



---

***Elective in Robotics***

# **Geomagic Touch**

**Prof. Alessandro De Luca**

DIPARTIMENTO DI INGEGNERIA INFORMATICA  
AUTOMATICA E GESTIONALE ANTONIO RUBERTI



**SAPIENZA**  
UNIVERSITÀ DI ROMA

# Geomagic Touch haptic device



2 devices available at **DIAG Robotics Lab**

# Phantom Omni (it is the same!)

---



PHANTOM Omni  $\Rightarrow$  now **Geomagic Touch**

**SensAble Technologies**  $\Rightarrow$  now **3D Systems**



# Geomagic Touch data sheet



Force feedback workspace	~6.4 W x 4.8 H x 2.8 D in > 160 W x 120 H x 70 D mm
Footprint (Physical area device base occupies on desk)	6 5/8 W x 8 D in ~168 W x 203 D mm
Weight (device only)	3 lbs 15 oz (1.42 kg)
Range of motion	Hand movement pivoting at wrist
Nominal position resolution	> 450 dpi ~ 0.055 mm
Backdrive friction	< 1 oz (0.26 N)
Maximum exertable force at nominal (orthogonal arms) position	0.75 lbf (3.3 N)
Continuous exertable force (24 hrs)	0.2 lbf (0.88 N)
Stiffness	X axis > 7.3 lbs / in (1.26 N / mm) Y axis > 13.4 lbs / in (2.31 N / mm) Z axis > 5.9 lbs / in (1.02 N / mm)
Inertia (apparent mass at tip)	~0.101 lbm (45 g)
Force feedback (3 Degrees of Freedom)	x, y, z
Position sensing [Stylus gimbal] (6 Degrees of Freedom)	x, y, z (digital encoders) [Pitch, roll, yaw ( $\pm$ 5% linearity potentiometers)]
Interface	RJ45 compliant on-board Ethernet Port or USB Port
Supported platforms	Intel or AMD-based PCs
OpenHaptics®Toolkit compatibility	Yes



# Geomagic Touch in action

(actually, a Sensable Omni...)



