



Overview

- Introduction to Kinematics
- Joint Motor Control ([overview](#) + [tutorial](#))
- Cartesian Control ([theory](#) + [API](#) + [tutorial](#))
- Gaze Control ([theory](#) + [API](#) + [tutorial](#))
- Assignment



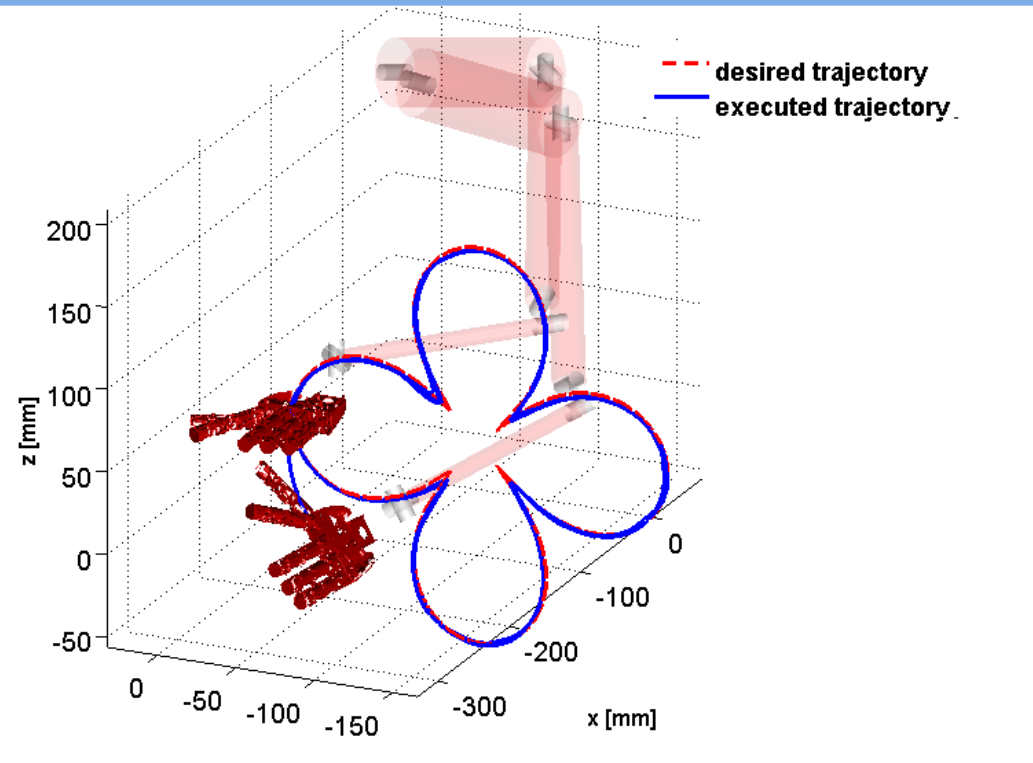
Basic Kinematics (1/2)

Study of properties of motion (position, velocity, acceleration) without considering body inertias and forces

The Problem

$$\begin{cases} \mathbf{x} = \mathbf{f}(\mathbf{q}) \\ \mathbf{q} \in \mathbb{R}^n \\ \mathbf{x} \in \mathbb{R}^6 \end{cases}$$

$$\mathbf{q} \stackrel{?}{=} \mathbf{f}^{-1}(\mathbf{x})$$





Basic Kinematics (2/2)

