Medical Robotics

SAPIENZA
UNIVERSITÀ DI ROMA

INTRODUCTION TO THE GEOMAGIC TOUCH HAPTIC DEVICE AND THE RELATIVE SOFTWARE LIBRARIES

HAPTIC comes from the greek word APTICOS, it means suitable portache and it refers to the capability to have a perception of the physical interactions between the object(forces).

HAPTIC INTERFACES

This is an important property(field) in medical robotics because during surgical operations the surgeon has to have a kind of feedback of tthe interaction between the environment. Very important in remote operations.

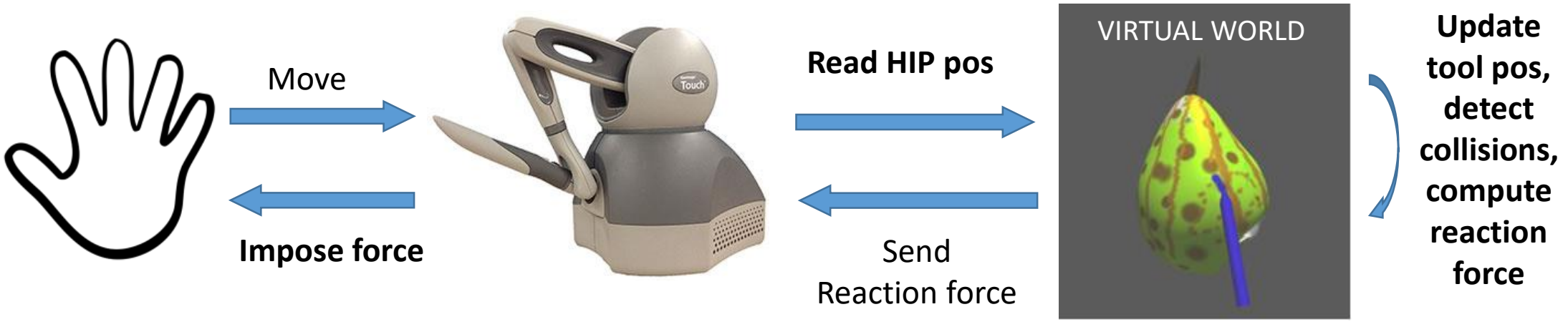
Robotic devices that allow physical interaction with a virtual environment or a teleoperated system.

geomagic touch device(we have this in the laboratories at the university)

How it works? the general idea is that we take an haptic interface point(HIP). It is a couple with a tool that can be in virtual space or in a correponding point in a teleoperated system. If we focus on the virtual space what we would like to achieve is that from the motion of this point there is a specific point on the kinematic structure of our haptic device we want to map the motion of this point to the motion of a specific virtual point inside a virtual world(virtual tool that we use).

HAPTIC RENDERING

Idea: HIP (Haptic Interface Point) coupled with a tool in virtual space. When the tool collides with a virtual object, the user feels an artificial reaction force.



Different kind of forces: Contact forces, Friction forces, Teleoperation forces, Effects..