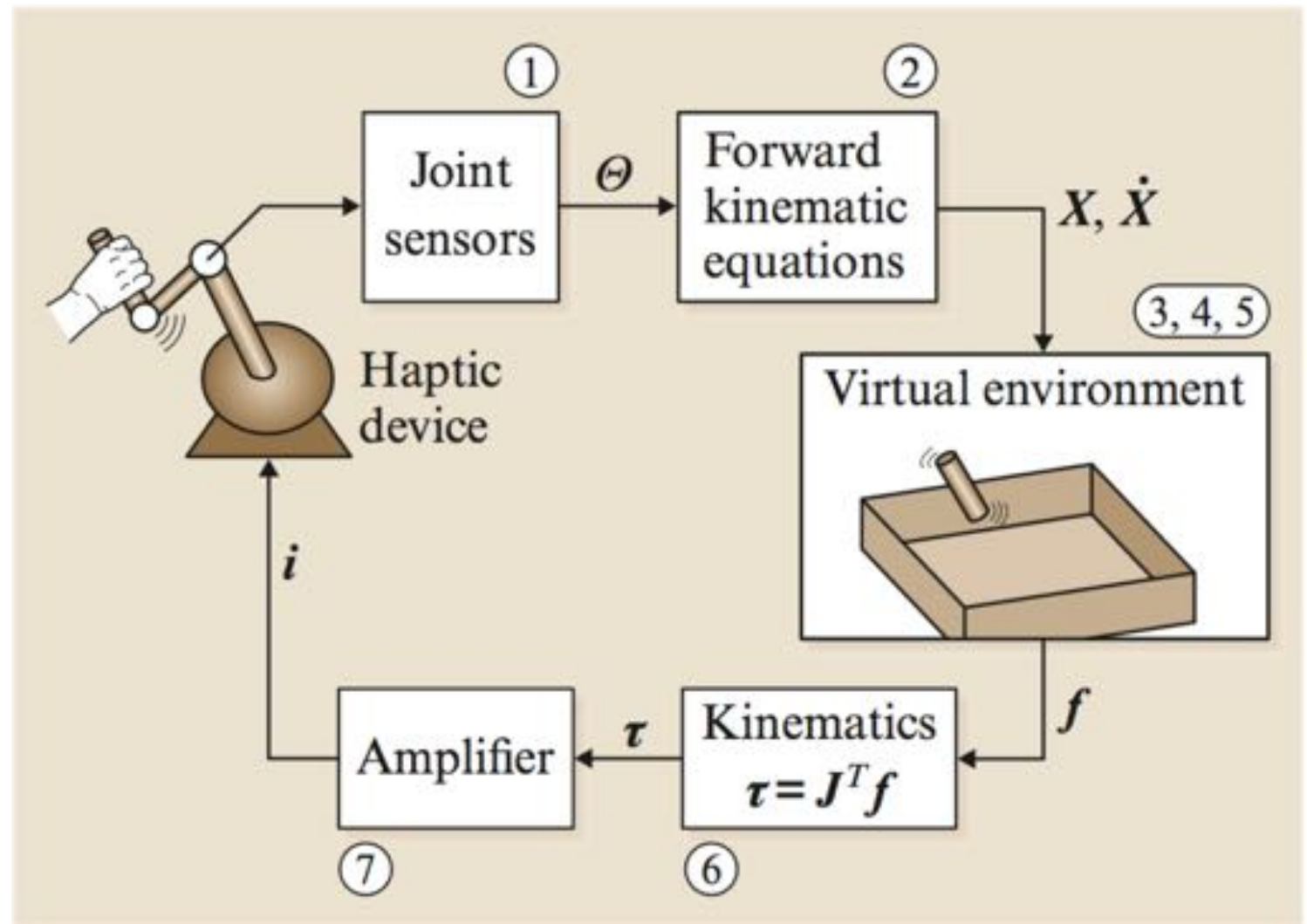
video

<https://youtu.be/REA97hRX0WQ>

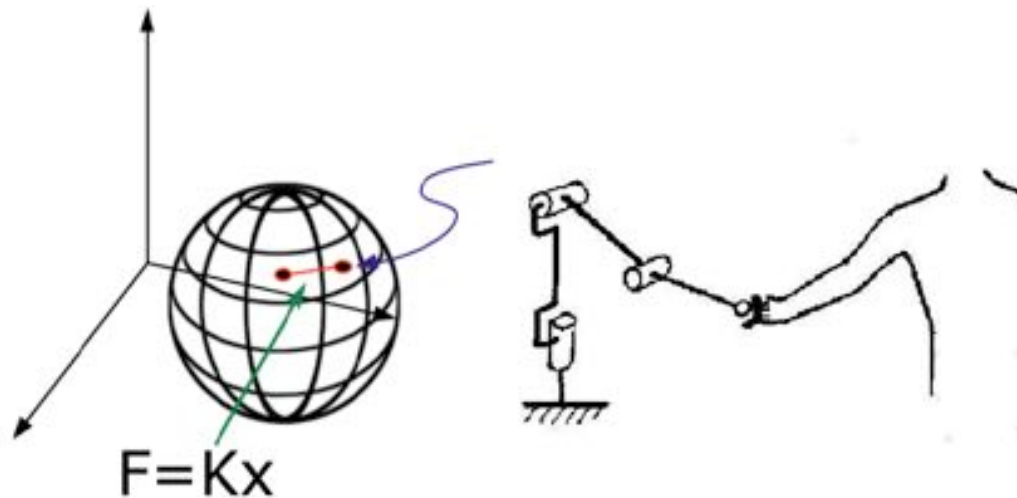


# Haptic rendering control loop

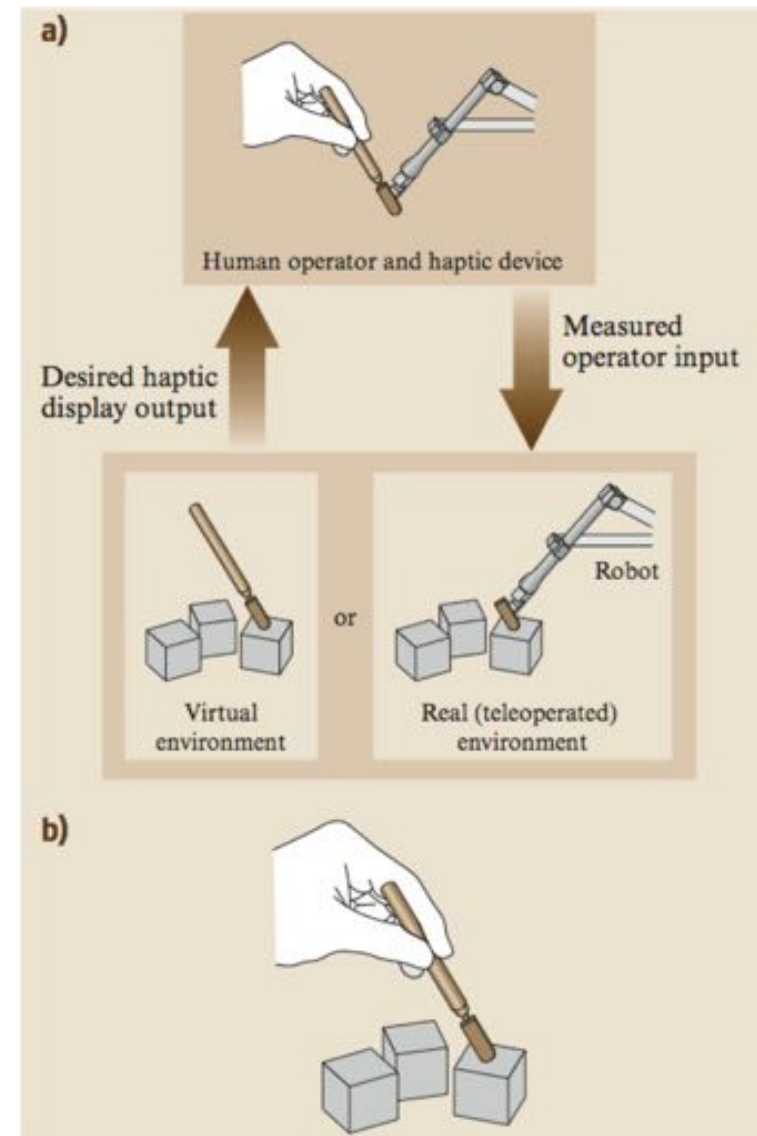
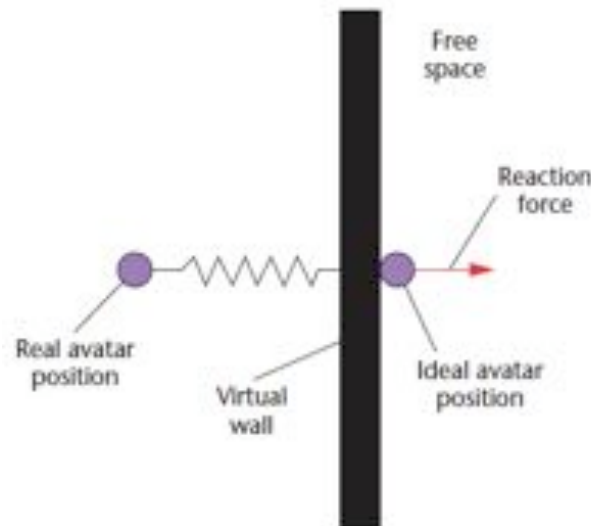
- ① joint displacement sensing (on device)
- ② (direct) kinematics
- ③ collision detection (environment geometry)
- ④ surface point determination
- ⑤ force calculation
- ⑥ kineto-statics
- ⑦ actuation (on device)



# Force feedback from Virtual or Real world



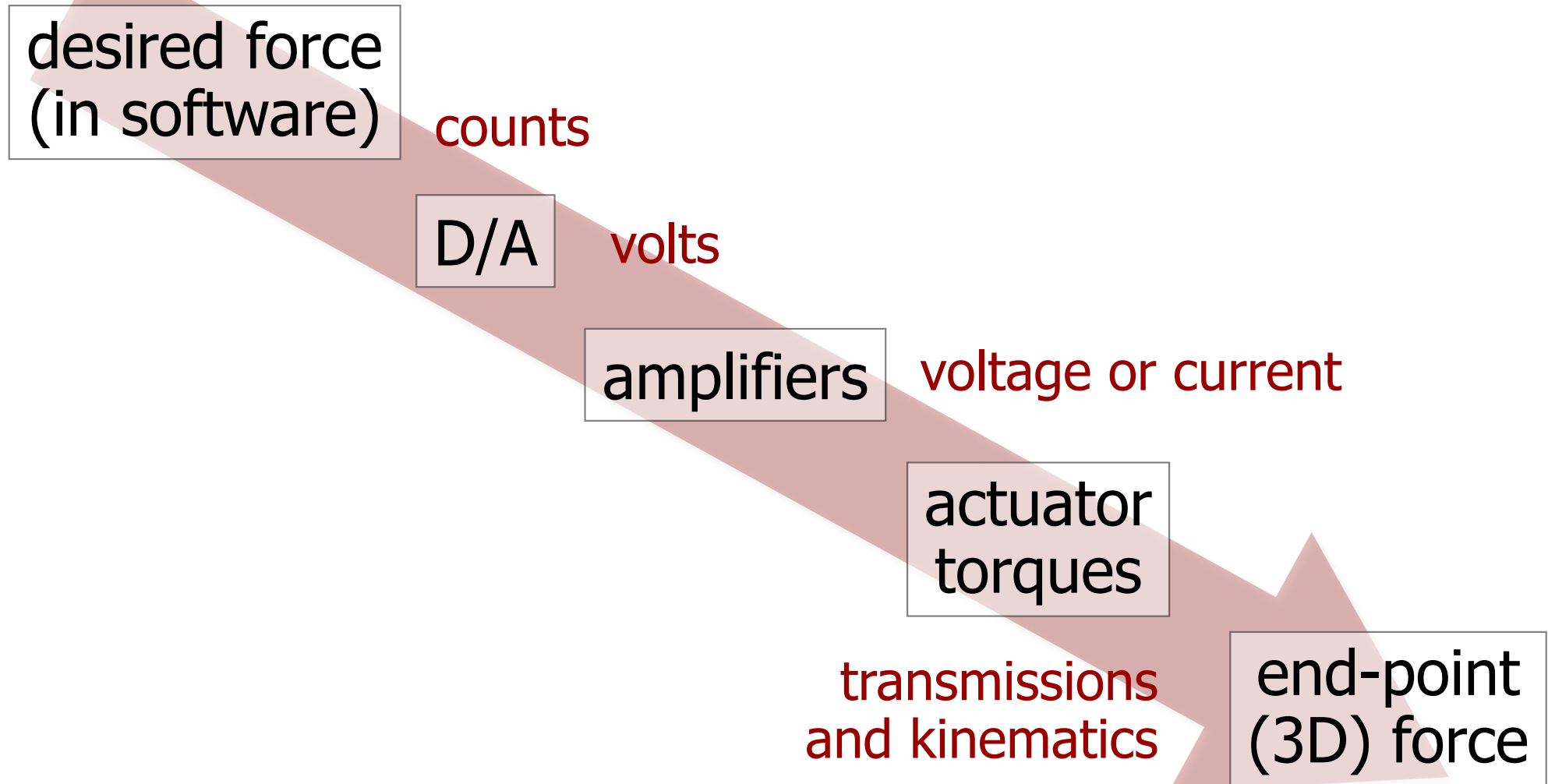
virtual  
environment  
compliance  
modeled with  
a **spring**/damper





