptic interface point is moved to provide the postion change on all axes as shown with the red circle in Figure 5.17.

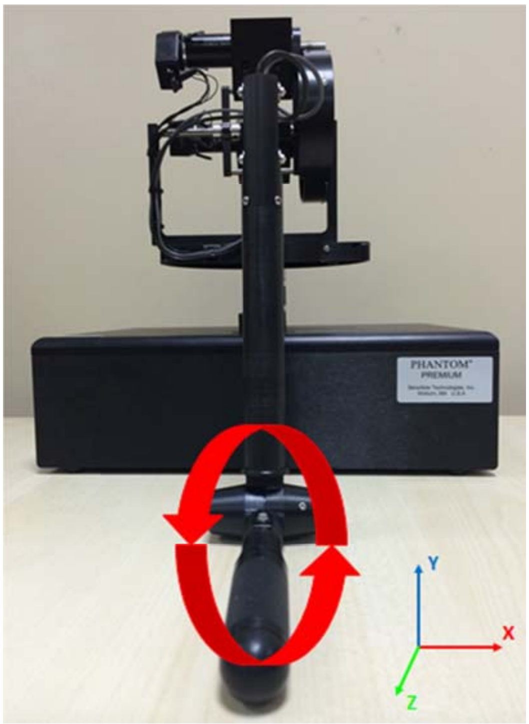


Figure 5.17 : Second experiment path.

In the first path, the haptic interface point is moved on the x, y and z axes to investigate the position, velocity, acceleration and the computed force effects seperately by creating sinusoidal signals for postion projections on all axes seperately.

In this path, a circular motion including movements on all axes simultaneously is determined. Investigating the effect of the sinusoidal position projection signals to each other is the purpose of determining this path.

By moving the haptic interface point on all axes simultaneously, multiple position projection signals are created simultaneously. The position projection signals effect the velocity and the acceleration projection signals. Consequently, complicated force values are computed. Therefore, investigating the projection signals of the position, the velocity, acceleration and the computed forces is giving more information to understand the resulting complicated force values.

Last 6 experiments are realized for this path. The position and force gains are same as the first 6 experiment.

Experiment 7: $K_1 = 1, K_2 = 0{,}01$.

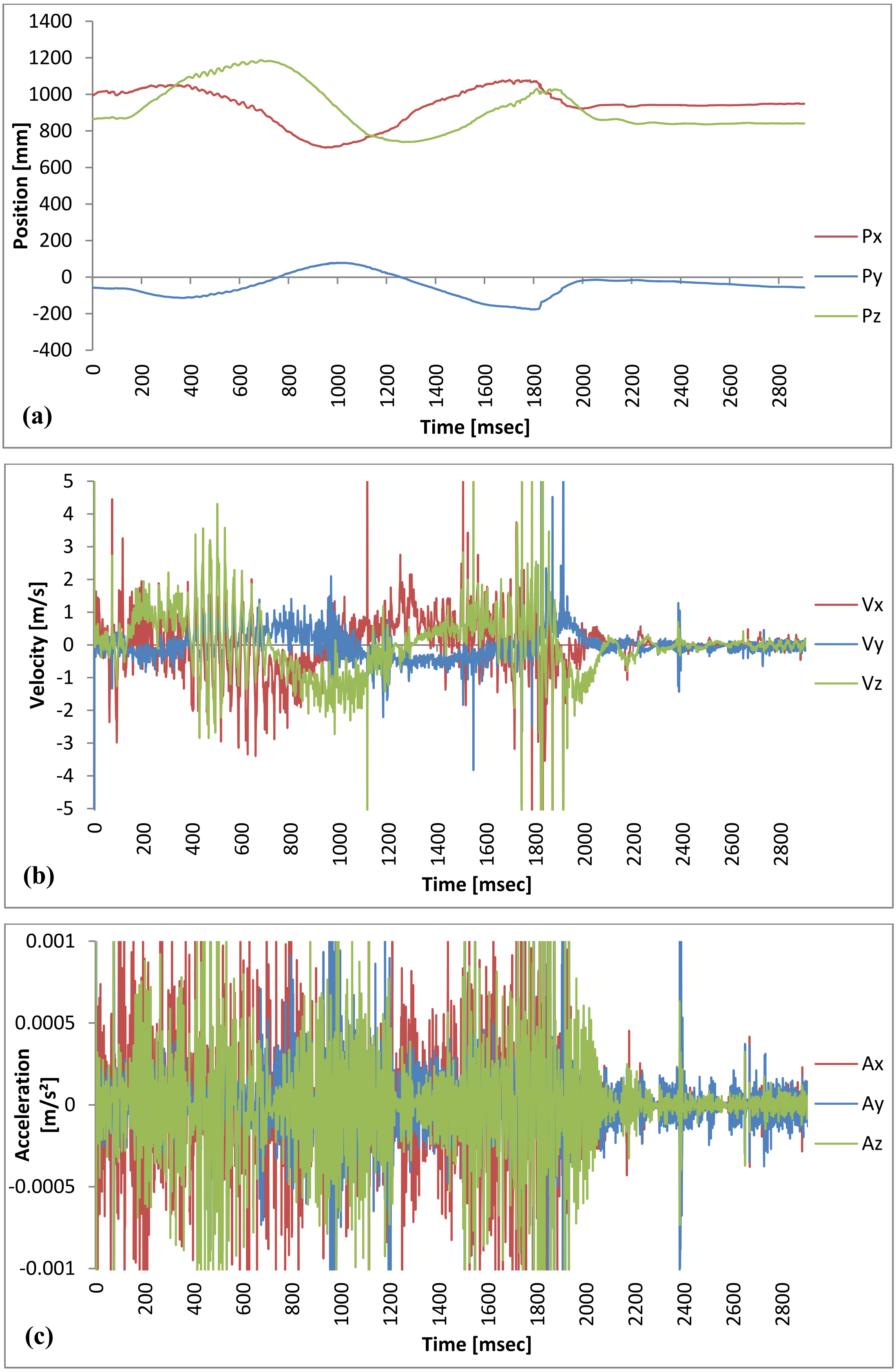


Figure 5.18 : Experimental results for first path, $K_1 = 1$ and $K_2 = 0{,}01$: (a)Position. (b)Velocity. (c)Acceleration.

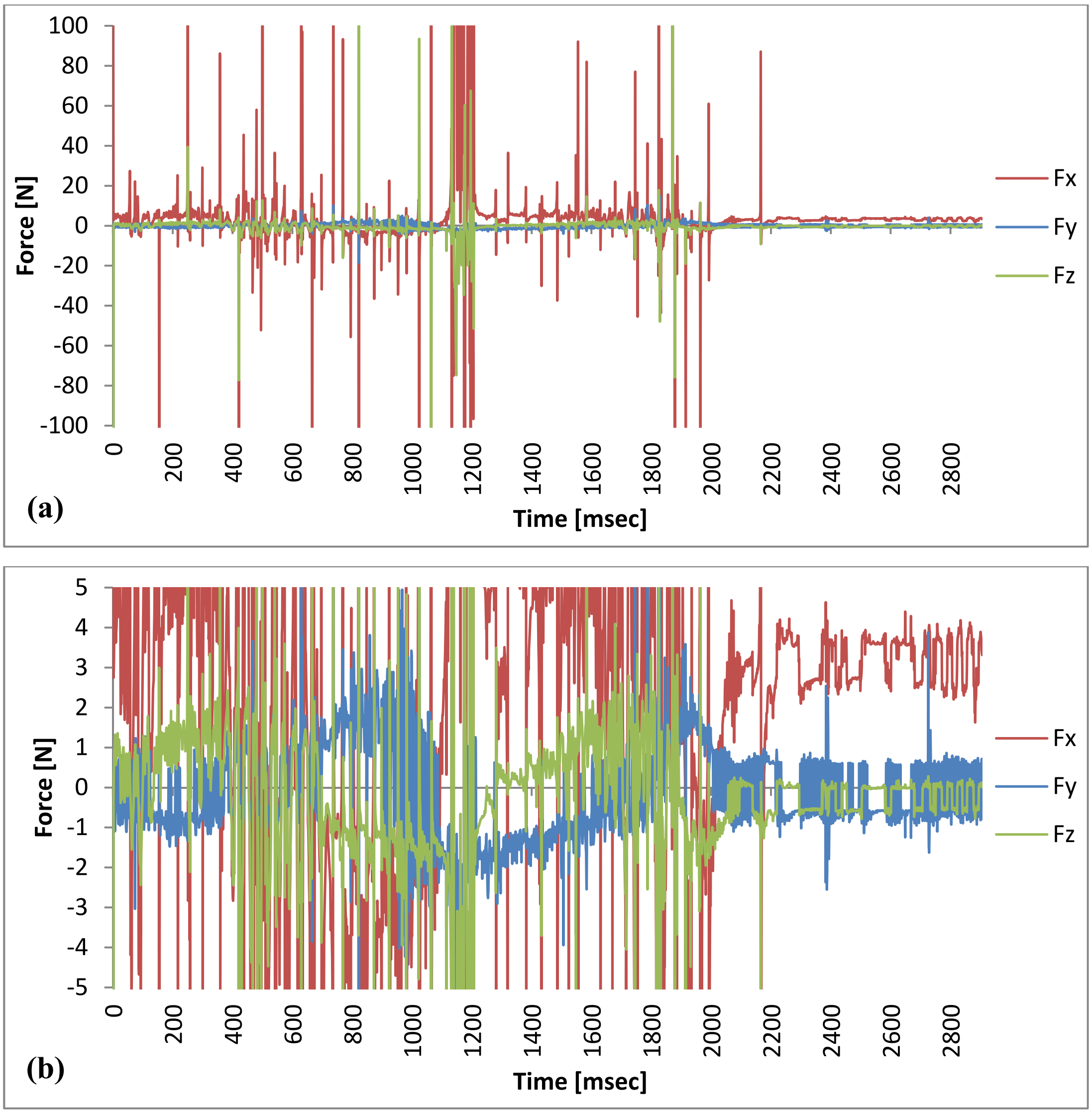


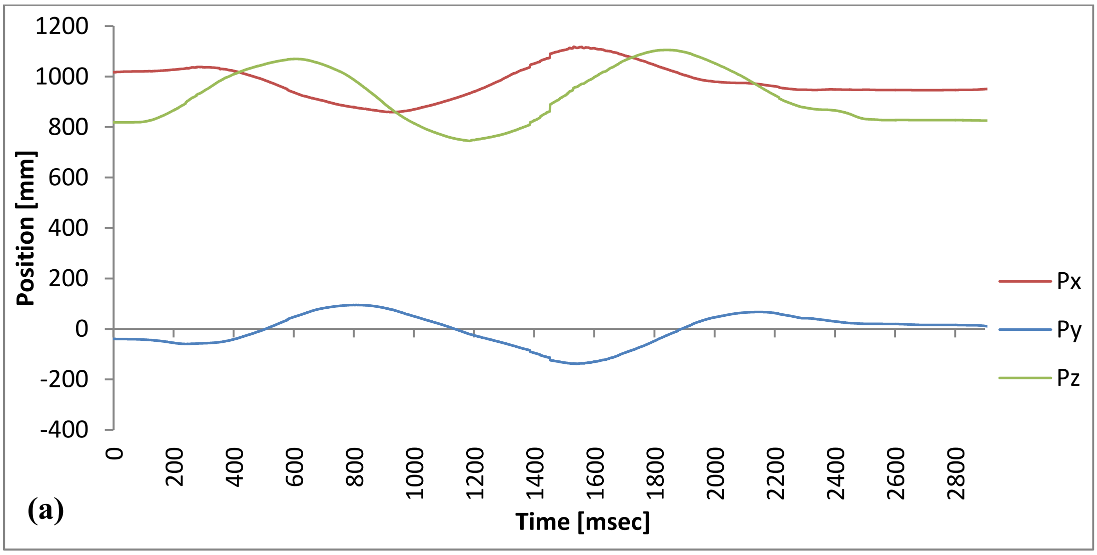
Figure 5.19 : Force graphs for first path, $K_1 = 1$ and $K_2 = 0{,}01$: (a)Computed end-effector force. b)Limited force applied to the haptic device.

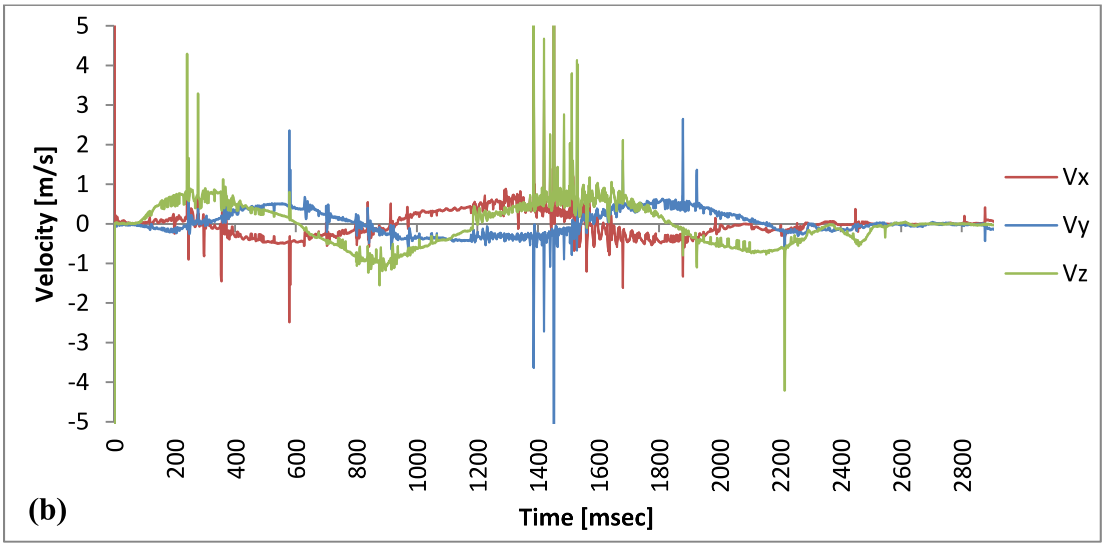
In experiment 7, there are noises on the position signals as can be seen in the position graph in Figure 5.18. The instability caused these noises and they influenced the velocity and acceleration values as can be seen in the velocity and the acceleration graphs in Figure 5.18.