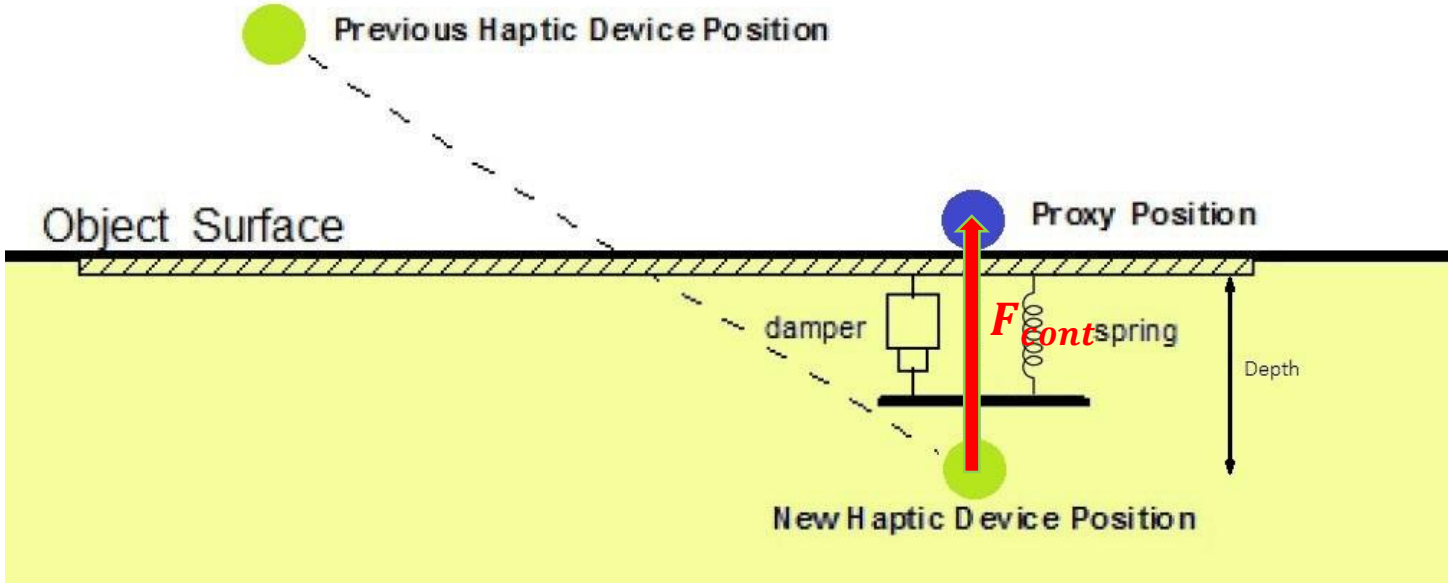
From one side we have the user that control and operates on the haptic device and moves its arm. When this happens, the device sends data about the motion of the haptic interface point to the virtual world or any other system that interacts with it. By moving the haptic devide we can do different operation or detect collision or compute reaction forces that can send back to the user.

Contact Forces

We focus on how we can model contact forces. By considering a virtual point that is supposed to interact with the surface of an object. The idea is that without making any particular modifications to the motion of this point we can have the controlled point can go through the surface of the object and enter inside the surface itself. Force that we want to model is related to the depth of insertion.

Interaction between virtual HIP (vHIP) and virtual object = spring-damper system

we can compute a new position for our controlled object

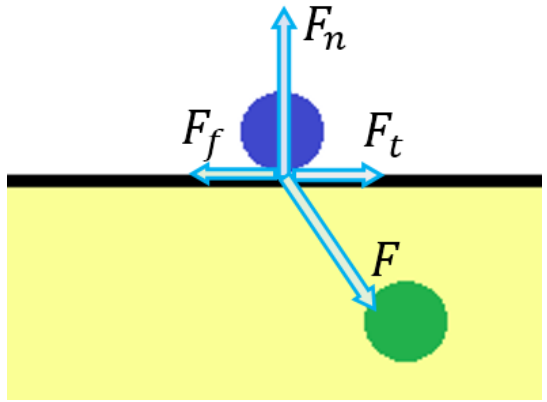


When vHIP inside object:

- Compute Proxy point
- Direct the reaction force from the vHIP toward the proxy
- $F_{cont} = k * depth + b * vel$
  depth = depth of penetration
  vel = velocity of the vHIP

How the haptic device is able to render the forces to the user? The trick is that the device itself can be condirered like a joystick in a way, because we have an operator that takes the stilus of the device and moves it.

Friction Forces



they can be designed in arbitrary way by assigning some kind of parameters

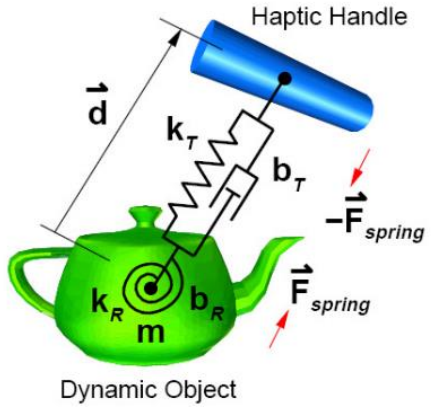
Static and dynamic friction forces are computed on the proxy

$$F_{f_s} = \mu_s * F_n$$
$$F_{f_d} = \mu_d * F_n$$

no movement if $F_{f_s} > F_t$

- Slight modification of the proxy algorithm for contact forces

Virtual Coupling Forces



virtual haptic interface point

"Feel" the inertia of a virtual body connected to the vHIP.