# Force generation signals

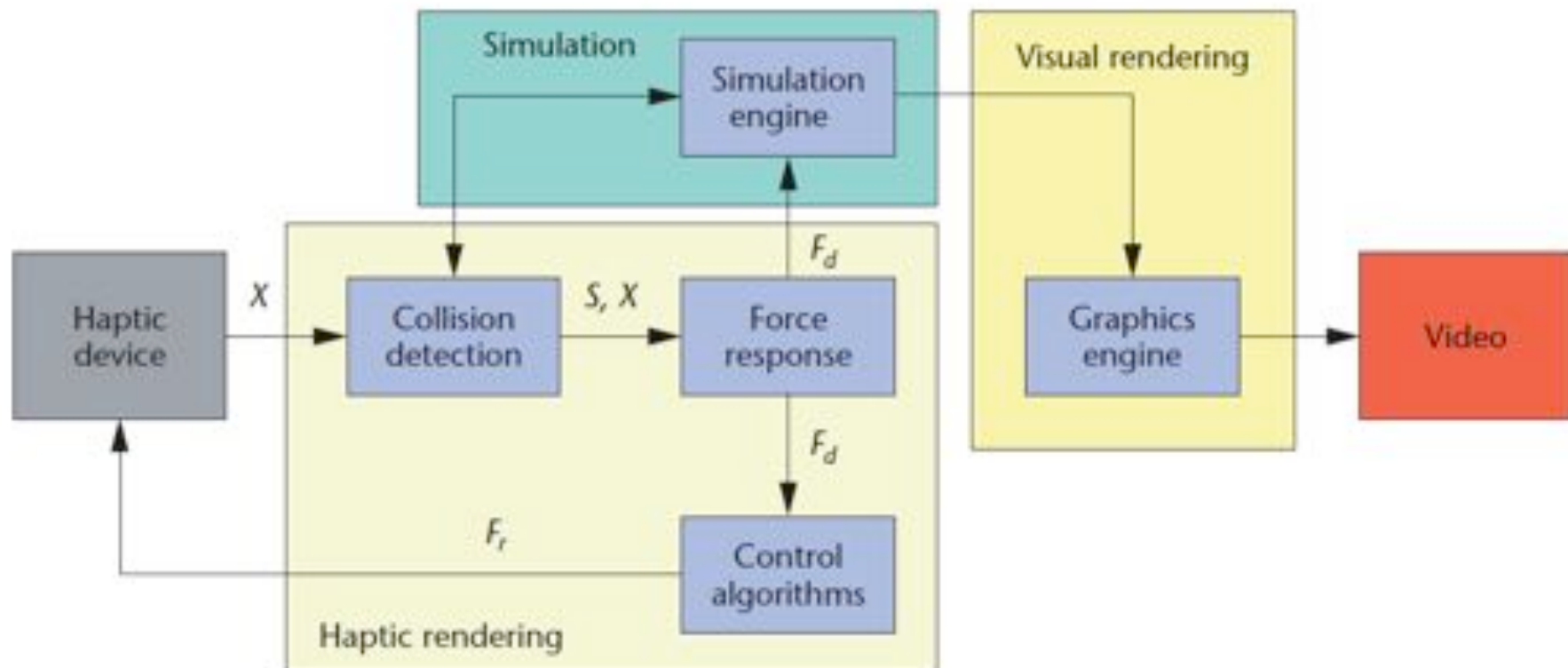
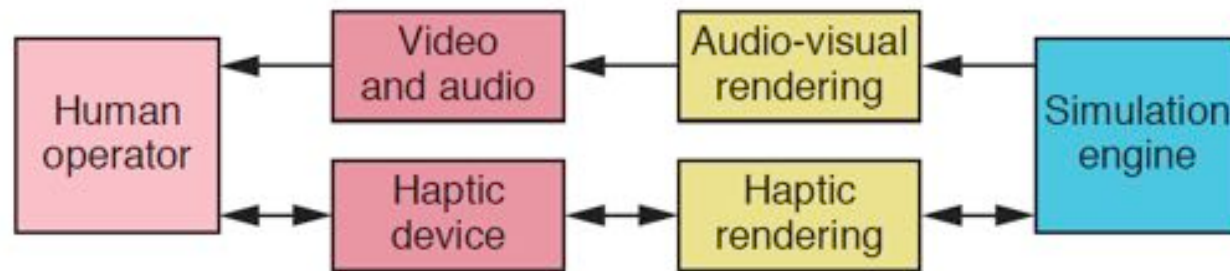
© Allison M. Okamura, 2015







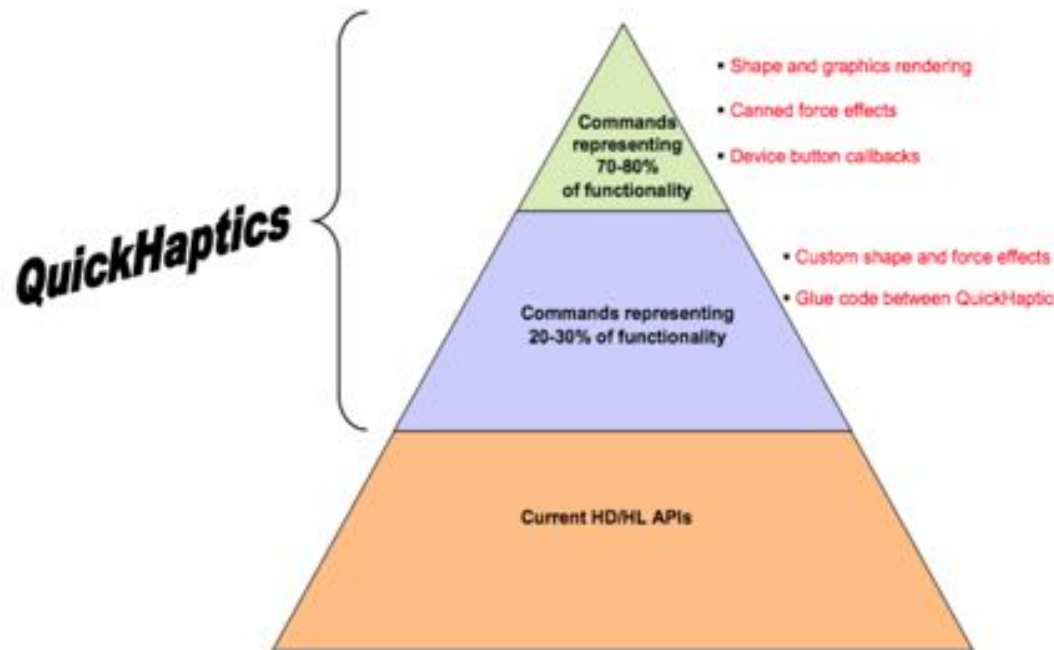
# Haptic/visual rendering architecture





# OpenHaptics 3.3.0

makes programming simpler by encapsulating the basic **common steps to all haptics/graphics applications** in C++ classes of the QuickHaptics micro API



## common steps

- parsing geometry files from popular animation packages
- creating graphics windows and initializing OpenGL environment
- initializing the haptic devices
- scene and camera design
- mapping force and stiffness parameters to objects in the scene
- setting up callback responses to interaction devices