The majority of the computed force values in the y and the z axes are within the application force limits as can be seen in Figure 5.19. Computed force values on the x axis are higher than the y and the z axes.

It can be seen in the Figure 5.19 that after 2200 milliseconds, the haptic interface point stands at approximately a constant point. However, the force values fluctuate. It also shows that the system is not stable adequately.

Experiment 8: $K_1 = 1$, $K_2 = 0{,}001$.





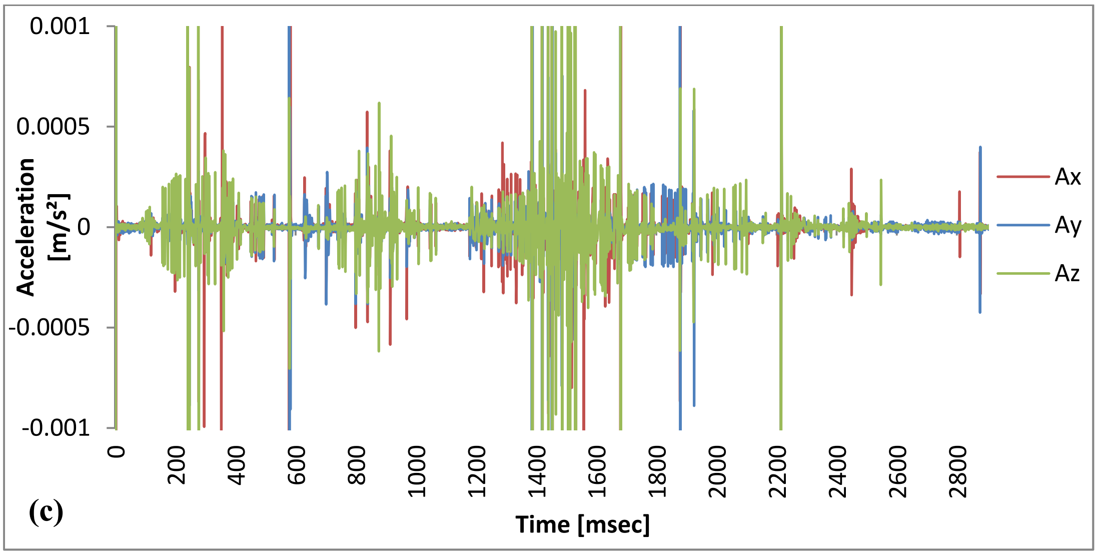


Figure 5.20 : Experimental results for first path, $K_1 = 1$ and $K_2 = 0{,}001$: (a)Position. (b)Velocity. (c)Acceleration.

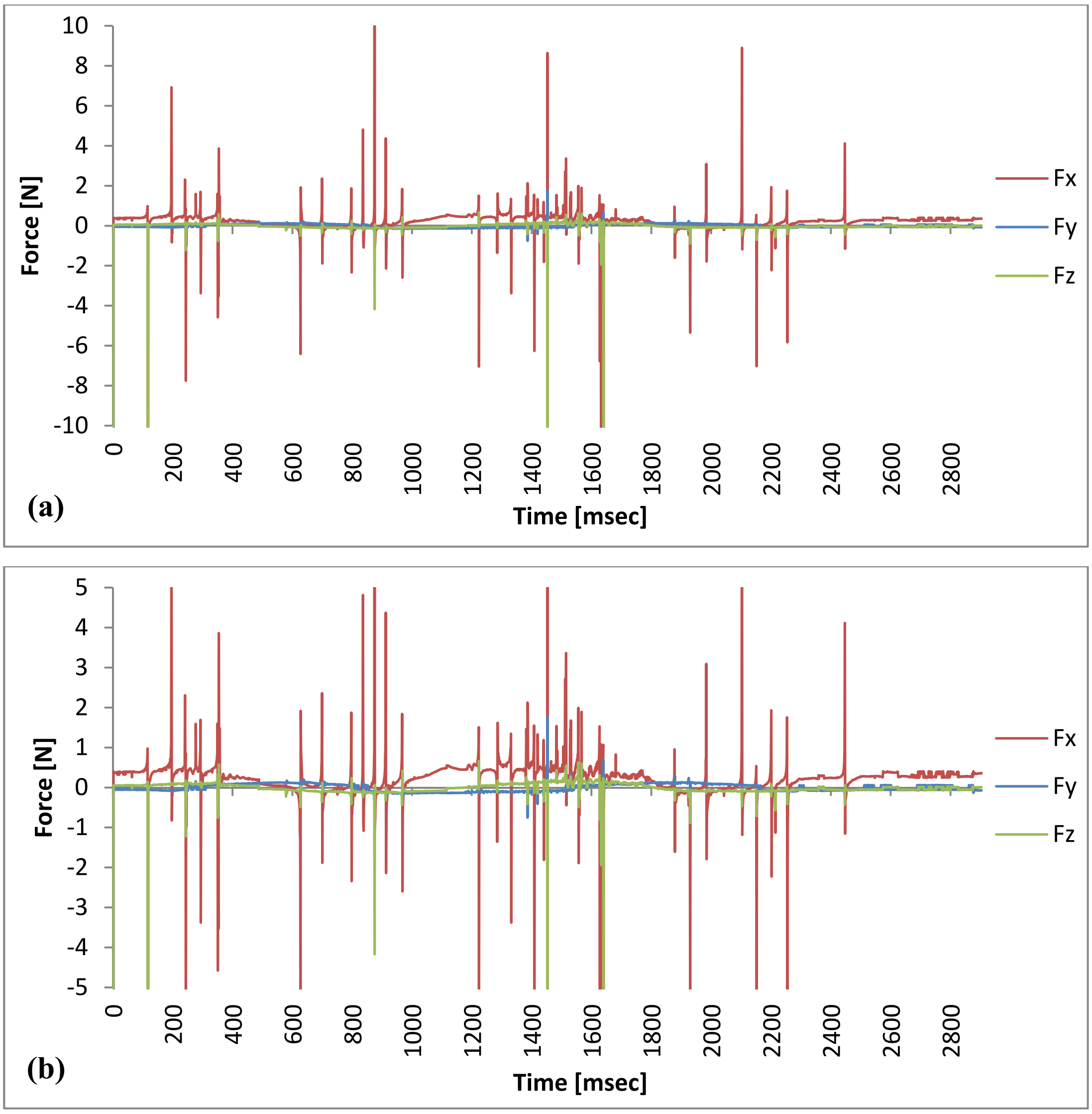


Figure 5.21 : Force graphs for first path, $K_1 = 1$ and $K_2 = 0{,}001$: (a)Computed end-effector force. b)Limited force applied to the haptic device.

In experiment 8, the force gain is reduced and the noise on the position, velocity and acceleration signals are reduced as can be seen in Figure 5.20.

The computed force values are decreased and the numerical computations caused the peak force values.

The complicated motion of the virtual Staubli manipulator results with the complicated force values, which occur at the end-effector. For example, between 1400 and 1600 milliseconds in Figure 5.20 and Figure 5.21, the acceleration values on the z axis are high, however, the computed force values are high on the x axis in same time interval. Therefore, investigating the projection values helps to investigate the effects of the values, which occur on the coordinate axis to each other.

Experiment 9: $K_1 = 1, K_2 = 0{,}0001$.

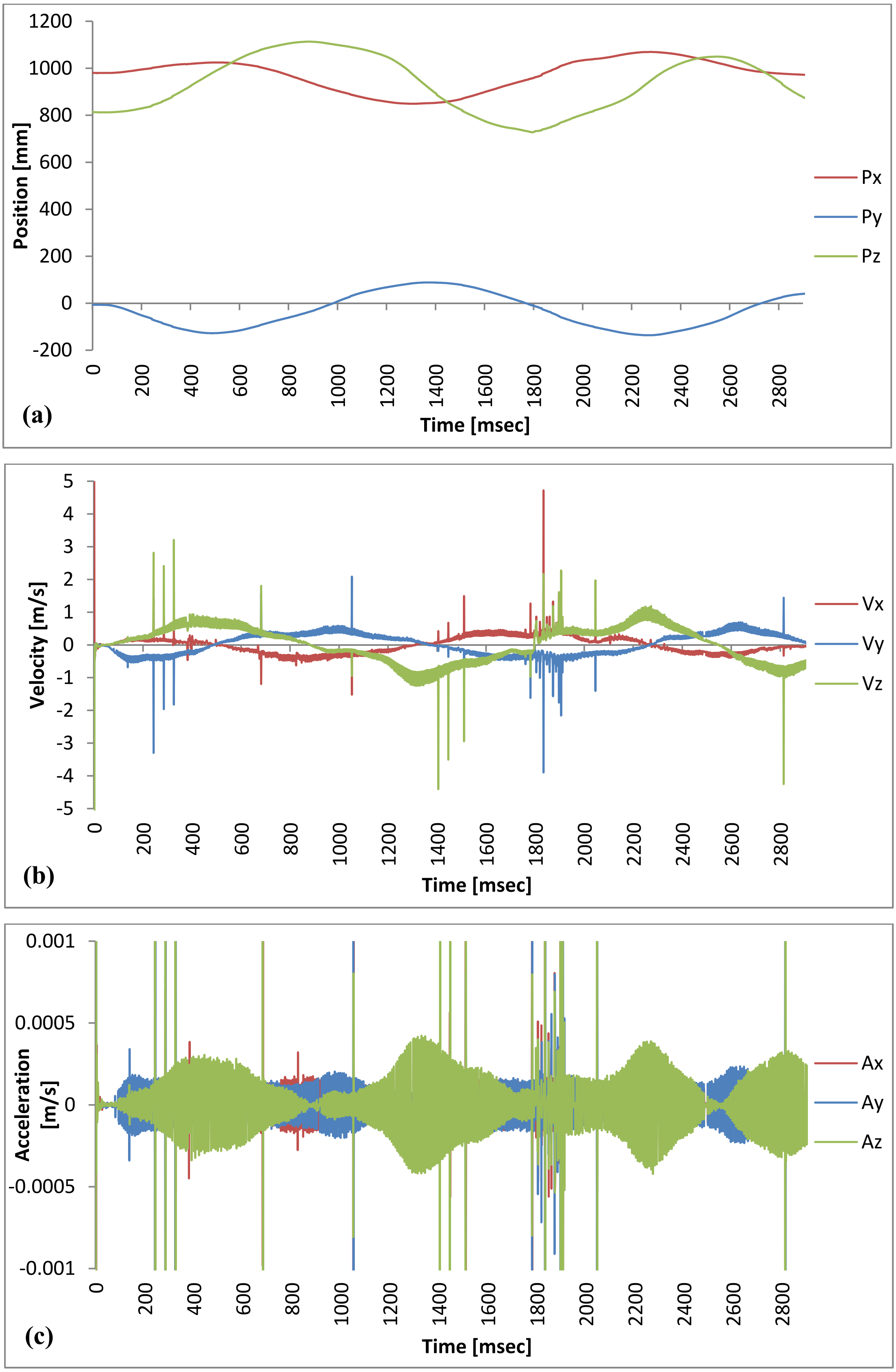


Figure 5.22 : Experimental results for first path, $K_1 = 1$ and $K_2 = 0{,}0001$: (a)Position. (b)Velocity. (c)Acceleration.

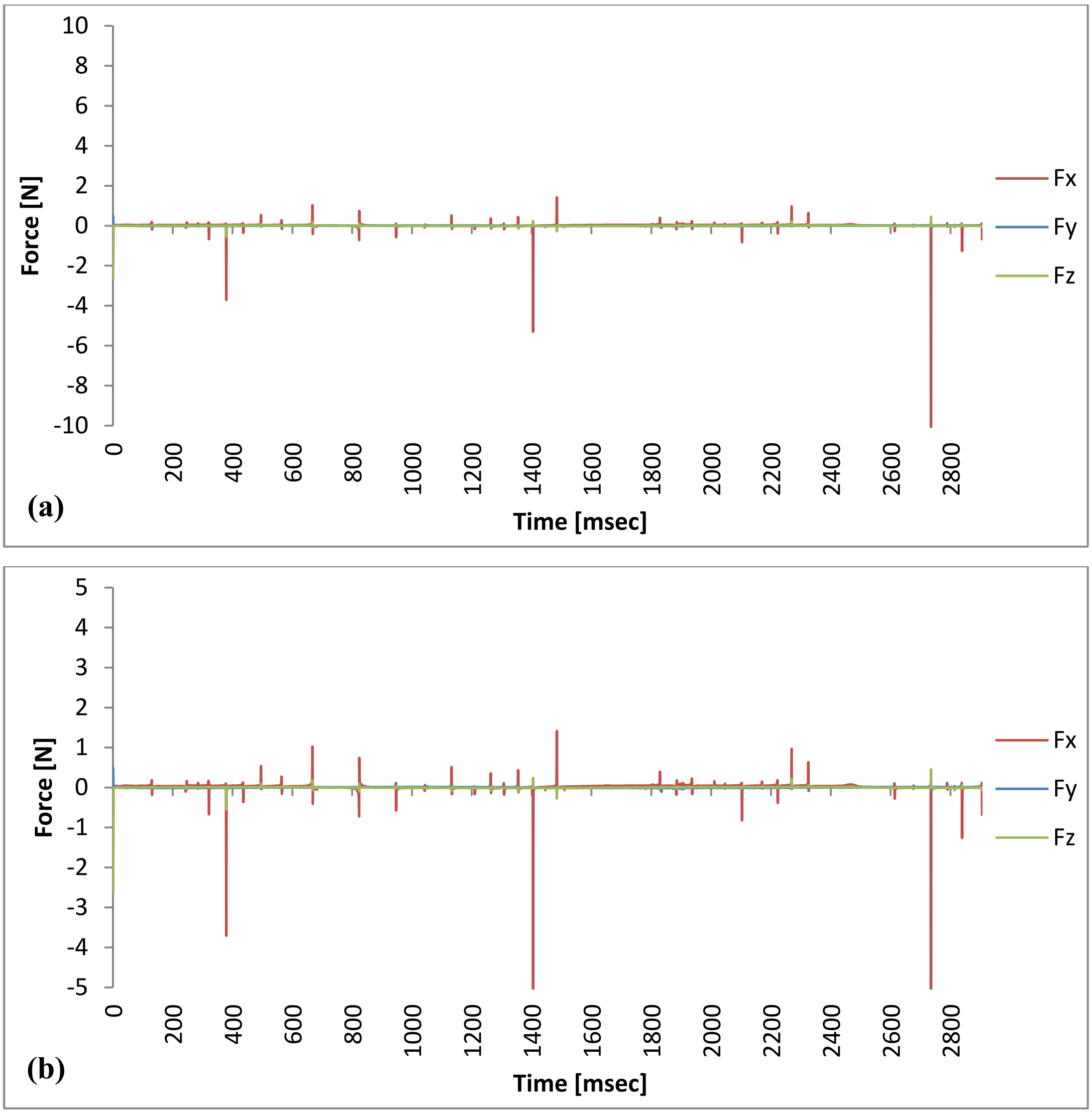


Figure 5.23 : Force graphs for first path, $K_1 = 1$ and $K_2 = 0{,}0001$: (a)Computed end-effector force. b)Limited force applied to the haptic device.