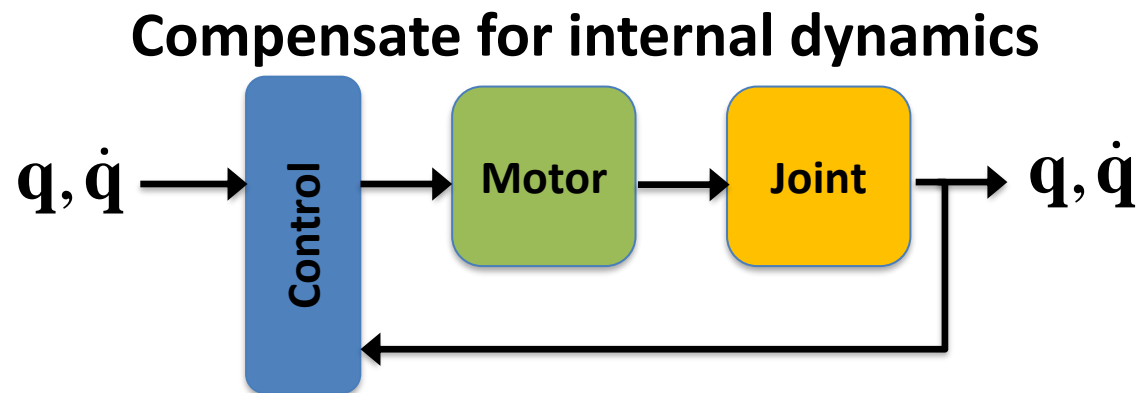
Dynamics – forces, torques, inertias, energy, contact with environment

$$\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}_v\dot{\mathbf{q}} + \mathbf{F}_s \operatorname{sgn}(\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} - \mathbf{J}^T(\mathbf{q})\mathbf{h}_e$$

VS.

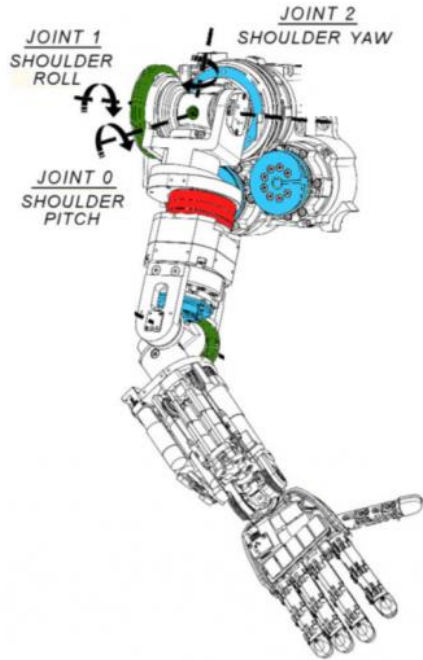
Kinematics – pure motion imposed to the manipulator's joints

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}} \\ \mathbf{J} = \partial \mathbf{f} / \partial \mathbf{q} \end{cases}$$





Joint Motor Control (1/4)



DC MOTORS



USER CODE @ 100 Hz

```
ipos->positionMove(poss.data());  
while (norm(poss-encs)>1.0) {  
    Time::delay(0.1);  
    ienc->getEncoders(encs.data());  
    cout<<encs.toString()<<endl;  
}
```