# OpenHaptics Toolkit

## TABLE OF CONTENTS



1	PREFACE	5
	OVERVIEW	5
	WHAT IS HAPTICS?	5
	RESOURCES FOR LEARNING THE OPENHAPTICS TOOLKIT	6
	THE DEVELOPER SUPPORT CENTER	6
2	INTRODUCTION	7
	WHAT IS OPENHAPTICS 3.3.0?	7
	QUICKHAPTICS MICRO API CLASSES	8
	DEVICSPACE CLASS	9
	QHWIN32 OR QHGLUT CLASS	9
	SHAPE CLASS	12
	CURSOR CLASS	12
	DESIGN OF A TYPICAL QUICKHAPTICS MICRO API PROGRAM	13
	OVERVIEW OF HAPI VS. HDAPI	13
	WHEN TO USE ONE OVER THE OTHER	14
	WHAT PARTS CAN BE USED TOGETHER?	14
3	QUICKHAPTICS MICRO API PROGRAMMING	16
	INTRODUCTION	17
	EXAMPLE 1—CREATING A SHAPE (SIMPLESPHERE)	18
	EXAMPLE 2—ADDING TEXTURE AND MOTION (EARTHSPIN)	20
	EXAMPLE 3—DEFINING MULTIPLE PRIMITIVE OBJECTS (COMPLEXSCENE)	21
	EXAMPLE 4—TRIMESH MODELS AND EVENT CALLBACKS (PICKAPPLES)	24
	EXAMPLE 5—MULTIPLE WINDOWS AND VIEWS (MULTIPLEWINDOWS)	28
	EXAMPLE 6—USING DEFORMATION PROPERTIES (SPONGYCOW)	31
	EXAMPLE 7—DEFINING CUSTOM FORCE LAWS (SKULLCOULOMBFORCE)	34
	EXAMPLE 8—HAPTIC RENDERING OF LARGE MODELS (SHAPEDEPTHFEEDBACK)	46
	EXAMPLE 9—A DENTAL SIMULATOR (TEETHCAVITYPICK)	52
	SUMMARY OF DEFAULT VALUES	63
4	CREATING HAPTIC ENVIRONMENTS	66
	INTRODUCTION TO FORCES	67
	FORCE RENDERING	67
	CONTACT AND CONSTRAINTS	69
	COMBINING HAPTICS WITH GRAPHICS	69
	COMBINING HAPTICS WITH DYNAMICS	70
	HAPTIC UI CONVENTIONS	70



# OpenHaptics Toolkit



## TABLE OF CONTENTS



6	HDAPI OVERVIEW.....	79
	GETTING STARTED.....	80
	THE DEVICE.....	80
	THE SCHEDULER.....	80
	DEVELOPING HDAPI APPLICATIONS.....	81
	MICROSOFT WIN32 VERSUS CONSOLE APPLICATIONS.....	84
	DESIGN OF TYPICAL HDAPI PROGRAM.....	85
7	HDAPI PROGRAMMING.....	86
	HAPTIC DEVICE OPERATIONS.....	87
	HAPTIC FRAMES.....	88
	SCHEDULER OPERATIONS.....	88
	STATE.....	90
	CALIBRATION INTERFACE.....	92
	FORCE/TORQUE CONTROL.....	94
	ERROR REPORTING AND HANDLING.....	96
	CLEANUP.....	97
8	HLAPI OVERVIEW.....	98
	GENERATING FORCES.....	99
	LEVERAGING OPENGL.....	99
	PROXY RENDERING.....	99
	DESIGN OF TYPICAL HLAPI PROGRAM.....	100
	THREADING.....	101
9	HLAPI PROGRAMMING.....	102
	DEVICE SETUP.....	103
	RENDERING CONTEXTS.....	103
	HAPTIC FRAMES.....	103
	RENDERING SHAPES.....	104
	MAPPING HAPTIC DEVICE TO GRAPHICS SCENE.....	110
	DRAWING A 3D CURSOR.....	113
	MATERIAL PROPERTIES.....	114
	SURFACE CONSTRAINTS.....	115
	PUSHING AND POPPING ATTRIBUTES.....	116
	EFFECTS.....	117
	EVENTS.....	118
	CALIBRATION.....	120
	DYNAMIC OBJECTS.....	121
	DIRECT PROXY RENDERING.....	122
	SCP DEPTH OF PENETRATION.....	123
	MULTIPLE DEVICES.....	123
	EXTENDING HLAPI.....	124

## Haptic Device API

## Haptic Library API





# OpenHaptics Toolkit

## TABLE OF CONTENTS

11 UTILITIES	131
VECTOR/MATRIX MATH	132
WORKSPACE TO CAMERA MAPPING	134
SNAP CONSTRAINTS	135
C++ HAPTIC DEVICE WRAPPER	136
HDUEXCEPTION	137
HDURECORD	138
HAPTIC MOUSE	138
12 TROUBLESHOOTING	140
DEVICE INITIALIZATION	141
THREAD SAFETY	141
RACE CONDITIONS	143
CALIBRATION	144
BUZZING	144
FORCE KICKING	148
NO FORCES	149
DEVICE STUTTERING	149
ERROR HANDLING	149

### Matrix Utilities

The <HDU/hduMatrix.h> header exposes a simple API for common matrix operations.

#### Default constructor

hduMatrix mat1; // the identity matrix by default

```
HDdouble a[4][4] = {
    {a1,a2,a3,a4},
    {a5,a6,a7,a8},
    {a9,a10,a11,a12},
    {a13,a14,a15,a16}
};
```

mat1.set(a);

#### Constructor from sixteen values

```
hduMatrix mat(a1,a2,a3,a4,a5,a6,a7,a8,
              a9,a10,a11,a12,a13,a14,a15,a16);
```

#### Constructor from an array

```
HDdouble a[4][4] = {
    {a1,a2,a3,a4},
    {a5,a6,a7,a8},
    {a9,a10,a11,a12},
    {a13,a14,a15,a16}
};
```

hduMatrix mat2(a);

#### Assignment

hduMatrix mat3 = mat2;

#### Get values

double vals[4][4];

mat3.get(rotVals);

#### Usual operations

mat3 = mat2 + 4.0 \* mat1;

#### Invert

mat3 = mat2.getInverse();

#### Transpose:

mat3 = mat2.transpose();

#### Create a rotation

hduMatrix rot;

rot = createRotation(vec1, 30.0\*DEGTORAD);

HDdouble rotVals[4][4];

rot.get(rotVals);

glMultMatrixd((double\*)rotVals);

### Vector Utilities

The <HDU/hduVector.h> header exposes a simple API for common vector operations in three dimensional space. the functions follows:

#### Default constructor

```
hduVector3Dd vec1;
vec1.set(1.0, 1.0, 1.0);
```

#### Constructor from three values

```
hduVector3Dd vec2(2.0, 3.0, 4.0);
```

#### Constructor from an array

```
HDdouble x[3] = {1.0, 2.0, 3.0};
```

```
hduVector3Dd xvec = hduVector3Dd(x);
```

#### Assignment

```
hduVector3Dd vec3 = hduVector3Dd(2.0, 3.0, 4.0);
```

#### Usual operations:

```
vec3 = vec2 + 4.0* vec1;
```

#### Magnitude:

```
HDdouble magn = vec3. magnitude();
```

#### Dot product:

```
HDdouble dprod = dotProduct(vec1, vec2);
```

Cross product:

```
hduVector3Dd vec4 = crossProduct(vec1, vec2);
```

#### Normalize:

```
vec4.normalize();
```



# QuickHaptics micro API

## classes and properties



implemented in C++, with  
4 primary functional classes



location of the  
Haptic Interface Point (HIP)  
on the Touch device

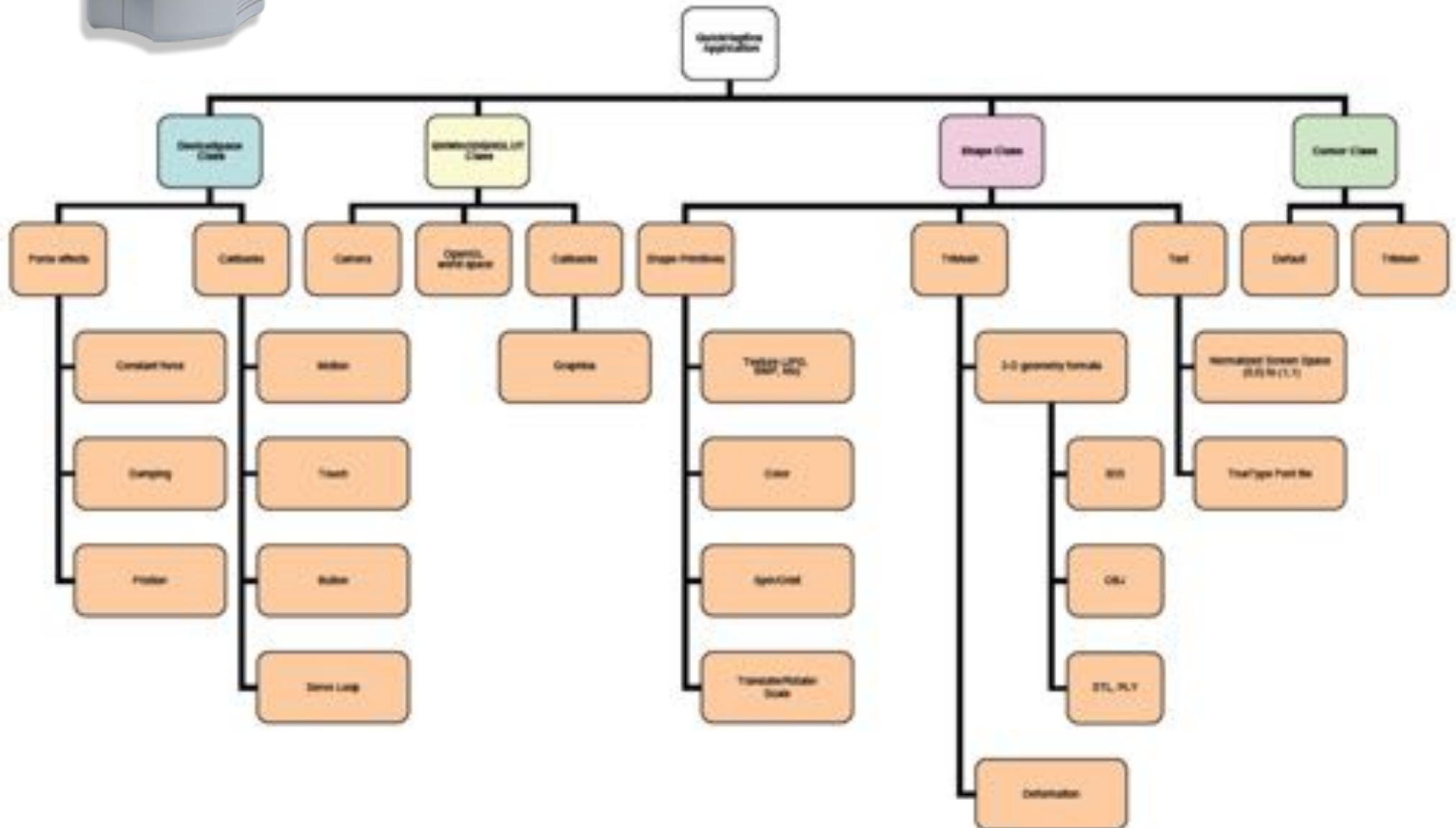
- DeviceSpace
  - workspace of motion for haptic device
- QHRenderer (OpenGL)
  - on-screen window that renders shapes from a camera viewpoint and lets the user feel those shapes via the device
- Shape
  - base class for one or more geometric objects that can be rendered both graphically and haptically
- Cursor
  - graphical representation of the end point of the second link of the device (HIP)



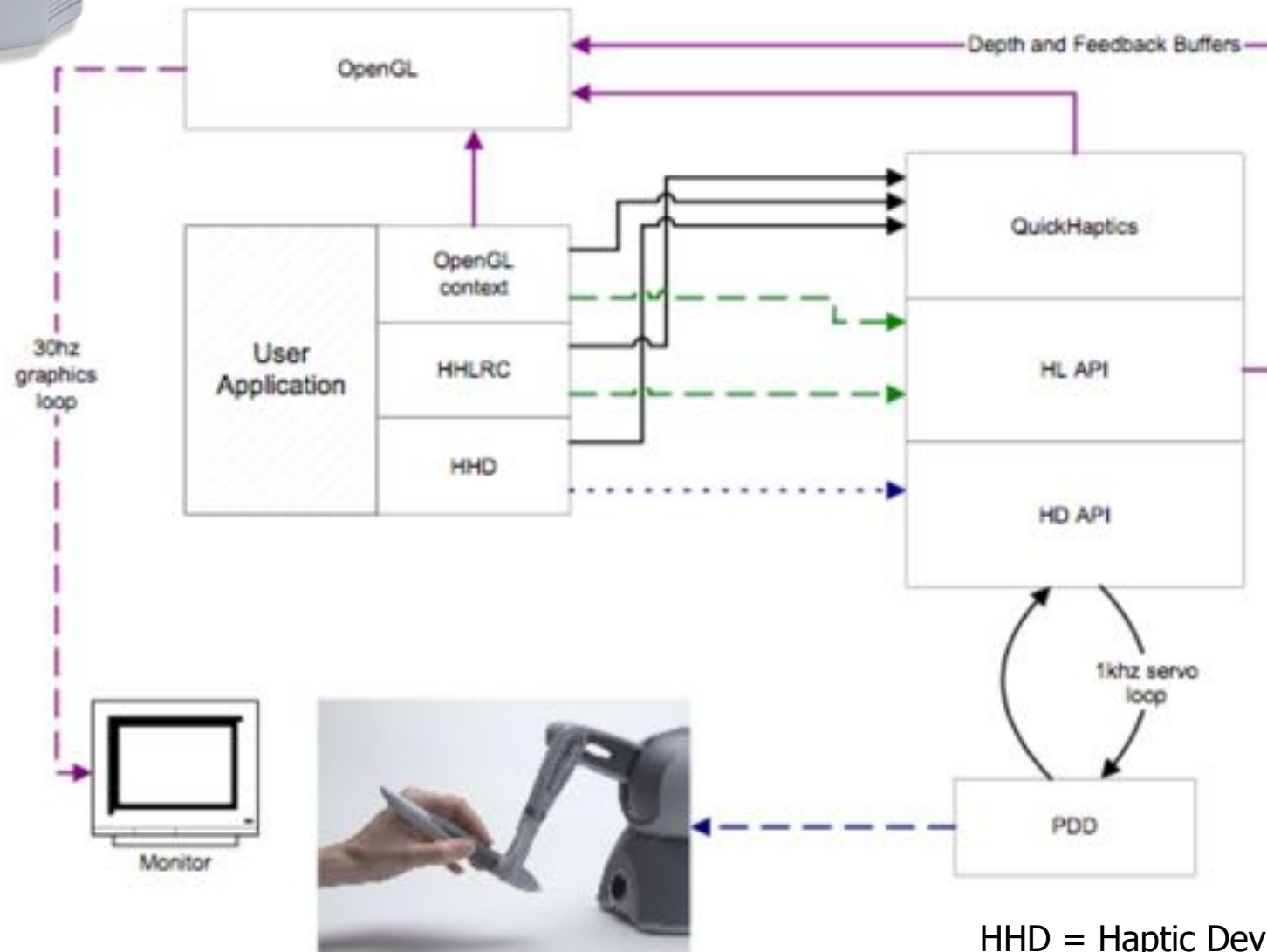


# QuickHaptics micro API

## classes and properties



# OpenHaptics Overview

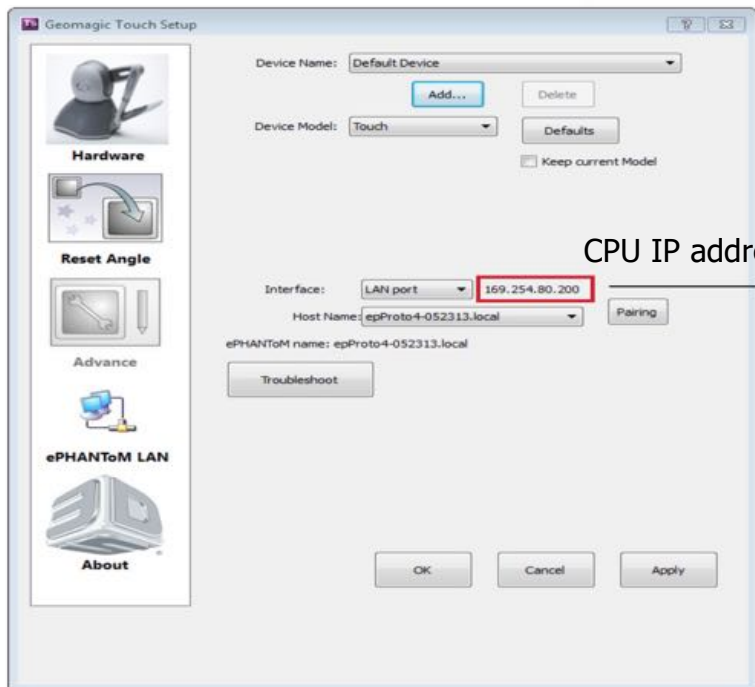


HHD = Haptic Device Handle  
HHLRC = Haptic Rendering Context  
PDD = ... Control