In experiment 9, the force gain is reduced again and the noise on the position, velocity and acceleration signals are reduced as can be seen in Figure 5.22.

The position signal is quite smooth. However, there are noises on the velocity and acceleration signals. These are the results of the numerical differentiation. The effects of the numerical differentiation can be seen in this experiment clearly.

The computed force values are quite reduced and the sense of reality is lost. The numerical computations caused to occur high peak force values. By applying the low force gain to these high peak force values, they reduced within the application force limits. Thus, they caused to abrupt force changes. During the experiment, these peak force values are felt.

Experiment 10: $K_1 = 10$, $K_2 = 0{,}01$.

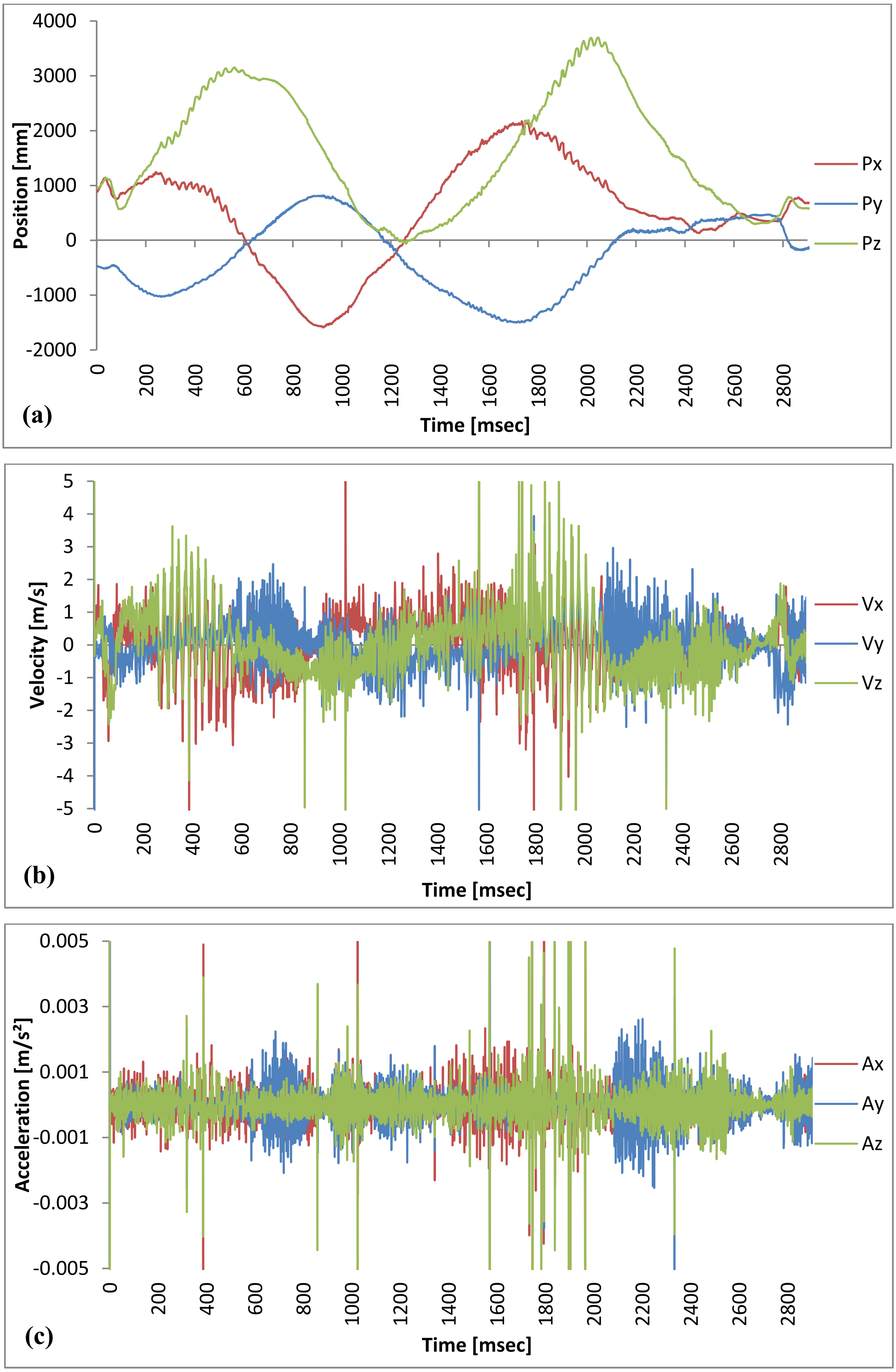


Figure 5.24 : Experimental results for first path, $K_1 = 10$ and $K_2 = 0{,}01$: (a)Position. (b)Velocity. (c)Acceleration.

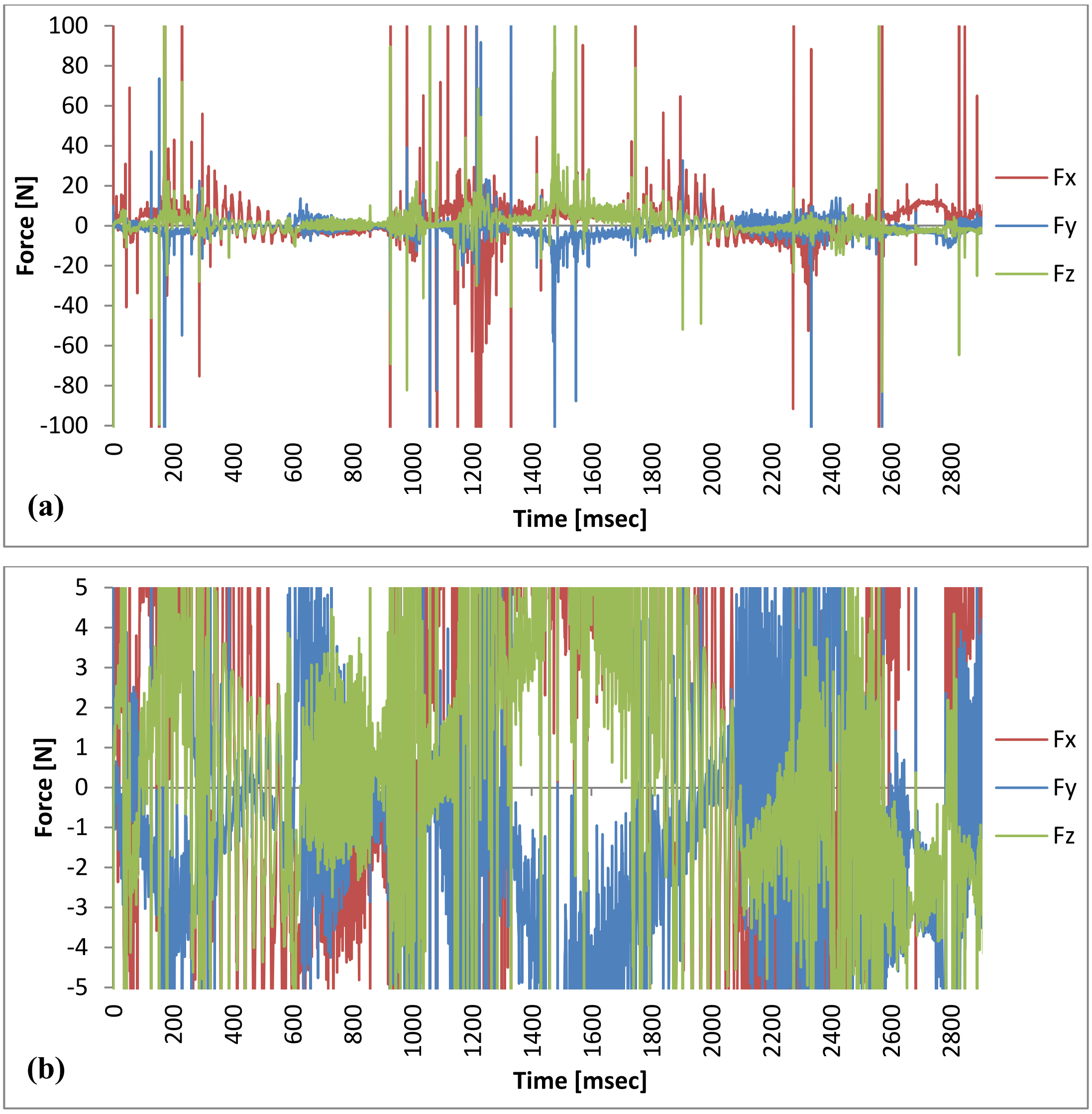


Figure 5.25 : Force graphs for first path, $K_1 = 10$ and $K_2 = 0{,}01$: (a)Computed end-effector force. b)Limited force applied to the haptic device.

In experiment 9, the position gain is increased and the force gain is low. Therefore, there are noises on the position signals as can be seen in the position signals in Figure 5.24 and they influenced the velocity and the acceleration signals.

It can be seen in the position graph in Figure 5.24 that between 200 and 800 milliseconds and between 1800 and 2200 milliseconds, the position is increased on the z axis. The position increase on the z axis influences all the axes. Because there are similar noises on all the axes in these time intervals.

The computed force values are high and the major part of the computed forces are out of the application limits. Therefore, the system is quite unstable under these conditions.

Experiment 11: $K_1 = 10$, $K_2 = 0{,}001$.

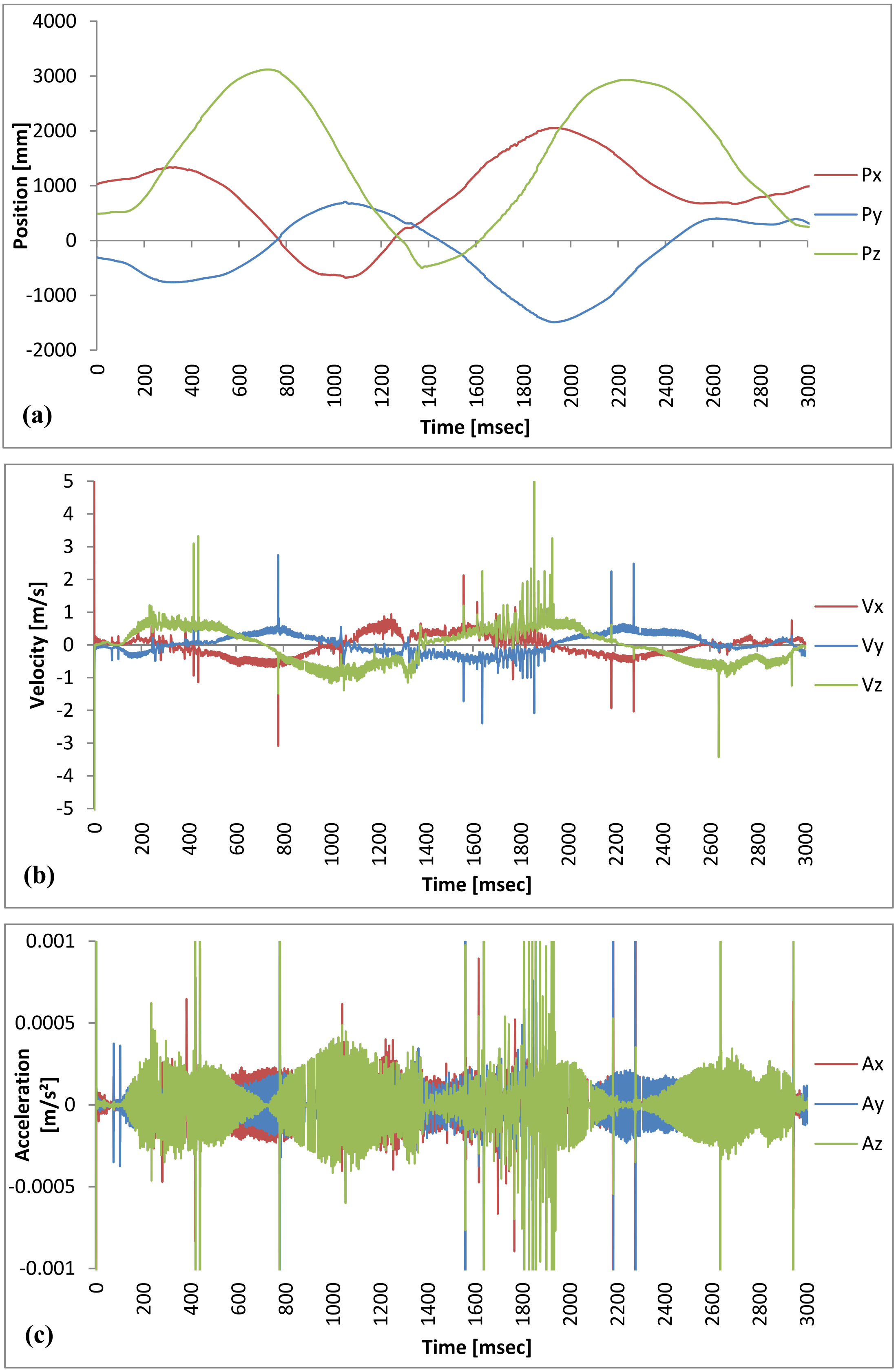


Figure 5.26 : Experimental results for first path, $K_1 = 10$ and $K_2 = 0{,}001$: (a)Position. (b)Velocity. (c)Acceleration.