- Typical of position/position teleoperation schemes.

Force Effects

Forces that arise from the movement in a portion of the virtual space.
Not linked to any virtual object.

- Viscous friction
- Stick-slip effect
- Vibration

we can have frictions or vibrations

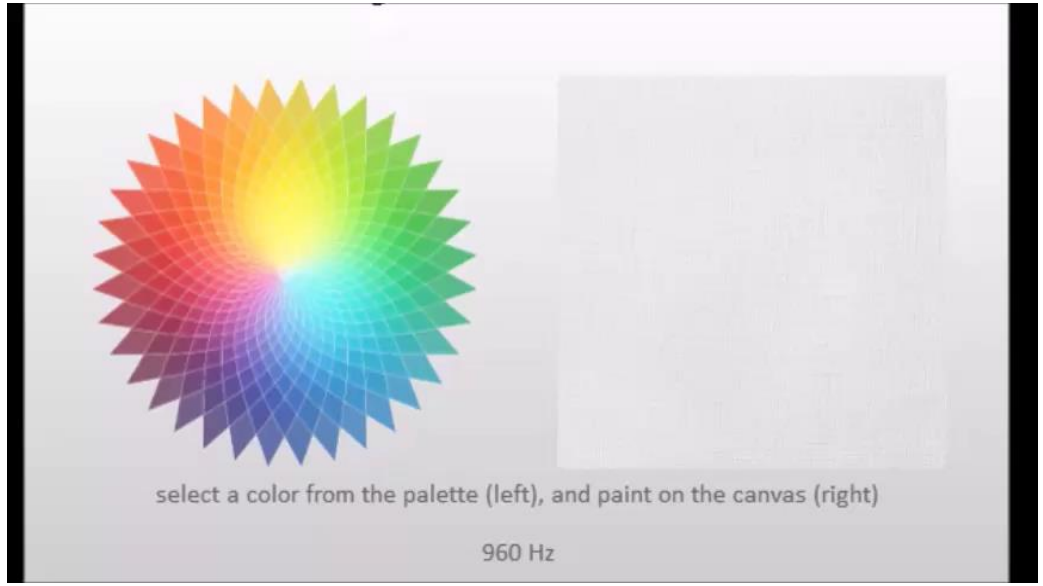
Constraints = Virtual Fixtures

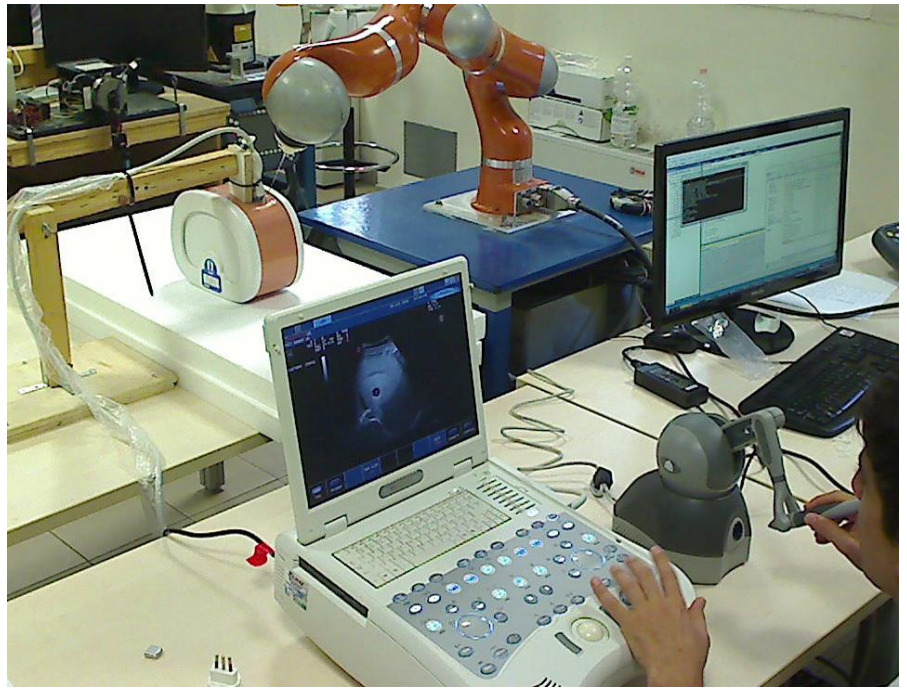
Attractive/Repulsive forces that constraint the vHIP (then the HIP!) to a specific geometric region (dot, line, plane, surface) in the virtual environment.