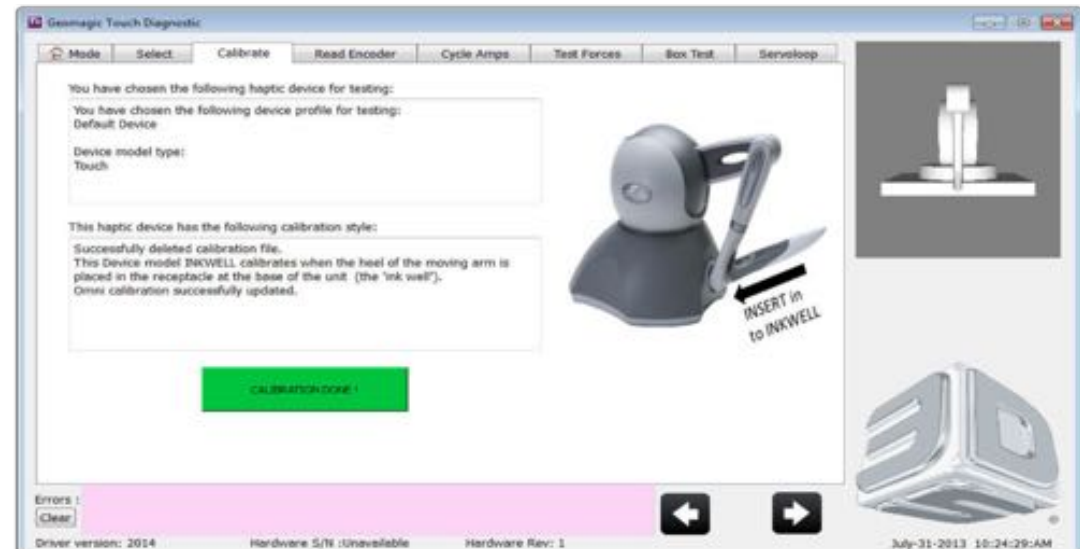
# Setting up the system



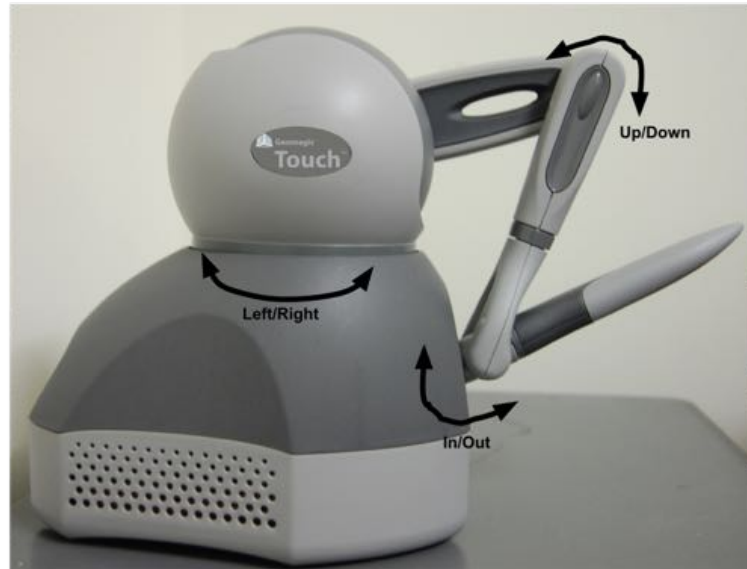
pairing the device



calibrating the device (stylus in the ink well)



# Range of device motion



macro movements



micro movements  
(of the stylus)  
with fixed HIP

only the **first three**  
joints are actuated!



## pen buttons

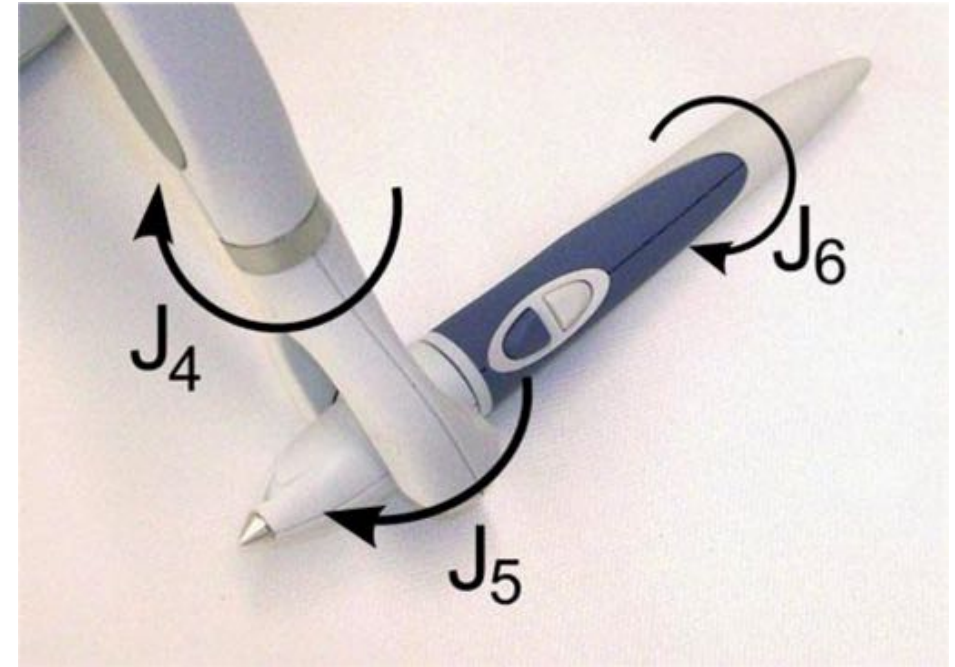
- stylus switch ON if pressing **dark** gray button
- presence switch ON if pressing light gray button



# Degrees of freedom



first three joints  
(positioning of the HIP):  
**actuated**



last three joints as spherical wrist  
(orientation of the stylus):  
**passive**



# What could be studied?

- DH parameters and forward kinematics
- inverse kinematics
- Jacobian matrix and singularities
- joint level PD and PID control
- trajectory planning (joint space vs. task space)
- various haptic (force) rendering laws
  - force fields, “god point”, hard and soft contacts
- *on-going development of a software environment for the simulation of the haptic device...*