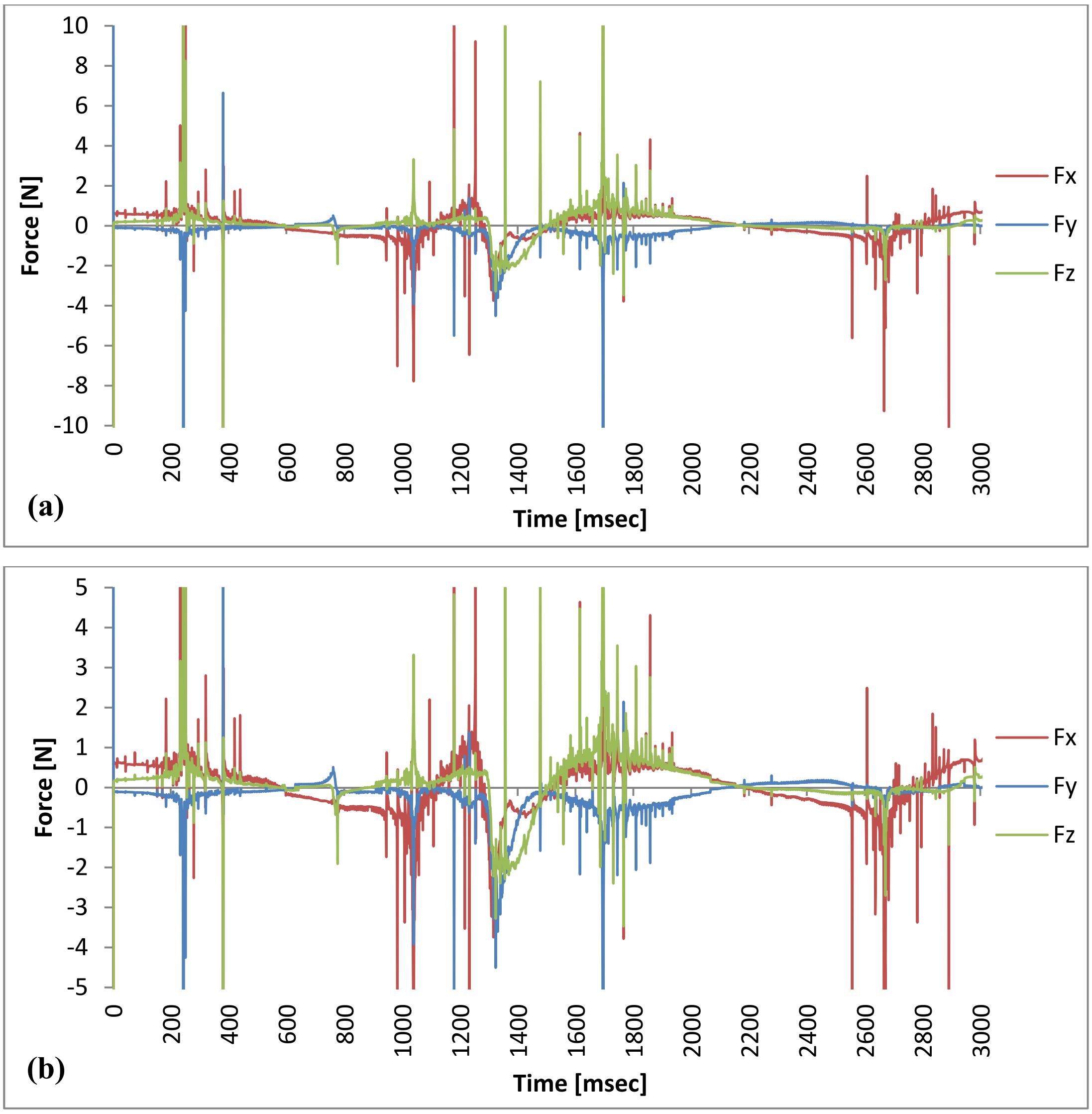


Figure 5.27 : Force graphs for first path, $K_1 = 10$ and $K_2 = 0{,}001$: (a)Computed end-effector force. b)Limited force applied to the haptic device.

In experiment 11, the force gain is increased and the position gain is high. The position curves are quite smooth as can be seen in the position graph in Figure 2.26. However, there are still noises in the velocity and acceleration curves. The numerical differentiation caused these noises.

There are noises in the computed force signals as can be seen in Figure 2.27. The majority of the computed force values are within the application force limits. However, abrupt force changes occur in the system and the numerical computations caused the peak force values.

The position projections are close to the 0 between 1000 and 1400 milliseconds and it results with the force fluctuation in this time interval.

Experiment 12: $K_1 = 10$, $K_2 = 0{,}0001$.

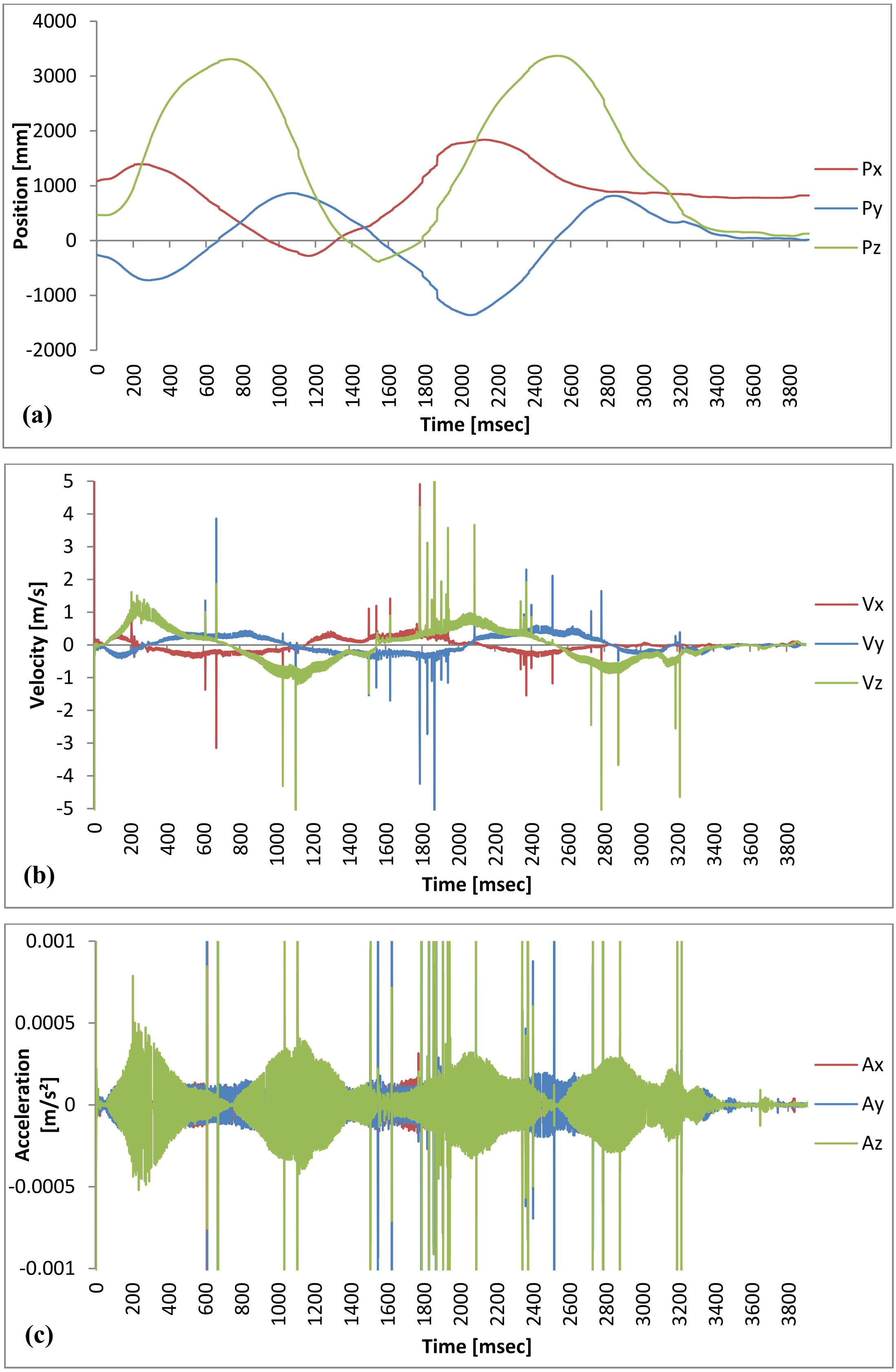


Figure 5.28 : Experimental results for first path, $K_1 = 10$ and $K_2 = 0{,}0001$: (a)Position. (b)Velocity. (c)Acceleration.

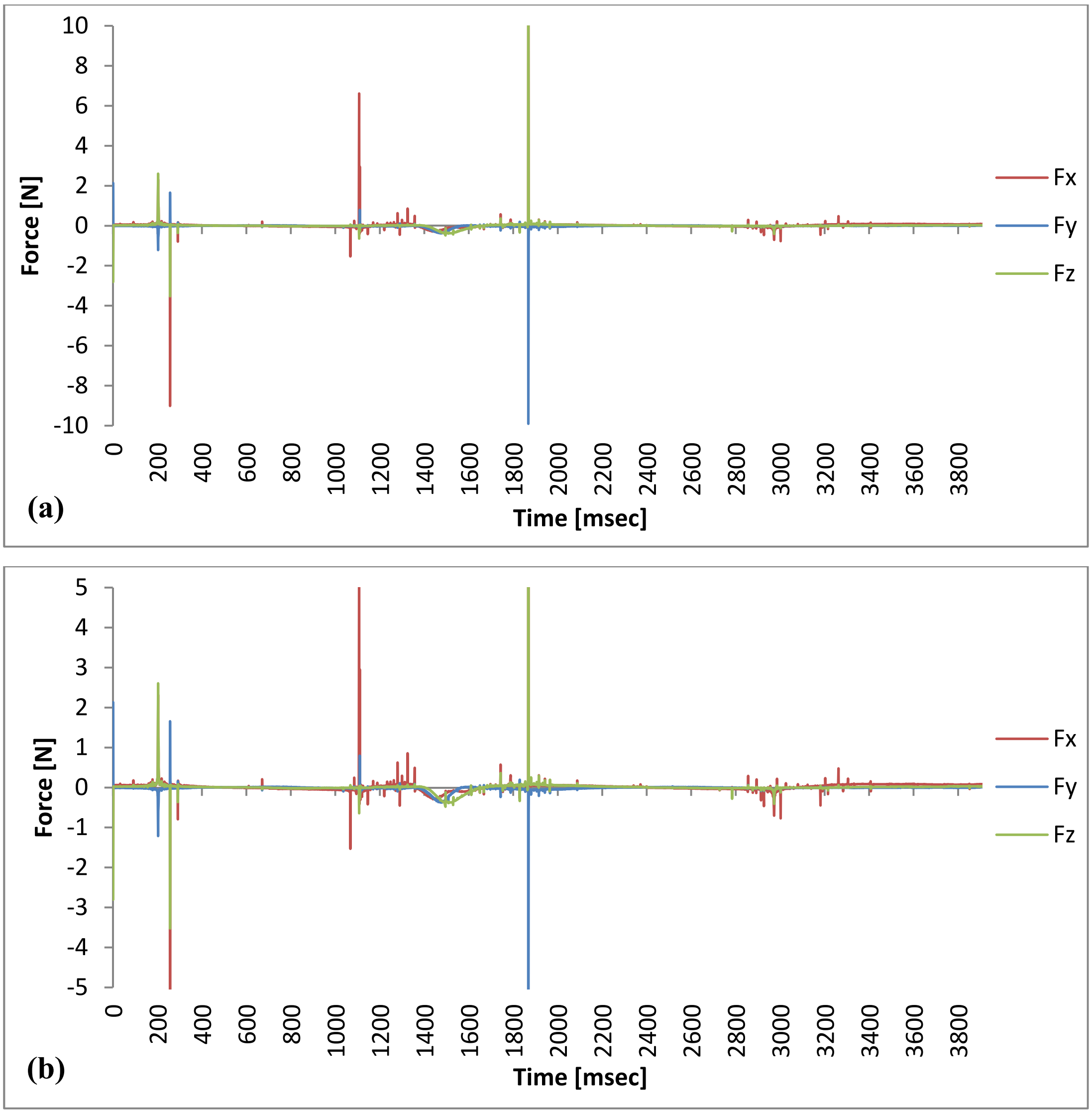


Figure 5.29 : Force graphs for first path, $K_1 = 10$ and $K_2 = 0{,}0001$: (a)Computed end-effector force. b)Limited force applied to the haptic device.

In experiment 12, the force gain is increased again and the position gain is high. The position curves are still quite smooth as can be seen in the position graph in Figure 2.28. There are noies on the velocity and the acceleration signals. The numerical differentiation caused these noises. There are position changes on all the axes simultaneously between 1200 and 1600 milliseconds and all the position projections are close to the 0 in this time interval. This results with the force fluctuations in this time interval as can be seen in Figure 5.29.