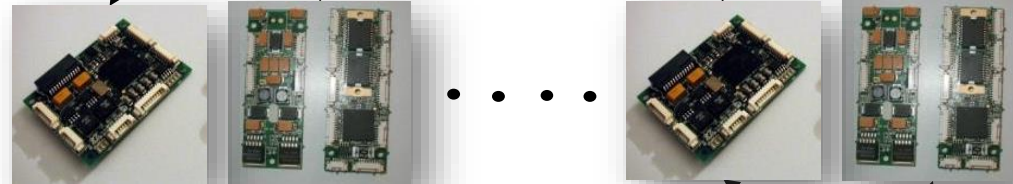


Gbit Ethernet

PC104

FIRMWARE @ 1 KHz

CONTROLLERS

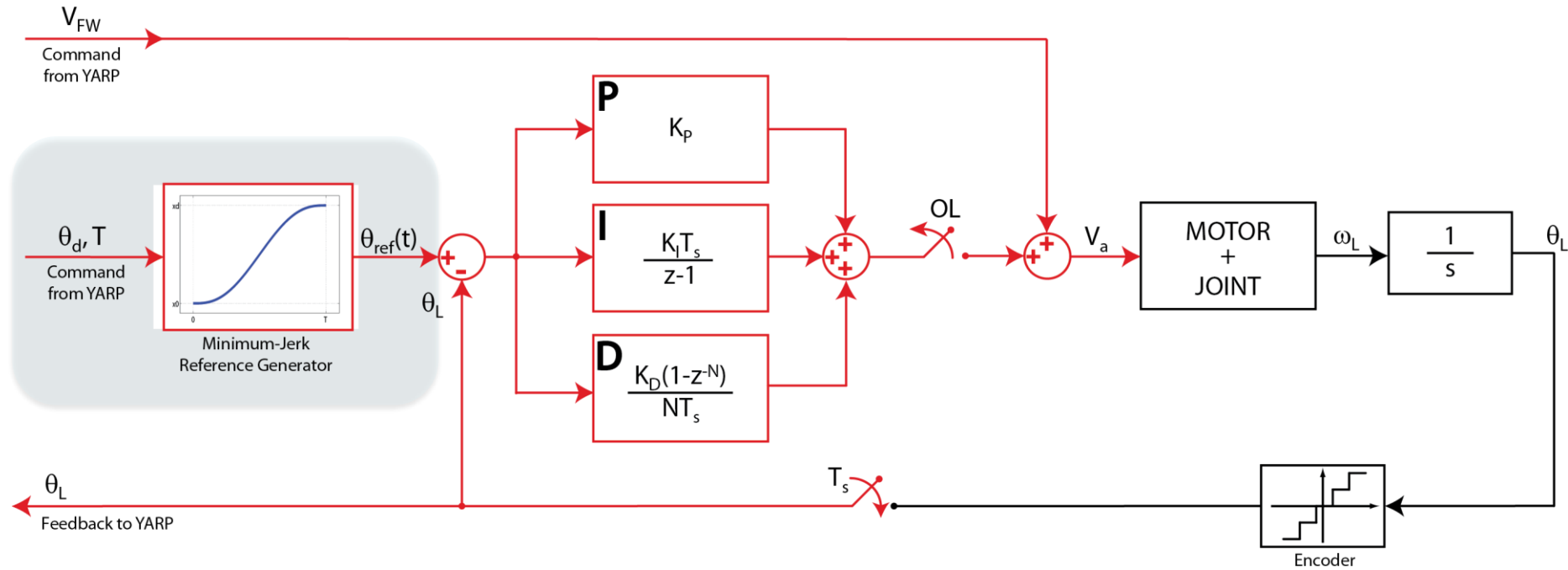


Sensors & Actuators



Joint Motor Control (2/4)