- Elastic/Magnetic attraction toward a constraint
- Viscous friction in forbidden directions

Applications of Haptic Interfaces

- Touch surfaces
- Manipulate objects (move, deform, decompose)
- Create new objects (object carving, 3D painting)
- Navigation in virtual environments
- Teleoperation of virtual or real robots
- Coordination of a team of robots





Examples from CHAI3D:
'Textures', 'Magnets', 'Effects',
'Paint' , 'Map' , 'Object' , 'Voxel'

Difference between the type of force feedback that can be produced and tthe sensing capabilities. If you look at the datasheet you can view that the forces feedback produced by the geomagic touch is on x, y, z axis (3-dof) while for the position sensing we can measure both the position x, y, z and the roll, pitch, yaw angle for the orientation. This comes from the fact that the whole kine structure has 6-dof.

GEOMAGIC TOUCH

It has the same structure of the anthropomorphic arm

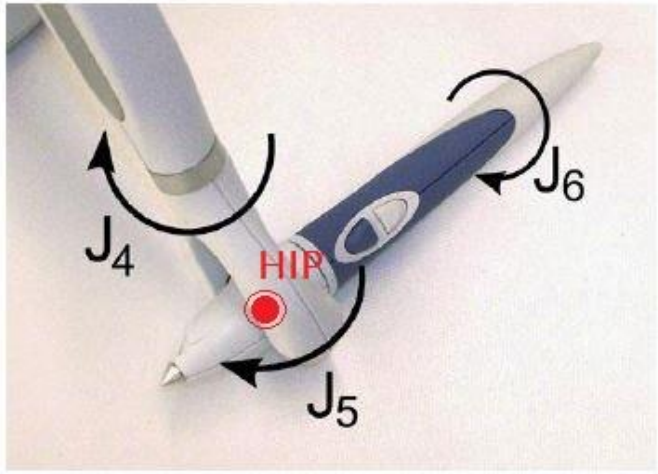
- Geomagic Touch by 3D System (previously called Phantom Omni by SensAble Technologies)
- 2 buttons at the end of the stilus that can be used to map different actions from the user
- Compatible with OpenHaptics 3.4 and CHAI3D libraries.

Workspace	~6.4 W x 4.8 H x 2.8 D in > 160 W x 120 H x 70 D mm
Range of motion	Hand movement pivoting at wrist
Nominal position resolution	> 450 dpi ~0.055 mm
Maximum exertable force and torque at nominal position (orthogonal arms)	0.75 lbf/3.3 N
Stiffness	x-axis > 7.3 lb/in (1.26 N/mm) y-axis > 13.4 lb/in (2.31 N/mm) z-axis > 5.9 lb/in (1.02 N/mm)
Force feedback	x, y, z
Position sensing/input the whole structure has 6-dof **[Stylus gimbal]**	x, y, z (digital encoders) [Roll, pitch, yaw (± 5% linearity potentiometers)]
Interface	RJ45 Compliant Ethernet Port

GEOMAGIC TOUCH

anthropomorphic arm

First three joints are actuated



Last three joints form a passive spherical wrist

They cannot be actuated but the position can be measured from the device

HIP (Haptic Interface Point) is the center of the spherical wrist

OpenHaptics Toolkit 3.4

OpenHaptics®
Developer Edition

Proprietary (3DSystems) C++ libraries that allow:

- Interfacing with all the haptic devices of the Geomagic family (by 3DSystems)
- Development of software for interaction with a virtual environment through the interface
- Integrated functions for graphic and haptic rendering of the virtual scene

3 groups of API from high-level to low-level: **QuickHaptics**, **HLAPI**, **HDAPI**

ease of use ←

level of customization →

Quick haptics

Briefly: High-level API. Easy to use. User just has to define the shapes using given function. Graphic and haptic rendering managed by OpenHaptics.