The majority of the computed force values are within the application force limits. However, the computed force values are very low. Therefore, the sense of reality is lost under these conditions.

6. CONCLUSION AND RECOMMENDATIONS

An open source and developable computer software is developed for this thesis and future haptics researches. By using this software, the experiments are realized for determined conditions.

The experiments are realized for two different paths. For each path, 1 and 10 position gains and for each poisition gain, 0.01, 0.001 and 0.0001 force gains are used as experiment conditions. Then, resulting position, velocity, acceleration and force values are obtained and plotted with respect to time.

The position and force gain effects on the stability of the system is investigated. Numerical differentiation caused to occur noises on the velocity and acceleration signals and they influenced the stability of the computed force values. Numerical computations caused the peak position, velocity, acceleration and computed force values.

Increasing the position gain causes to occur high velocity and acceleration values. Hence, the high force values are obtained and they cause to instability. Decreasing the force gain improves the stability, however, the sense of reality is reduced.

The instability causes to abrupt force and position changes. Thus, vibration occurs on the system. Furthermore, numerical computations cause the peak forces. The peak forces and abrupt major force changes result in large discontinuities in force magnitude, hence, force kicking occurs in the system.

The vibration is a noise on both the position input and the force feedback signal and it needs to be filtered to improve the stability of the system. To achieve this, diverse filters, which are used in the signal processing applications can be used. Another way is placing a virtual spring-damper system between the haptic interface point and the end-effector of the virtual Staubli manipulator. Thus, the noise, which is occurred on the signals can be eliminated. In addition to these methods, artificial intelligence algorithms can be applied to the system to detect the noise and improve the stability in future works.

REFERENCES

[1] **J. Gamez Garcia, J. Gomez Ortega, A. Sanches Garcia, S. Satorres Martinez.** (2010). Open Software Architecture for Advanced Control of Robotic Manipulators in Non-structured Environments, ICRA 2010 Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications.

[2] **Emmanuel Nuno and Luis Basanez**. (2006). Bilateral Haptic Guided Robot Teleoperation Via Packet Switched Networks Using Wave Variables with Impedance Adaptation.

[3] **Daniel Kubus and Friedrich M. Wahl.** (2009). Scaling and Eliminating Non-Contact Forces and Torques to Improve Haptic Feedbak Teleoperation, Intelligent Robots and Systems, IEEE/RSJ International Conference.

[4] **Carlos Vasquez and Jan Rosell.** (2007). Haptic Guidance Based on Harmonic Functions for the Execution of Teleoperated Assembly Tasks, IFAC Workshop on Intelligent Assembly and Disassembly, Spain.

[5] **Jaeheung Park, Oussama Khatib.** (2006). A Haptic Teleoperation Based on Contact Force Control, Artificial Intelligence Laboratoryi Stanford University.

[6] **Mandayam A. Sirinivasan**. What is haptics?.

[7] **Staubli.** (2008). Arm – RX series 160 family, Instruction manuel.

[8] **Sensable Technologies, Inc.** (2011). OpenHaptics Toolkit version 3.1 Programmer’s Guide.

[9] **Geomagic.** (2013). Phantom Haptic Devices, Phantom Premium 1.5 6 DOF/1.5 HF 6 DOF Device Guide.

[10] **Abdulmotaleb El Saddik, Mauricio Orozco, Mohamad Eid, Jongeun Cha**. (2011). Haptics Technology: Bringing Touch to Multimedia.

[11] **Kenneth Salisbury, Francois Conti, Frederico Barbagli.** (2014). Haptic Rendering: Introductory Concepts.

[12] **Petr Kadlecek**. Haptic rendering for 3/6-DOF haptic devices, 04/2013.

[13] **Url-1** <*https://www.opengl.org/wiki/Rendering_Pipeline_Overview*>, accessed at 08.04.2014.

[14] **W. Khalil, E. Dombre.** (2002). Modeling, Identification and Control of Robots.

[15] **Egemen Zengin**. (2013). Staubli RX160 Manipülatörün Modellenmesi, Tanınması ve Kontrolü, Istanbul Technical University, Istanbul.

[16] **J. Y. C Luh, M. W. Walker, R. P. C. Paul.** (1980). On-Line Computational Scheme for Mechanical Manipulators. Journal of Dynamic Systems, Measurement and Control, 102(2), 69-75.

APPENDICES

APPENDIX A: QuickHaptics micro API classes and properties

APPENDIX B: Resulting position, velocity, acceleration and computed force values (Magnitudes) of the experiments

APPENDIX A

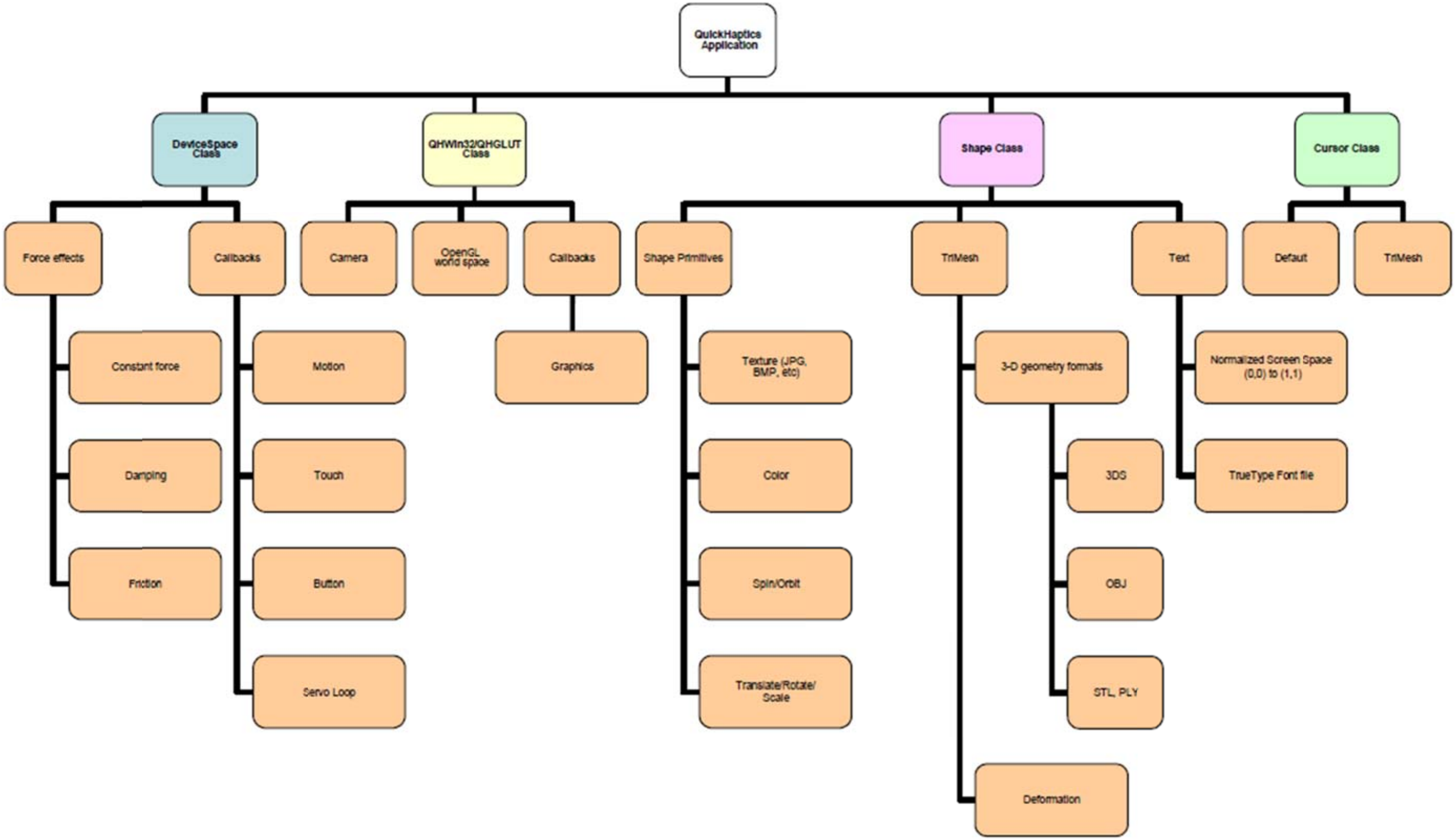


Figure A.1 : QuickHaptics micro API classes and properties[8].

APPENDIX B

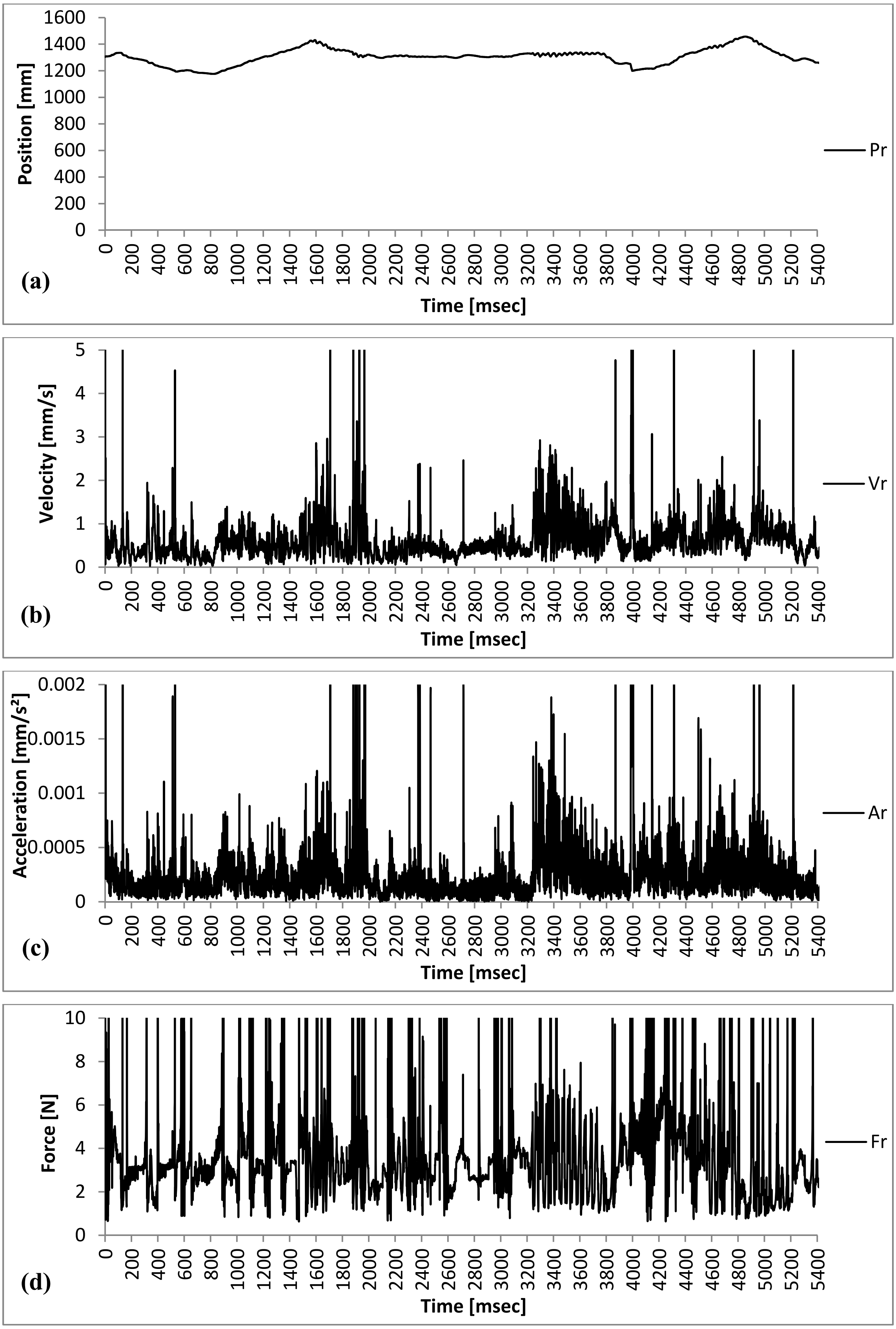


Figure B.1 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force.