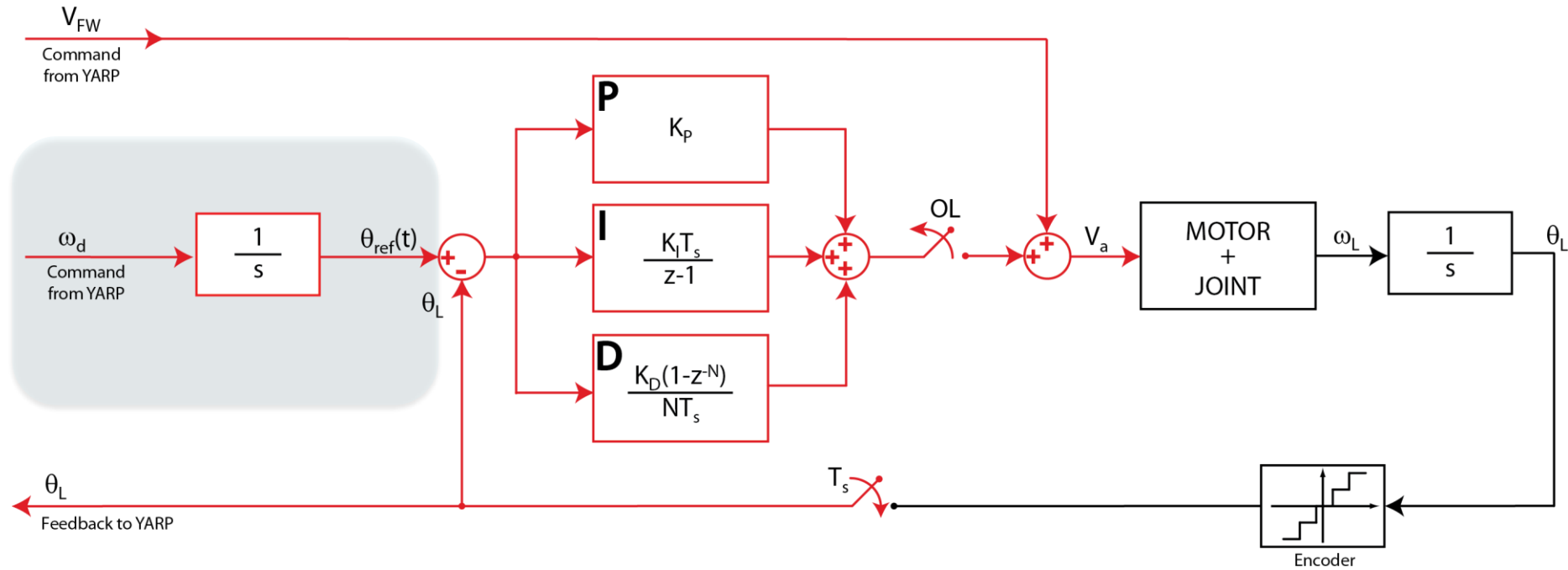
Position Control





Joint Motor Control (3/4)

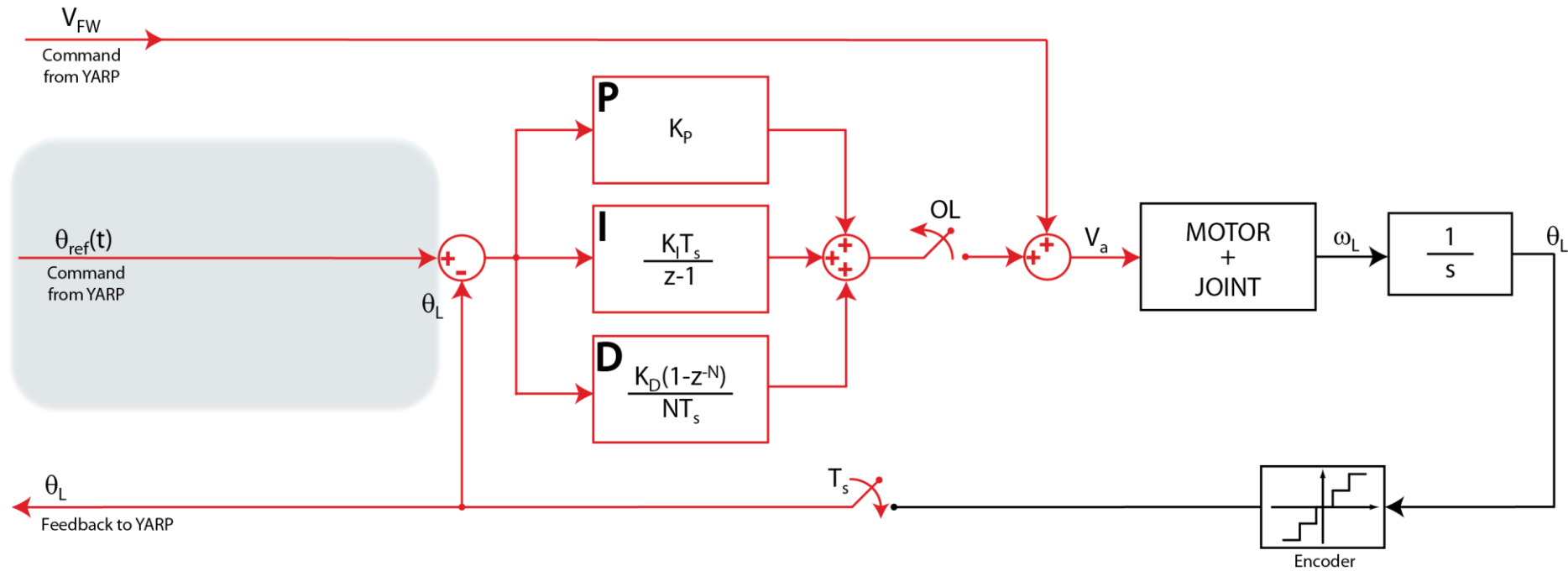
Velocity Control





Joint Motor Control (4/4)

Position-Direct Control





Tutorial Time !