# Medical application

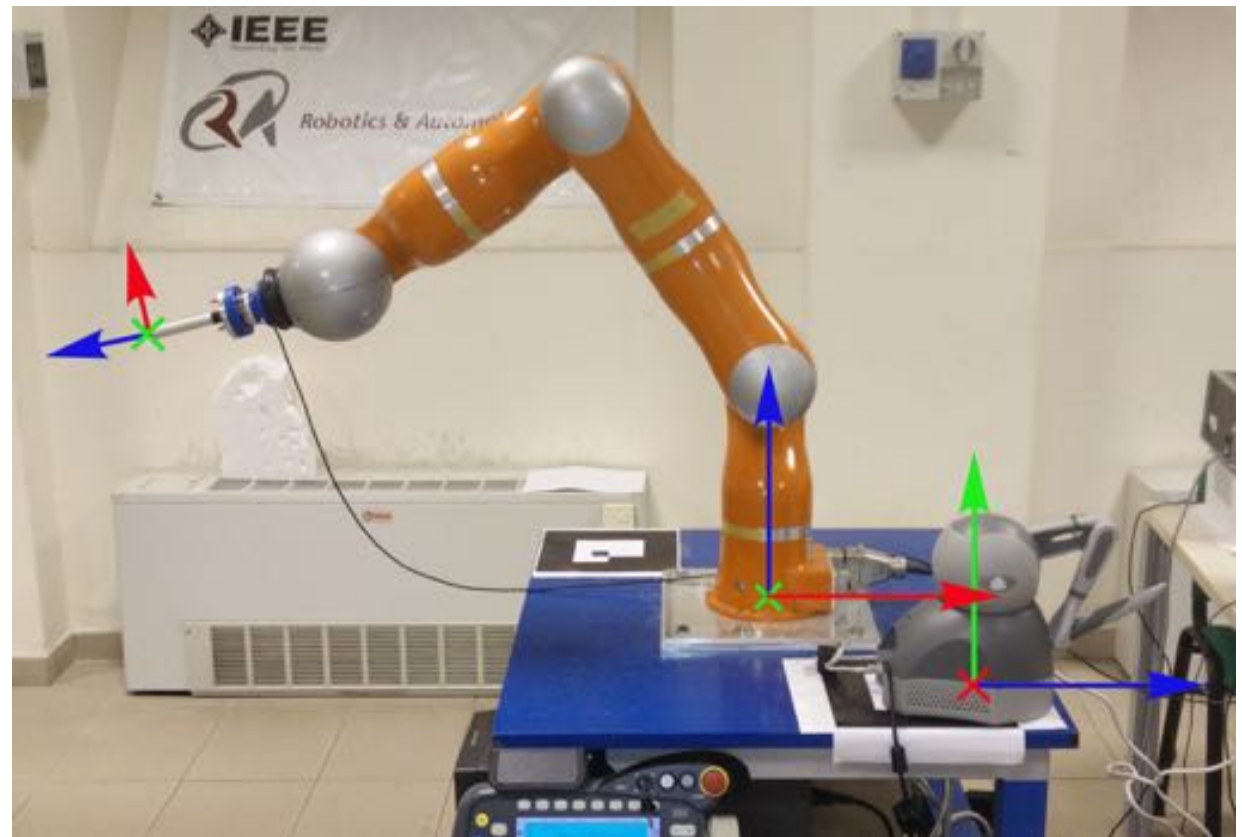
---



# Laboratory test bed with KUKA LWR4



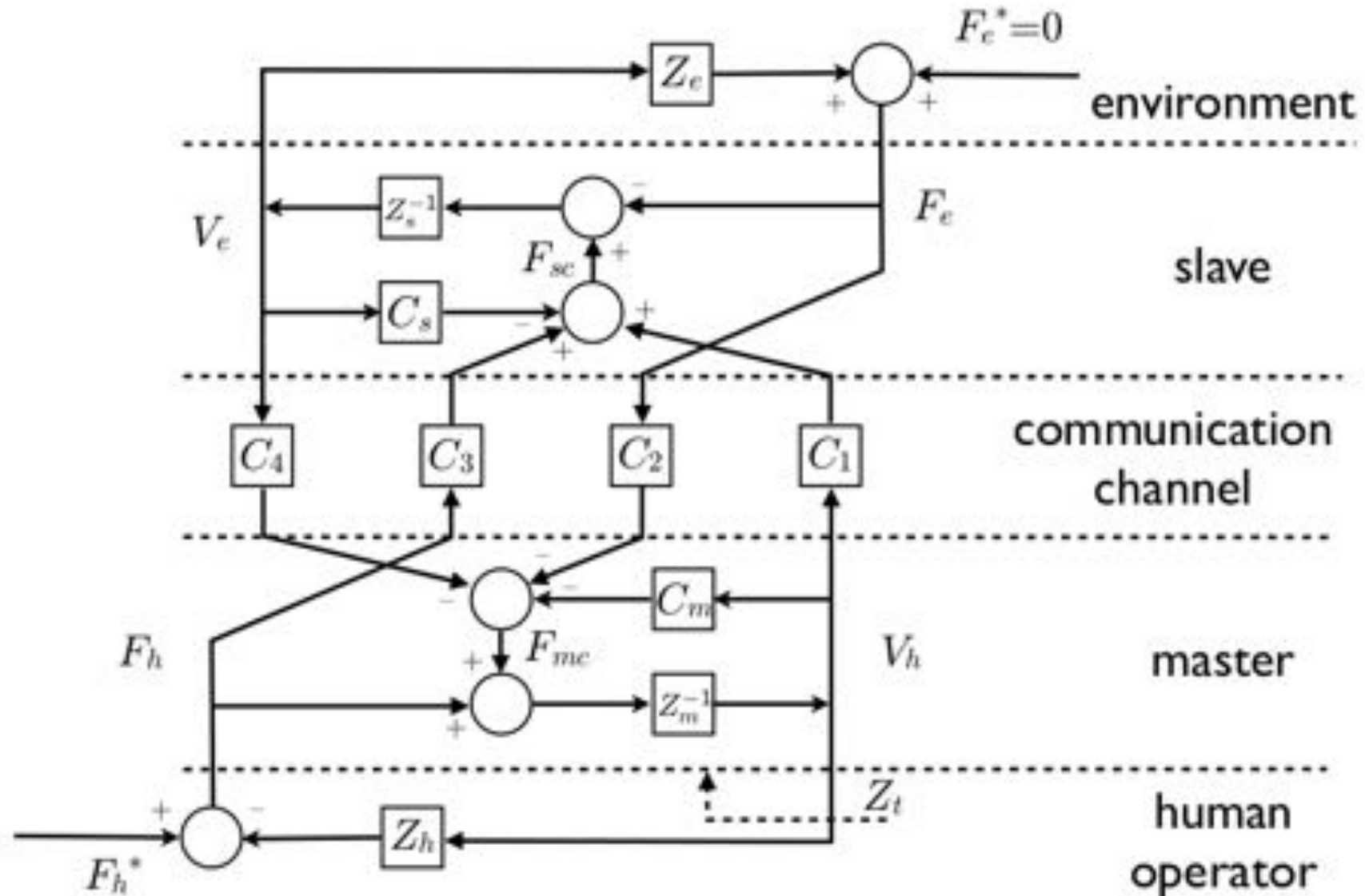
reference frames



slave = KUKA  
robot manipulator

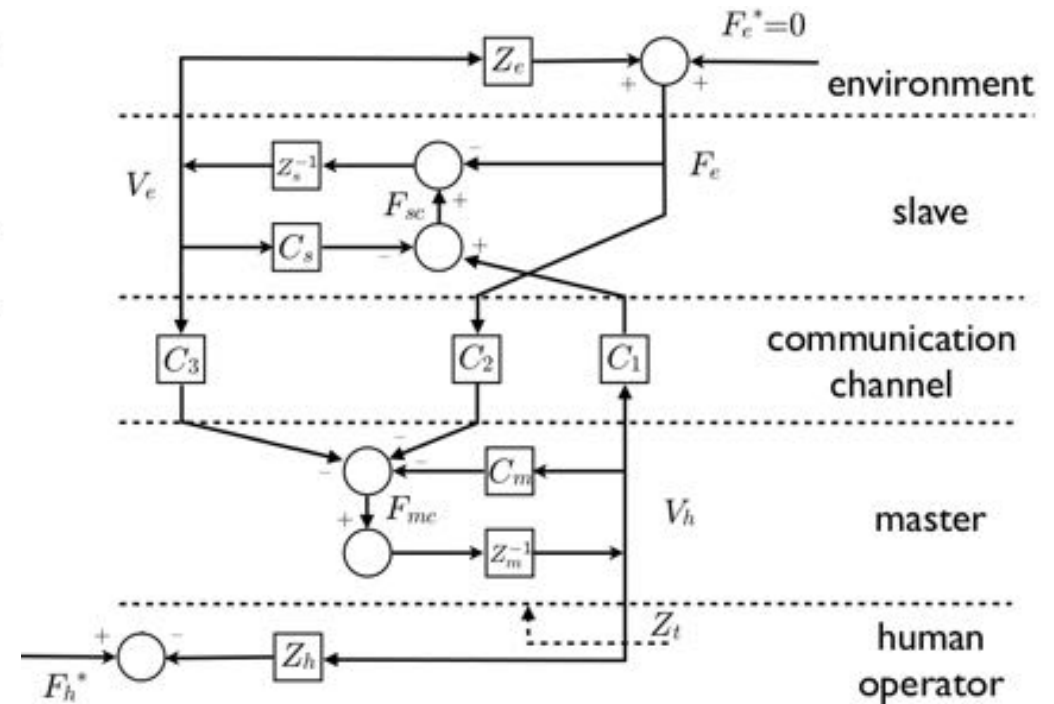
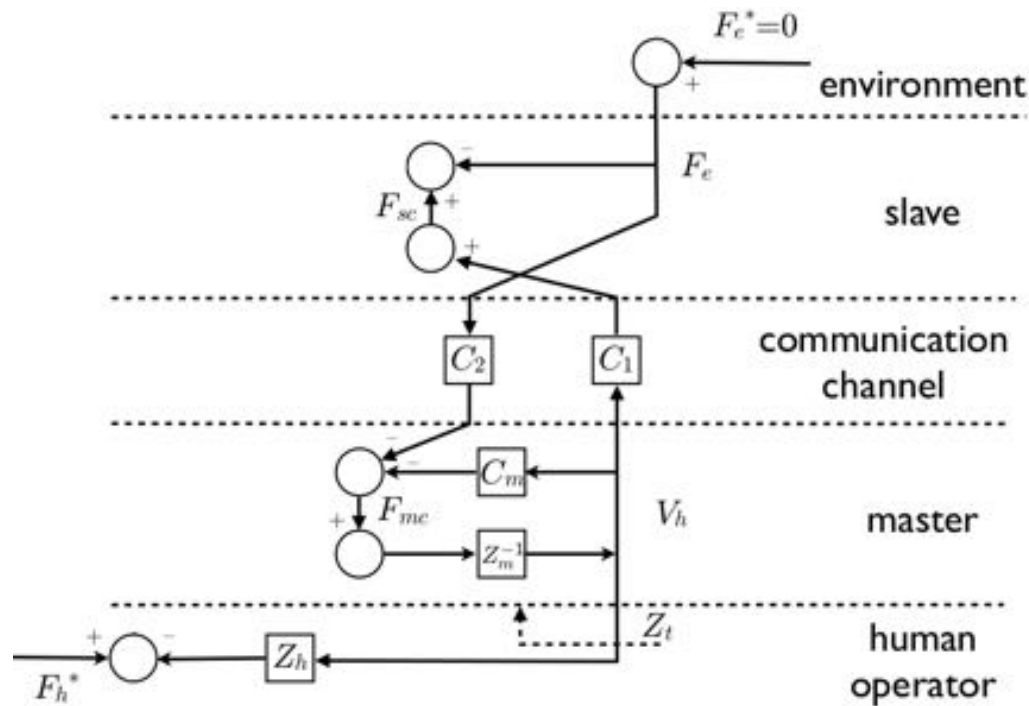
master = Touch  
haptic device

# Teleoperation control (4-way) general scheme



# Teleoperation control

## 2-way and final 3-way implemented solution







# Bibliography

---

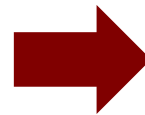
- B. Hannaford, A.M. Okamura, "Haptics", *Springer Handbook of Robotics* (O. Khatib, B. Siciliano, Eds.), Springer, 2008
- K. Salisbury, F. Conti, F. Barbagli, "Haptic rendering: Introductory concepts," *IEEE Computer Graphics and Applications*, vol. 24, pp. 24-32, 2004



# "hands on" trial

Medical Robotics & Robotics II - Project Report  
Academic year 2014/2015

First demo presented in April 2016 by  
Andrea Perica and Francesco Iodice



This year demo is on December 1, 2017  
16:00-18:00, c/o room A4  
introduced by Prof. Marilena Vendittelli

## Remote needle insertion with reconstructed force feedback

Evangelista Daniele<sup>1</sup>, Iodice Francesco<sup>2</sup>, Monorchio Luca<sup>3</sup>, Perica Andrea<sup>4</sup>

### Abstract

Telemedicine operation in medical robotics is the new frontier of surgery in which several factors must be taken into account, e.g. to ensure the health of the patient the sensibility of the force feedback that the robot sends to the surgeon must be reconstructed with extremely accuracy. This is a fundamental priority in this project, just think about how it can be helpful to feel physical contacts while staying in a different place, also very far away from the surgery room. Nevertheless, these applications are set to become the future of surgery entrusting a large part of the work from the surgeon to the medical robots.

<sup>1</sup> Master in Artificial Intelligence and Robotics, mat. 1665872

<sup>2</sup> Master in Artificial Intelligence and Robotics, mat. 1651209

<sup>3</sup> Master in Artificial Intelligence and Robotics, mat. 1650427

<sup>4</sup> Master in Control Engineering, mat. 1313336

### Contents

Introduction	1
1 Formalization of the problem	2
1.1 Teleoperation architectures	2
2 Our approach	3
2.1 Teleoperation problems to be solved	3
2.2 Slave velocity	4
2.3 Force feedback	4
External force • Actuated external force	
3 Experimental setup	5