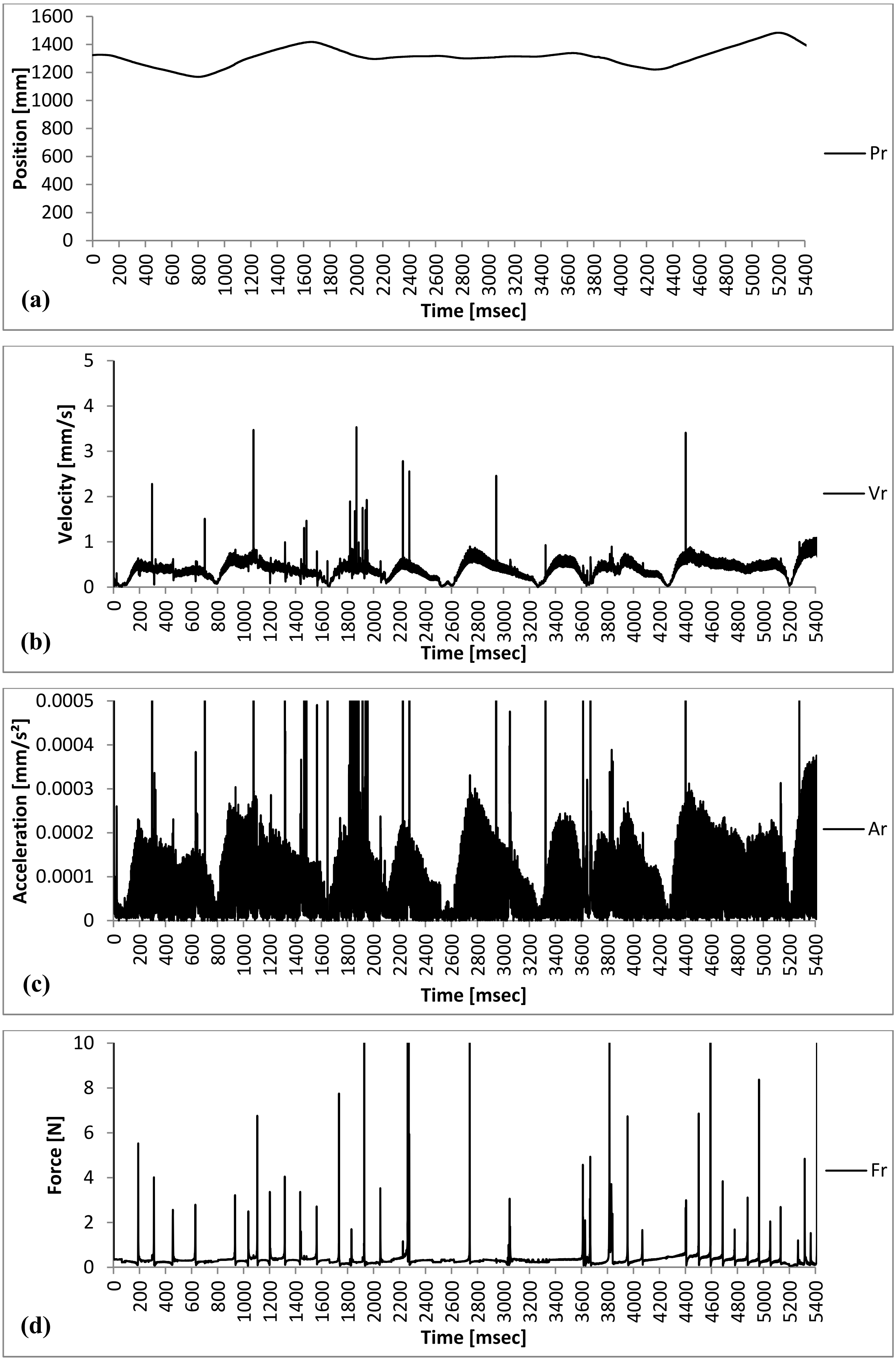


Figure B.2 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force.

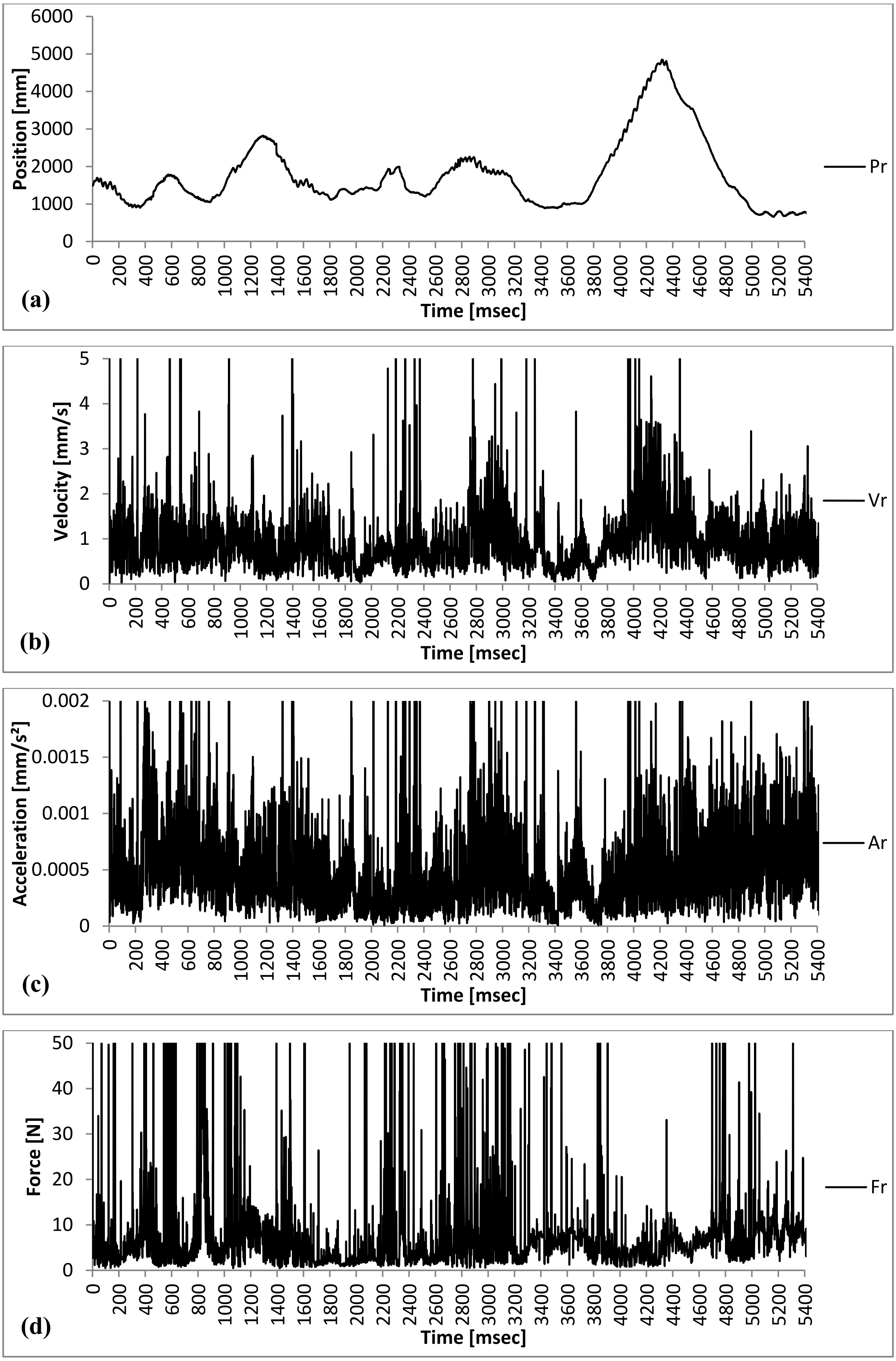


Figure B.3 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force.

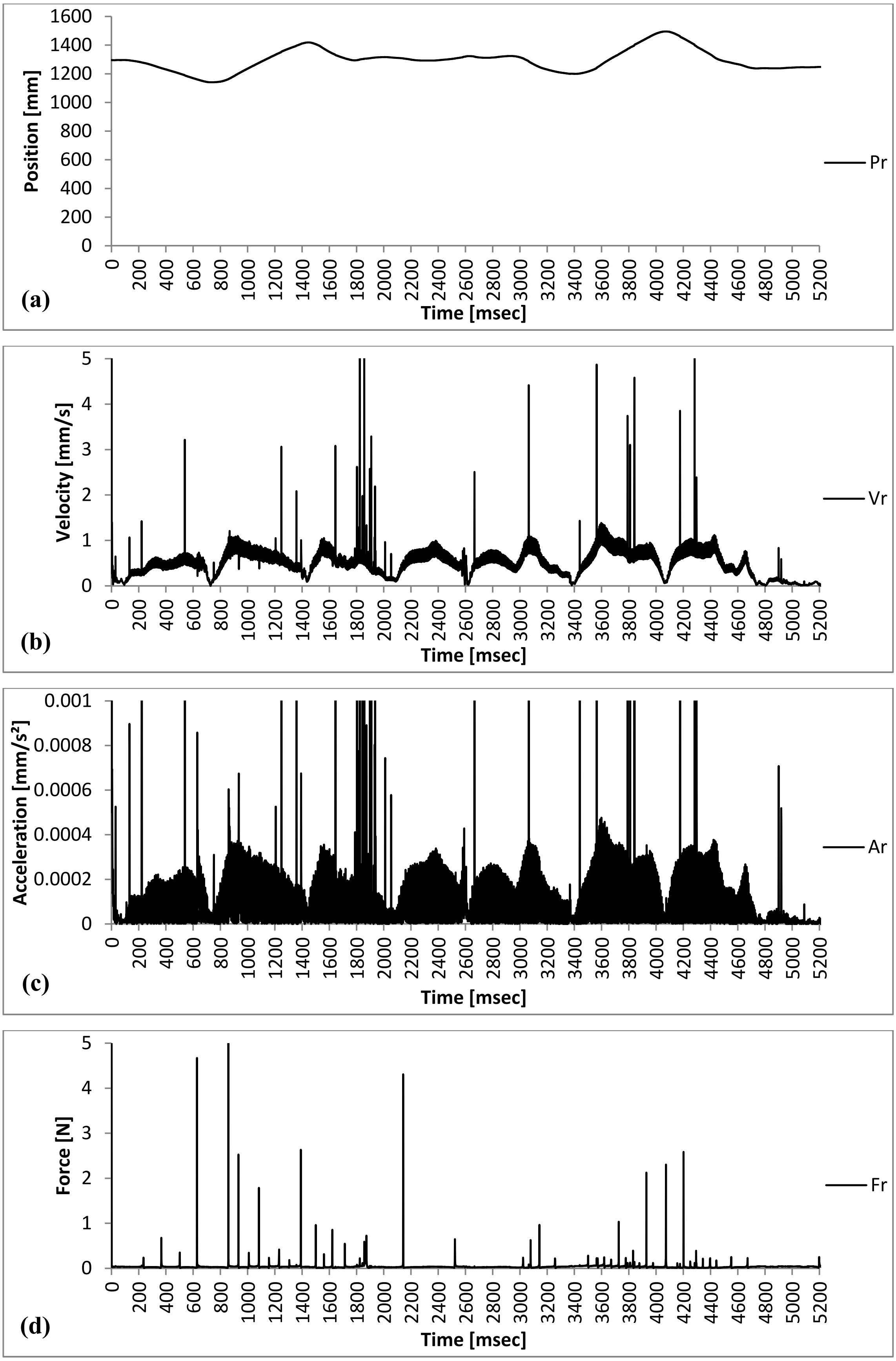


Figure B.4 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force.

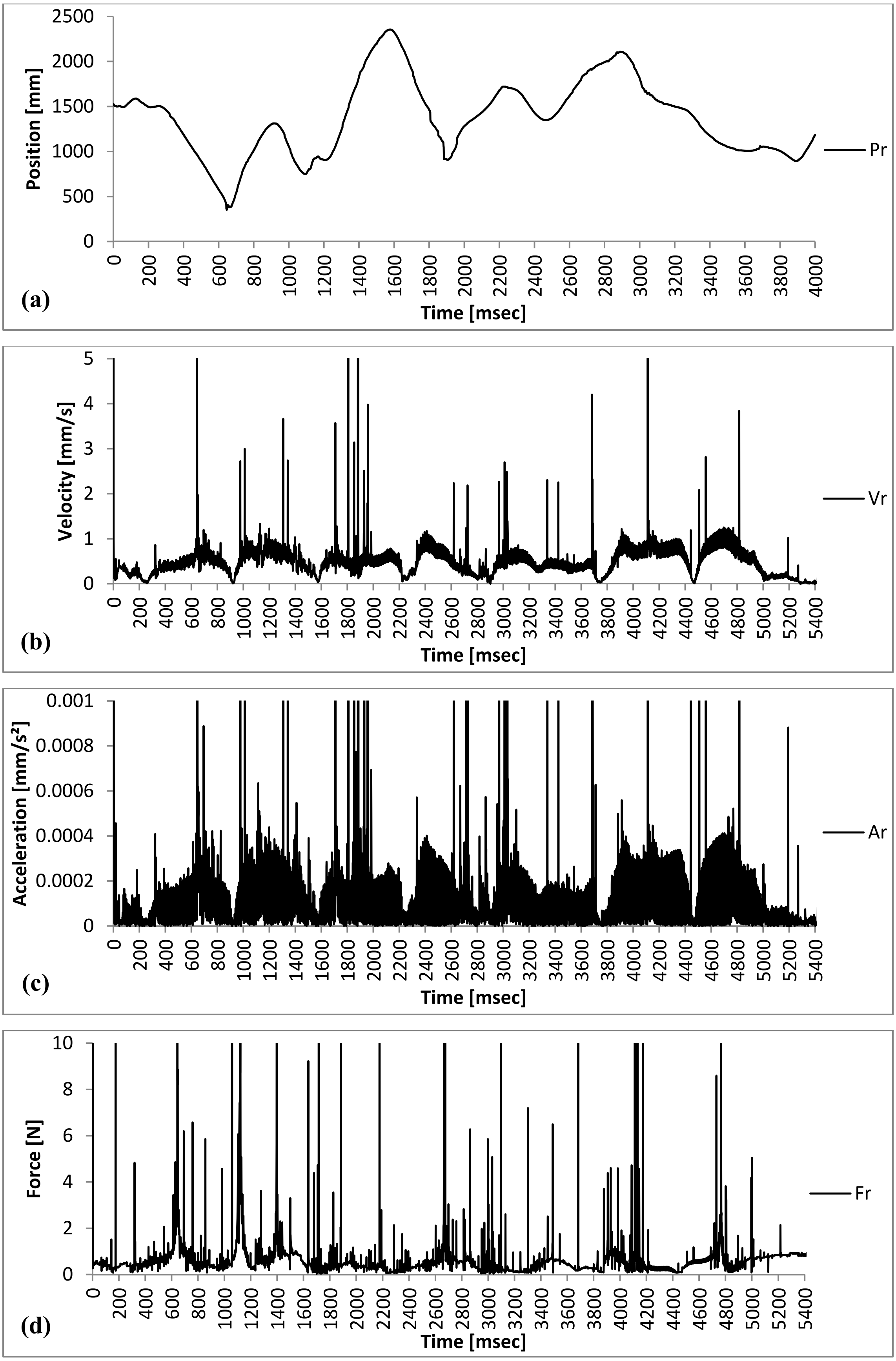


Figure B.5 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force.