What you can do with QuickHaptics:

- Create simple shapes
- Interact with the shapes graphically and haptically
- Use pre-defined force effects
- Define callbacks on contact event

QUICK HAPTICS (2)

What you are required to do:

- Create a window where to graphically render the scene
- Define the device space
- Define the shapes and their properties
- Define the cursor (collect informations about device and shapes)

What QuickHaptics will manage:

- Insertion of many default parameters
- Communication with the device
- Graphic and haptic rendering
- Collisions
- Synchronization between graphic rendering loop (30Hz) and haptic rendering loop (1000Hz)

Disadvantages: API functions are very “closed”, you don’t have access to many informations and cannot perform complex tasks.

HLAPI (HAPTIC LIBRARY API)

Briefly: User has to define shapes and graphic rendering using OpenGL formalism. Haptic rendering managed by OpenHaptics.

What you can do with HLAPI:

- Create complex graphic and haptic scene using OpenGL formalism
- Define custom callbacks for a set of events
- Define constraints (virtual fixtures)

What you are required to do:

- Define shapes using openGL formalism (vertices, edges, transformations, material properties). The physical properties are treated as classical OpenGl properties

What HLAPI will manage:

- HLAPI uses graphic informations to compute collisions and haptic rendering
- Communication with the device
- Synchronization of graphic and haptic rendering

HDAPI (Haptic Device API)

Briefly: Low-level API. Difficult to use. User must define shapes, graphic rendering (OpenGL) and also collisions and haptic rendering. You can do a lot but not everything works as expected..

What you can do with HDAPI:

- Directly impose custom forces to the device
- Read low-level sensor data from the device (joint velocities, torques)
- Define custom haptic rendering and collision detection algorithms
- Define custom effects

What you are required to do:

- Separately define functions for graphics rendering, collision detection and haptic rendering
- Use efficient collision detection and haptic rendering algorithms (they run once for every haptic loop iteration)
- Manage the synchronization of the graphic loop (30Hz) and the haptic loop (1000Hz)

What HDAPI will manage:

- Initialization and communication with the device

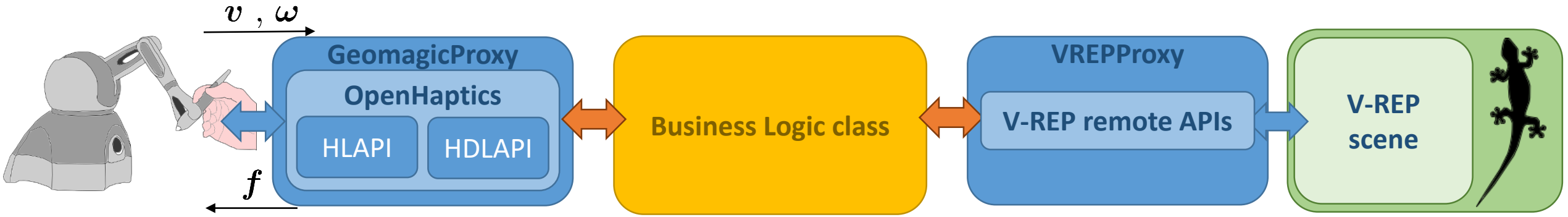
USE OPENHAPTICS IN OUR CODE
A LOOK TO THE PAST: THE CHAI3D LIBRARY

- CHAI3D = COMPUTER HAPTICS AND ACTIVE INTERFACE
- Open-source multi-platform set of C++ libraries for haptics and visualization
- Released in 2004 by AI Lab at Stanford University + EPFL + University of Siena

Features:

- Allows communication with several haptic devices (both 3 and 6 dof)
- Manages graphic and haptic rendering. Has its own graphic engine (OpenGL based), provides multiple collision detection and force rendering algorithms
- Allow to read/impose values to/from the haptic device
- Has a clear documentation
- Provides integration modules for third party softwares like V-REP
- Apparently not maintained recently and no more compatible with recent V-REP versions (>3.3.2)!

ALTERNATIVE CUSTOMIZED SOLUTION FOR GEOMAGIC-VREP COMMUNICATION



The motor inside the device is actuated by setting a specific force that at a certain point can come.

GeomagicProxy class

- query OpenHaptics and update of the device *state* (i.e., HIP position, orientation, velocity)
- Request the actuation of the device motors, to render the input commanded force
- send information to the higher-level *BL* classes (i.e., implementing a specific Task or Control)

The operator moves the pen. By moving the pen the geomagic proxy queries and catches info on the velocity of the sitlus. This velocity is sent to the businness logic class, this class processes this velocity info and actually transfers or converts this velocity to a virtual object in the simulator by asking V-REP through VREProxy class to set specific velocity or position to a given object.