3.1 The Geomagic Touch haptic device	5
3.2 The KUKA Light Weight Robot	6
3.3 Tools and materials	6
3.4 Experiments	6
Test on skin-fat-muscle	
4 Discussion	9
4.1 About the results and future works	9
Acknowledgments	9
References	9

### Introduction

In the last years robots have improved the quality of human life in many contexts, from factory to personal uses. However, today, fast technology evolution brings the human mind to think "how can this technology be used to improve safety and accuracy especially in medical applications?". Medical Robotics is the responsible branch for those problematics, starting from disease discovery to surgery applications. Advanced tools and sensors have been developed in the past years

of research, one example of medical robots, already in use in many surgery scenarios is the Da Vinci Surgical System. It is a surgical robot designed to facilitate complex surgeries using a minimally invasive approach like prostatectomies, cardiac valve repair and gynecologic surgical procedures.

The project that we are going to illustrate is called "Remote needle insertion with reconstructed force feedback", a telemedicine operation in which a haptic interface (from now called *master device*) manages the movements of a robot (from now called *slave device*) that will be responsible mainly for two tasks: needle insertion and force feedback response. Actually this application could be used from a surgeon to operate a patient from a different location with respect to the surgery room and, at the same time, perceiving all the forces coming from robot/patient interactions. The experiments have been done in the Laboratory of Robotics in the Dipartimento di Ingegneria Automatica e Gestionale at La Sapienza University of Rome.

The master device is the *Phantom Geomagic Touch* interface, a haptic device designed by Geomagic, instead the slave device is the *KUKA LWR Robot* developed by *KUKA Industries*.

In the next paragraphs the principal aspects of all the work will be illustrated, starting from formalization of the problem ending with the principal issues and solutions that we have been encountered during the way and some of the performed experiments. We will see how the obtained results show accurately that the needle insertion with feedback force reconstruction is an important aspect for the medical robotics applications, and lots of further improvements can be done in this direction.