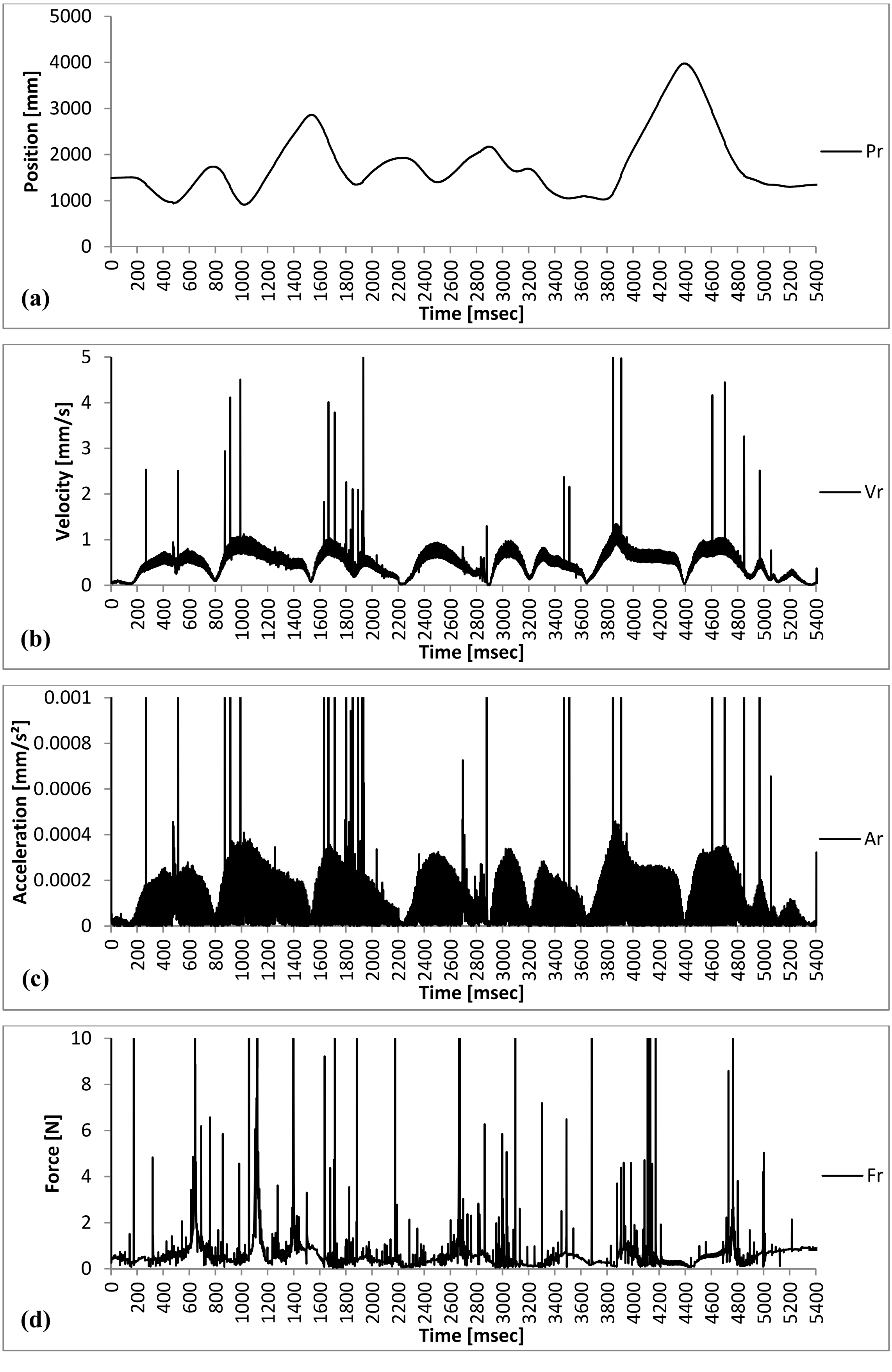


Figure B.6 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force.

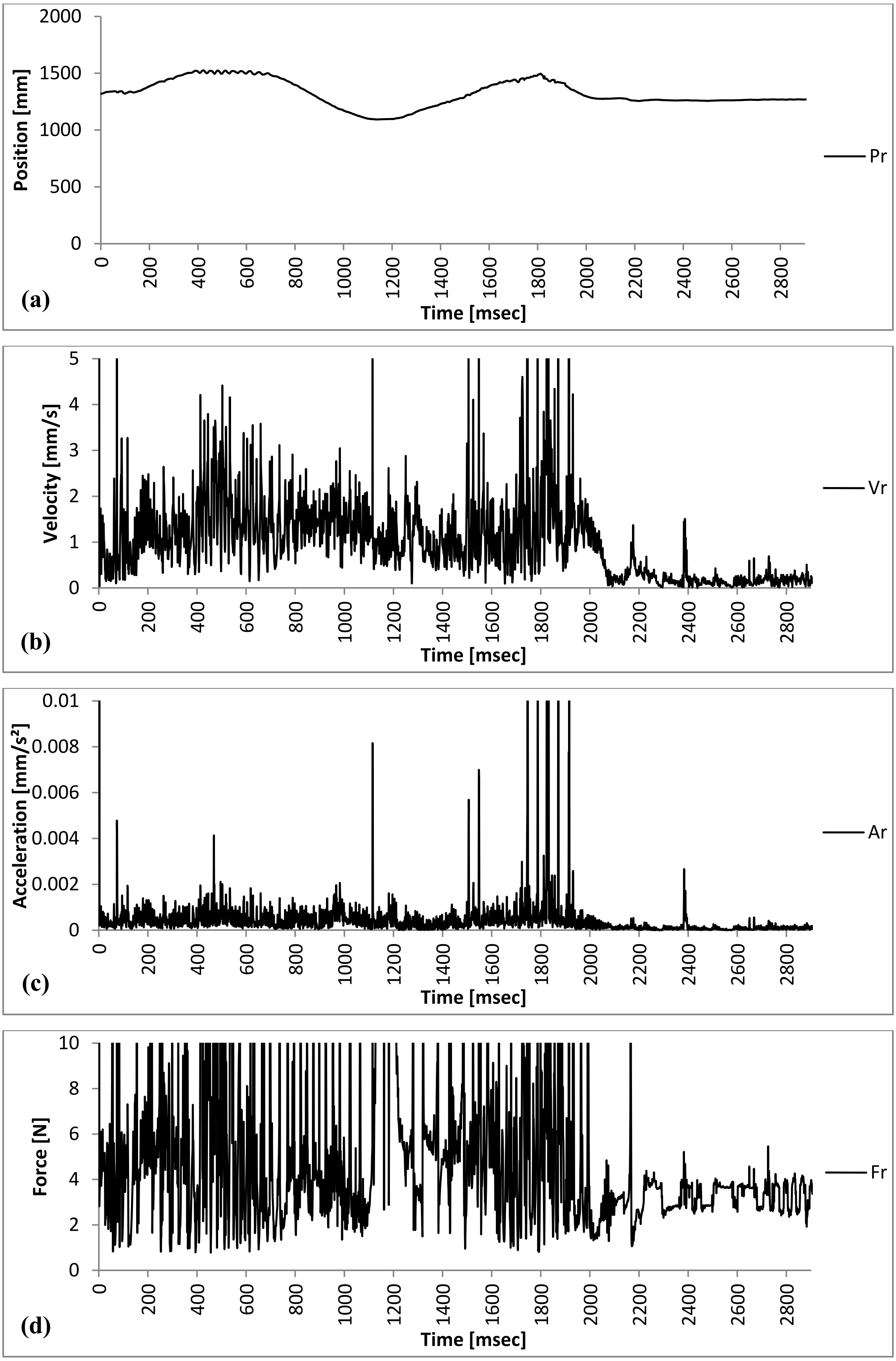


Figure B.7 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force.

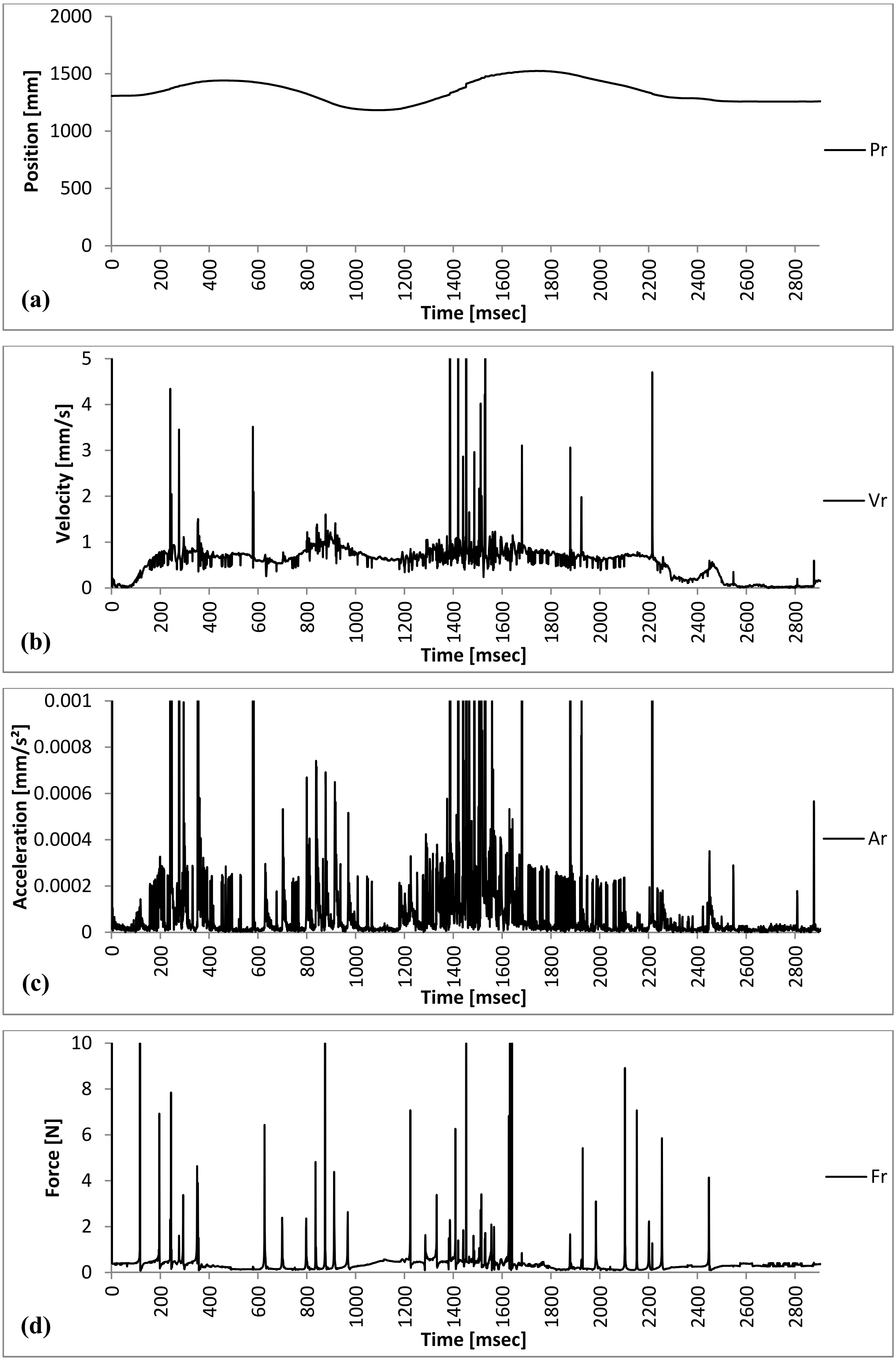


Figure B.8 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force.

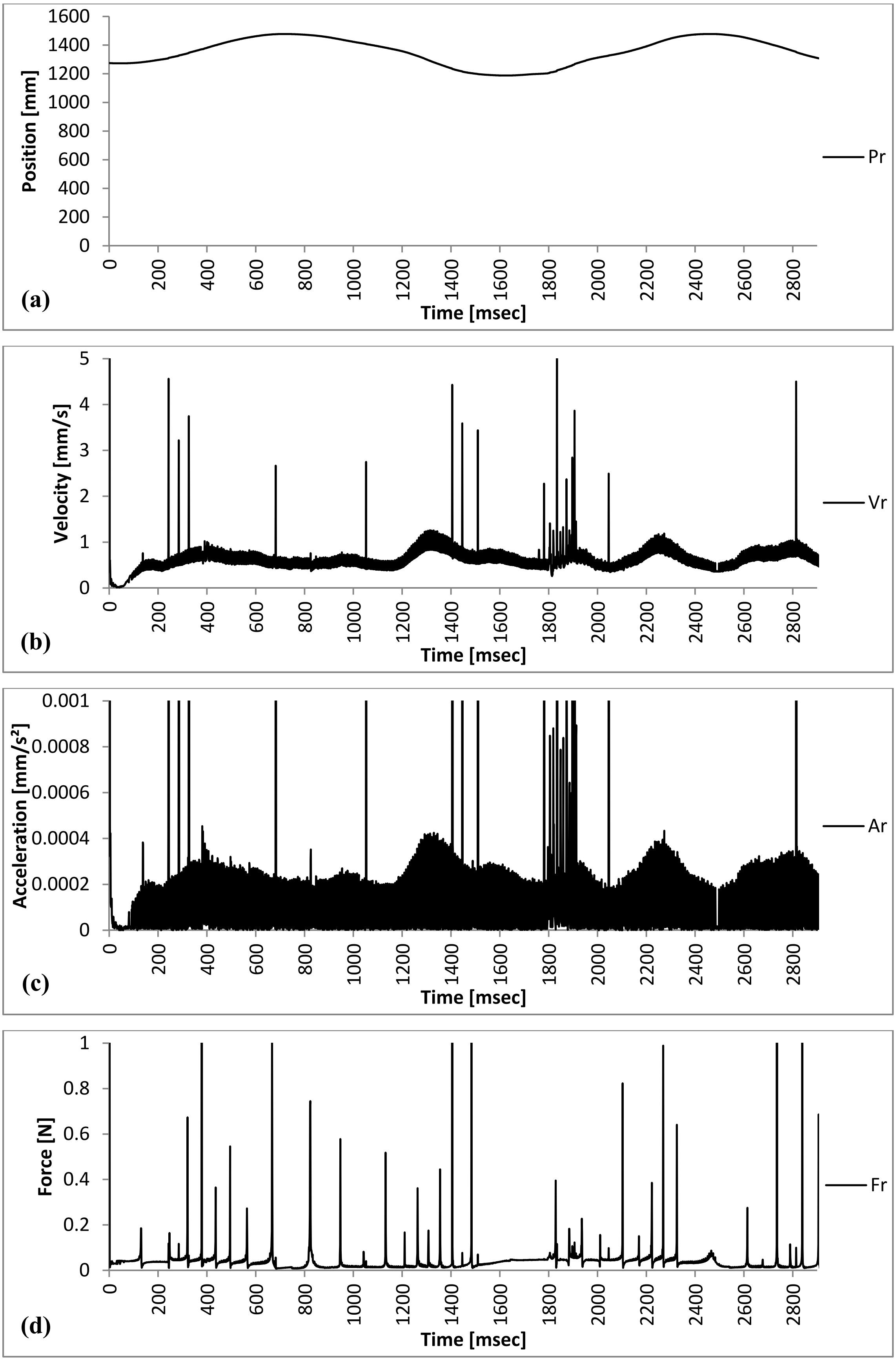


Figure B.9 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force.