BL classes

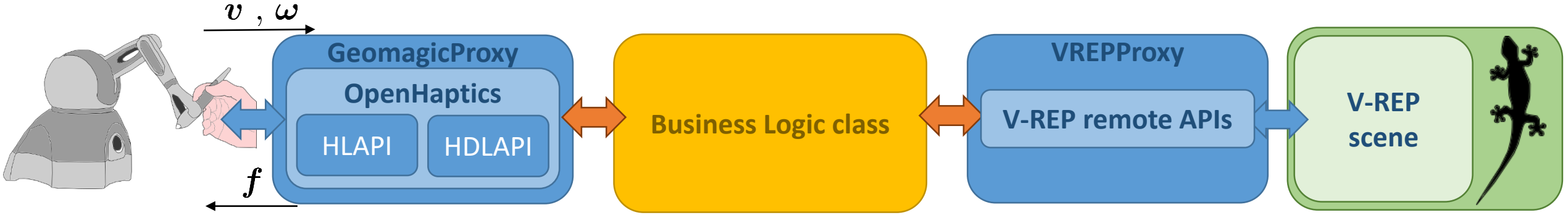
- process data from both *GeomagicProxy* and *VREPProxy*
- Act as a "*bridge*" allowing Geomagic and V-REP to communicate

VREPProxy class

- Implement requests received from BL classes (e.g., set virtual objects position)

The force is not computed by the geometric proxy but the computation is left to the businnes logic class based on chosen model and types of interaction.

ALTERNATIVE CUSTOMIZED SOLUTION FOR GEOMAGIC-VREP COMMUNICATION




```
/**** GeomagicProxy.hpp ****/
HDCallbackCode HDCALLBACK updateStateCallback(void* data); // internally set HapticStatus
HDCallbackCode HDCALLBACK forceFeedbackCallback(void* data); // Send force commands
                                                             // to the device motors
class GeomagicProxy{
   //...
   struct HapticStatus{
      hduVector3d stylusPosition;
      hduVector3d stylusOrientation;
      hduVector3d stylusLinearVelocity;
      hduVector3d stylusAngularVelocity;
      hduVector3d force;
      int stylusButtons;
   } geoStatus;

   inline HapticStatus getHapticStatus() {return this->geoStatus;}
   inline void setHapticForce(const hduVector3d& f) {this->geoStatus.force = f;}
}
```

```
/**** GeomagicProxy.cpp ****/
// Callbacks call
hdScheduleAsynchronous(forceFeedbackCallback, this, HD_MAX_SCHEDULER_PRIORITY);
while(geo.isRunning()){ // isRunning() evaluates a boolean condition to run the Geomagic thread
   hdScheduleSynchronous(updateStateCallback, this, HD_DEFAULT_SCHEDULER_PRIORITY);
}
```

```
/** Any Business Logic level class «using» GeomagicProxy and VREPProxy classes **/
GeomagicProxy geoProxy;
VREPProxy vrep;

while(ctrl.isRunning()){
   // Read the current status of the Geomagic device
   HapticStatus status = geoProxy.getHapticStatus();
   // Set the force vector on the Geomagic device
   geoProxy.setHapticForce(cmdForce); // assume cmdForce computed somewhere else
   // (Example) Transfer the Geomagic stylus motion on a virtual object in V-REP
   vrep.setObjectPosition(objectID,status.stylusPosition);
}
```

System Setting

Besides the device is recognized both in Windows and Ubuntu, the following procedure has been used and tested on Windows 10 only.

The following procedure is explained in detail in the appendix guide.

1. Download and install Geomagic touch drivers
2. Download and install OpenHaptics (it requires Microsoft Visual Studio)
3. Do the pairing and calibration procedure (Geomagic Driver)
4. Run the compiled program and have fun

APPENDIX:
GUIDE TO INSTALLATION OF GEOMAGIC DRIVER, OPENHAPTICS AND CHAI3D

Notes:

- Besides the device is recognized both in Windows and Ubuntu, the following procedure has been used and tested on Windows 10 only.
- OpenHaptics requires Microsoft Visual Studio 2005/2008/2010, while CHAI3D requires Microsoft Visual Studio 2012/2013/2015.

INSTALLATION OF GEOMAGIC DRIVER AND OPENHAPTICS

1. From *https://3dsystems.teamplatform.com/pages/102774?t=r4nk8zvqwa91* download the Geomagic touch driver for windows '*Geomagic_Touch_Device_Driver_2016.1.1.exe*' and the OpenHaptics v3.4 '*OpenHaptics_Developer_Edition_v3.4.0.zip*'
2. Install the Geomagic touch driver using the install (.exe)
3. Install the OpenHaptics using the install (.exe)

GUIDE TO INSTALLATION OF GEOMAGIC DRIVER, OPENHAPTICS AND CHAI3D

INSTALLATION OF CHAI3D

Note-1: CHAI3D requires Geomagic drivers but not nevessarily OpenHaptics. To use chai3d module for V-REP you need V-REP already installed in your computer (download from here *http://www.coppeliarobotics.com/downloads.html*).

Note-2: Highest V-REP compatible version for chai3d is 3.3.2! The following procedure is not guaranteed to work for V-REP versions > 3.3.2.

1. From *http://www.chai3d.org/download/releases* download the '*Multiplatform (Windows / Linux/ Mac OS X)*' version of CHAI3D
2. CHAI3D does not require installation, then unzip the archive in the location where you want the main folder of CHAI3D to be (e.g. '*C:*\')
3. In the '*doc*' folder you will find the documentation of CHAI3D. Please open '*getting-started.html*' and select '*on Windows visual studio*' to get the instructions to compile the CHAI3D examples. The 'root folder' mentioned in the instruction is the folder '*chai3d-3.1.1*' which contains all the other chai3d folders (included '*doc*').
4. In the same way you can compile the examples relative to the external modules (eg V-REP). For V-REP, go in '*chai3d-3.1.1/modules/V-REP/doc/getting-started*' and follow the instructions to compile the chai3d-vrep examples (similar to point 3).

INTERFACE THE DEVICE WITH PC, USE OPENHAPTICS AND CHAI3D

CONNECT THE DEVICE TO THE PC:

1. Connect the device Ethernet port 1 with the Ethernet port of the pc
2. In the installation directory of Geomagic DRIVERS (usually '*programs\3DSystems\Geomagic*'…) launch '*Geomagic_Touch_Setup.exe*'. Select '*Geomagic Touch*' under the '*Device Model*' box, click '*pairing*' and fastly press the button on the back of the Geomagic Touch. Click '*apply*' and exit.
3. In the installation directory of Geomagic DRIVERS (usually *'programs\3DSystems\Geomagic*'…) launch *'Geomagic_Touch_Diagnostic.exe*', click on the right arrow which is at the bottom-right of the window, until you arrive at the '*Calibration*' page. Wait for the '*Calibrate*' button to become green (or force calibration by clicking on it).
4. In case of 'communication error' please disconnect the Ethernet cable and repeat the procedure from point 2

WORK WITH OPENHAPTICS:

- The reference folder usually is '*C:\OpenHaptics\Developer\3.4.0*' . In this folder you can find the documentation ('*docs*') and useful examples ('*examples*').

INTERFACE THE DEVICE WITH PC, USE OPENHAPTICS AND CHAI3D

- The examples' binaries are contained in '*examples\bin*' , while the sources are in '*examples\HL*' and '*examples\HD*'.
- Use Visual studio 2005, 2008, 2010 to open an example project (e.g. use Visual Studio 2010 to open the file '*examples\HL\graphics\Events\Events_VS2010.vcxproj*')
- If you modify and compile the previous example for x64 systems, the generated binary program will be saved in examples\HL\bin\x64\HL\.
- A good way to start using OpenHaptics is to create a copy of an example folder (in the '*examples*' folder itself!!!), modify the example with Visual Studio, compile and see what happens..

WORK WITH CHAI3D

- The root folder usually is 'chai3d-3.1.1' that you have extracted from the archive during the installation of CHAI3D. In this folder you can find the documentation ('*docs*'), useful examples ('*example*') and the binaries of the compiled examples (*'bin'*).
- As before use Visual Studio 2012/2013/2015 to open an example project, modify it, and compile it. The binaries will be saved in '*chai3d-3.1.1\bin*'.