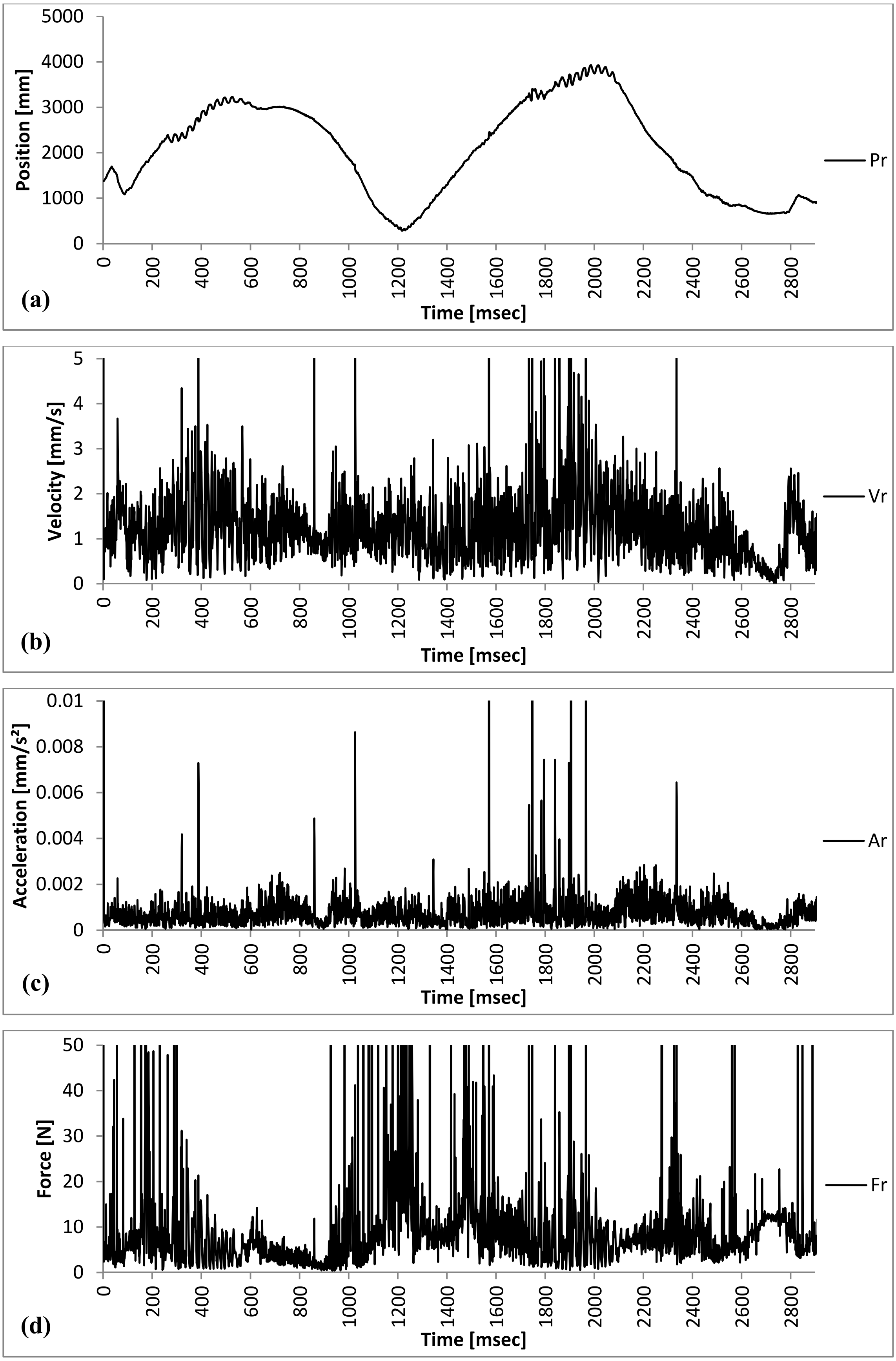


Figure B.10 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force.

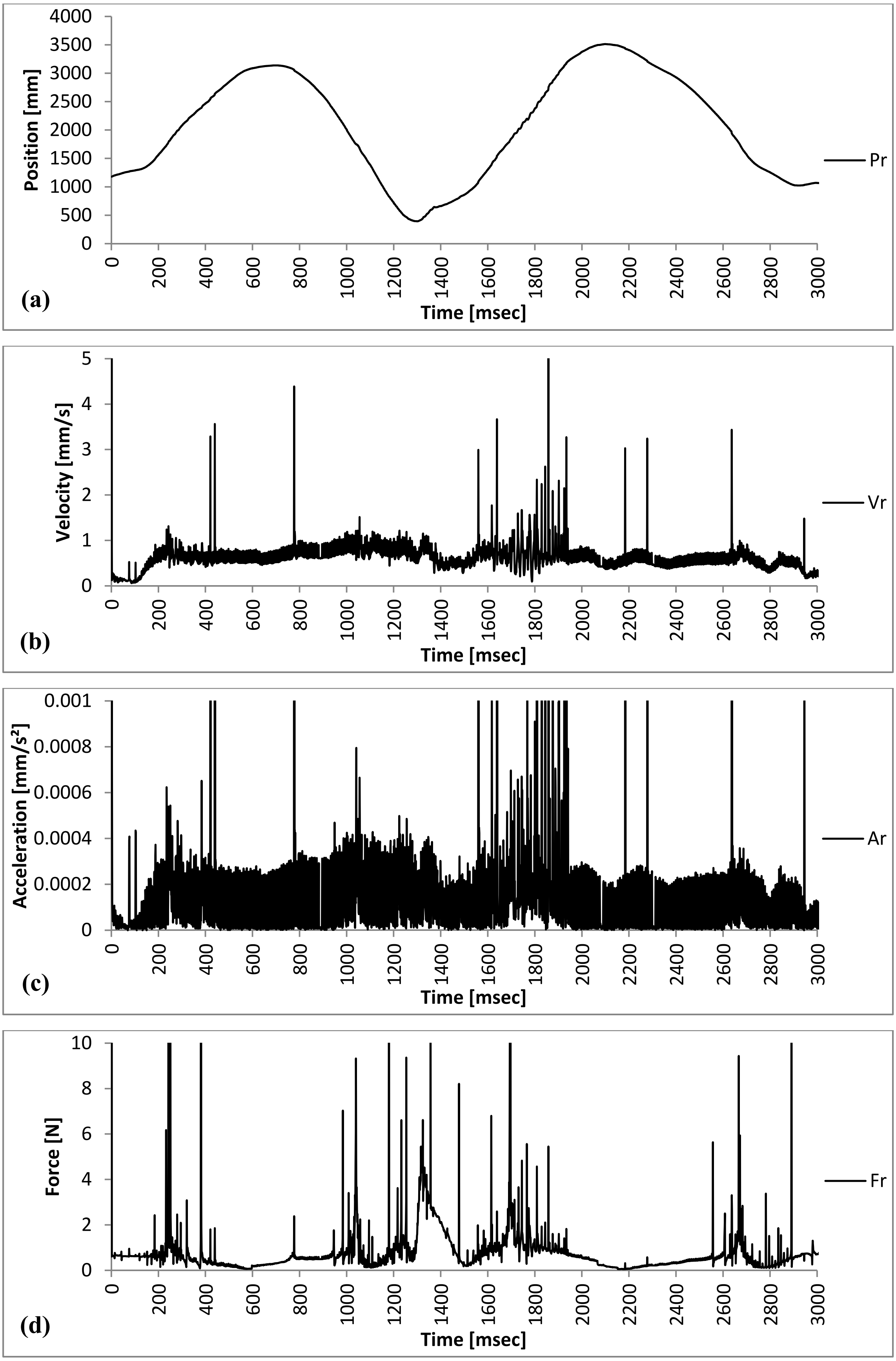


Figure B.11 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force.

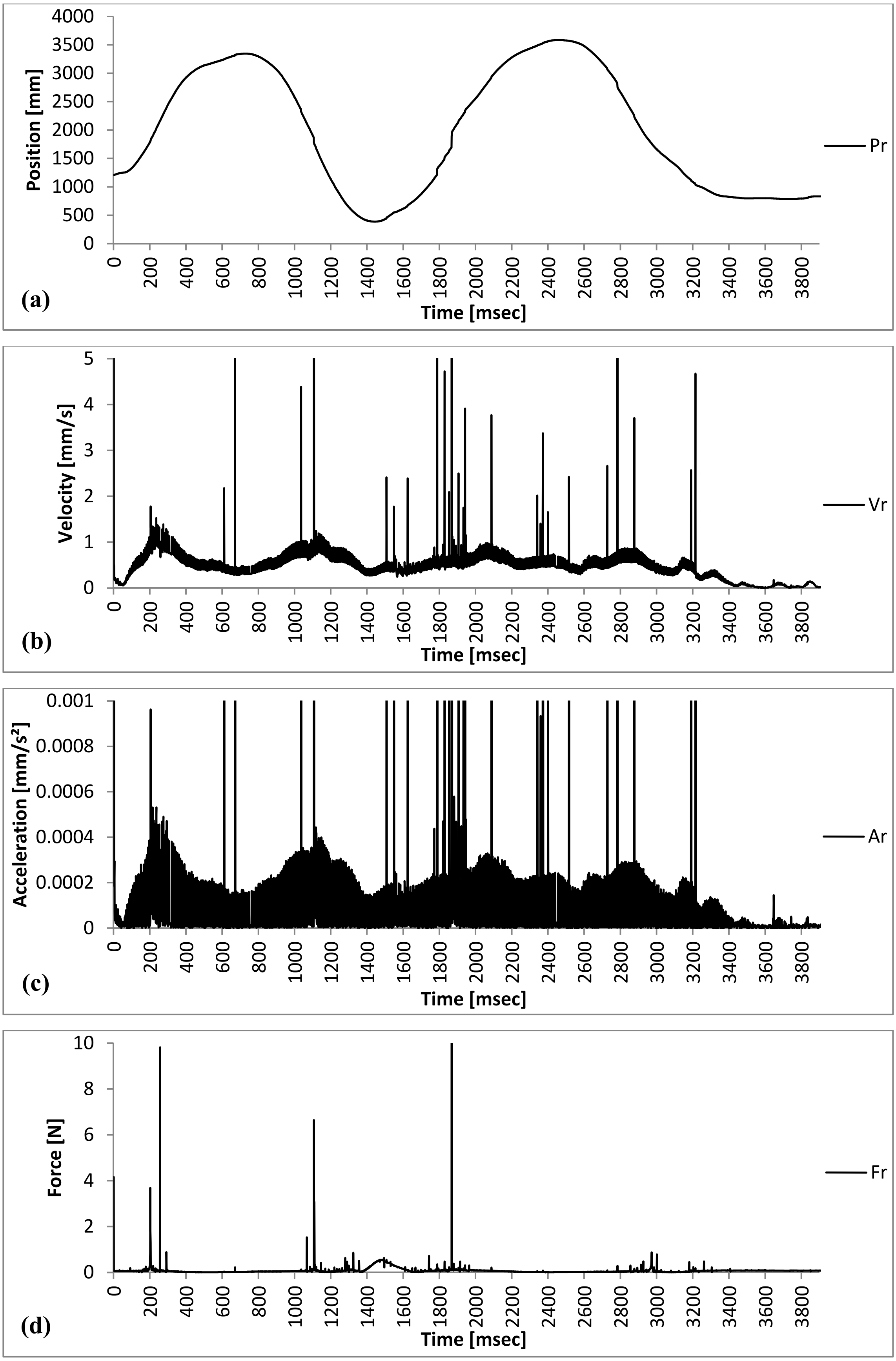


Figure B.12 : Resulting values (Magnitudes): (a)Position. (b)Velocity. (c)Acceleration. (d)Force.

CURRICULUM VITAE

Name Surname: Aykut GÖREN

Place and Date of Birth: EDİRNE-0.30.8.1988

Address: Innovia 1, Beylikdüzü/Istanbul

E-Mail: ayktgrn@gmail.com

B.Sc.: AKDENİZ UNIVERSITY

Professional Experience and Rewards:

ARÇELİK A.Ş.

Research and Development Engineer(2011-)

PUBLICATIONS

- "Numerical Investigation of Mixed Convection Heat Transfer from Porous Blocks in a Channel" - Journal of Engineers and Machinery(UNION OF CHAMBERS OF TURKISH ENGINEERS AND ARCHITECTS). Vol. 54. pp. 33-38., March 2013.