The Cartesian Controller (1/5)

Operational (Cartesian) Space Control

You know \mathbf{x} (3D/6D points), you cannot control directly the motors, you have to solve the **Inverse Kinematics (IK) problem** beforehand.



very “robotic” movements:

- snap onset
- exponential decay

Jacobian Transpose

$$\dot{\mathbf{q}} = \mathbf{J}^T \mathbf{K} \mathbf{e}, \quad \mathbf{e} = \mathbf{x}_d - \mathbf{x}_e$$

Jacobian Pseudoinverse

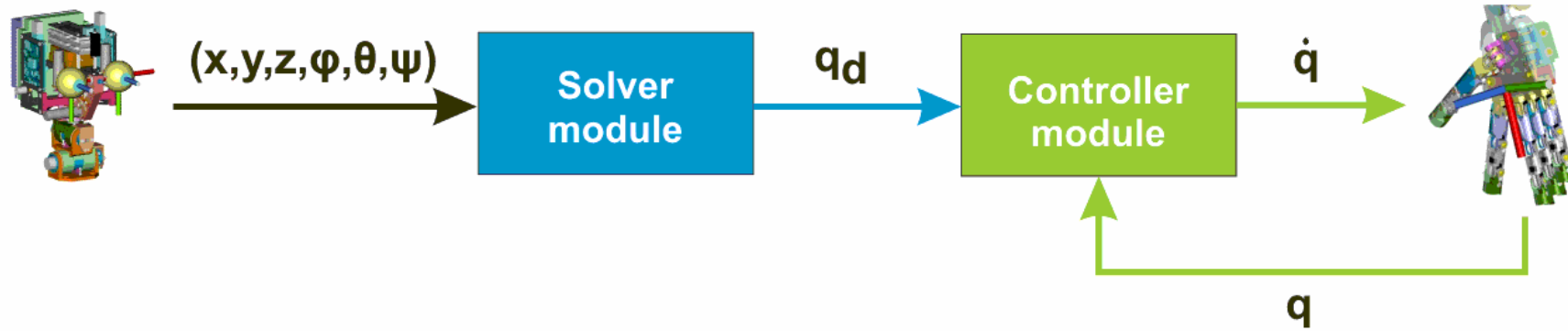
$$\dot{\mathbf{q}} = \mathbf{J}^\dagger \mathbf{K} \mathbf{e} + (\mathbf{I} - \mathbf{J}^\dagger \mathbf{J}) \dot{\mathbf{q}}_0, \quad \mathbf{J}^\dagger = \mathbf{J}^T (\mathbf{J} \mathbf{J}^T)^{-1}$$

Damped Least-Squares

$$\dot{\mathbf{q}} = \mathbf{J}^* \mathbf{K} \mathbf{e}, \quad \mathbf{J}^* = \mathbf{J}^T (\mathbf{J} \mathbf{J}^T + \lambda^2 \mathbf{I})^{-1}$$



The Cartesian Controller (2/5)



nonlinear constrained optimization:

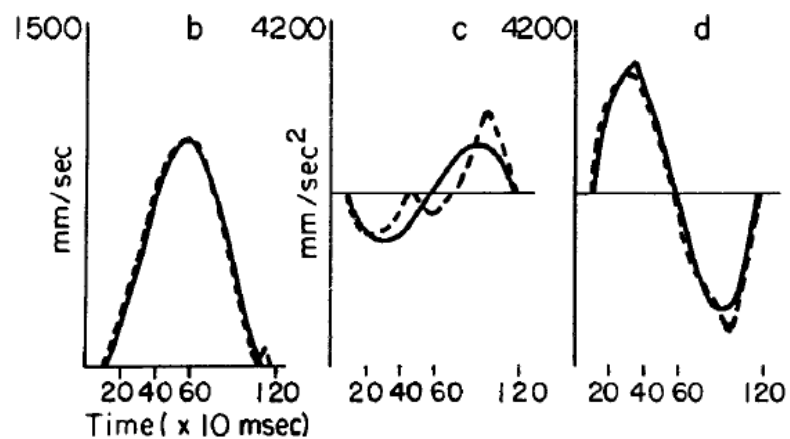
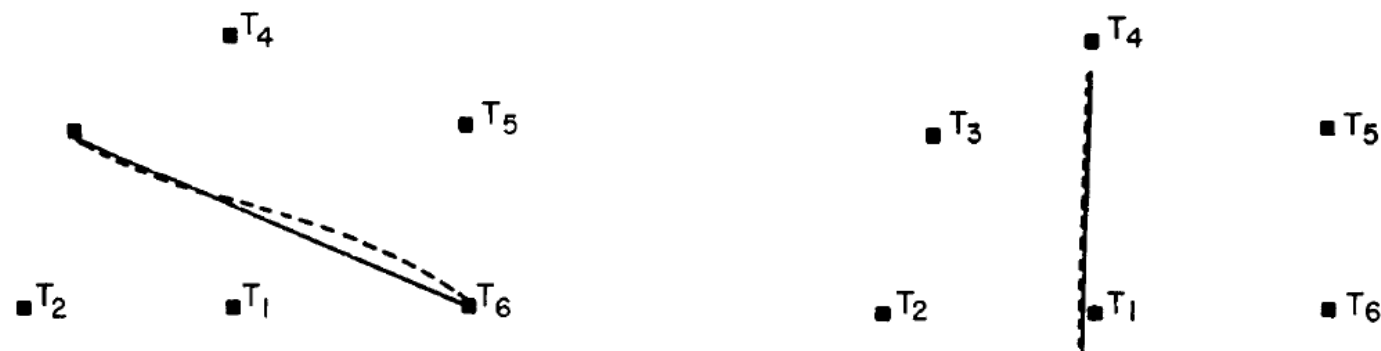
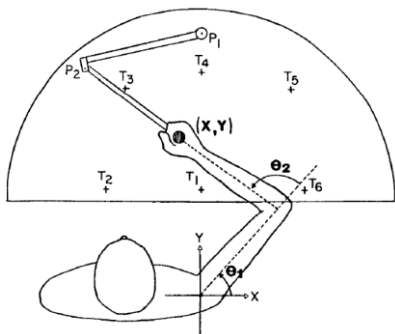
- fast
- scalable

human-like movements (min-jerk):

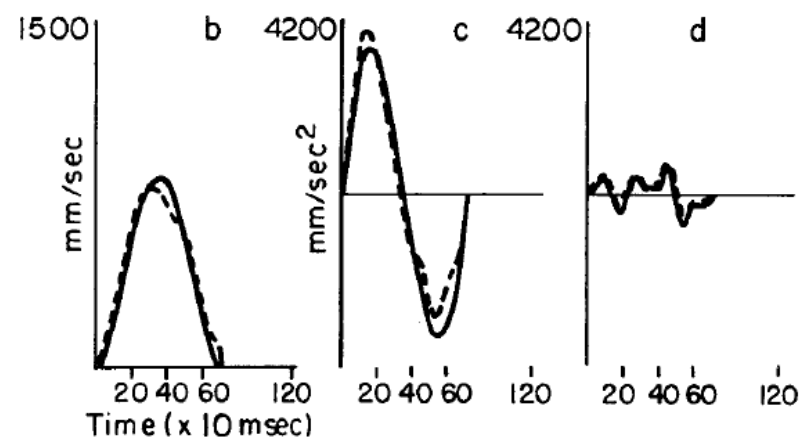
- bell-shape velocity profile
- quasi-straight path



The Cartesian Controller (3/5)



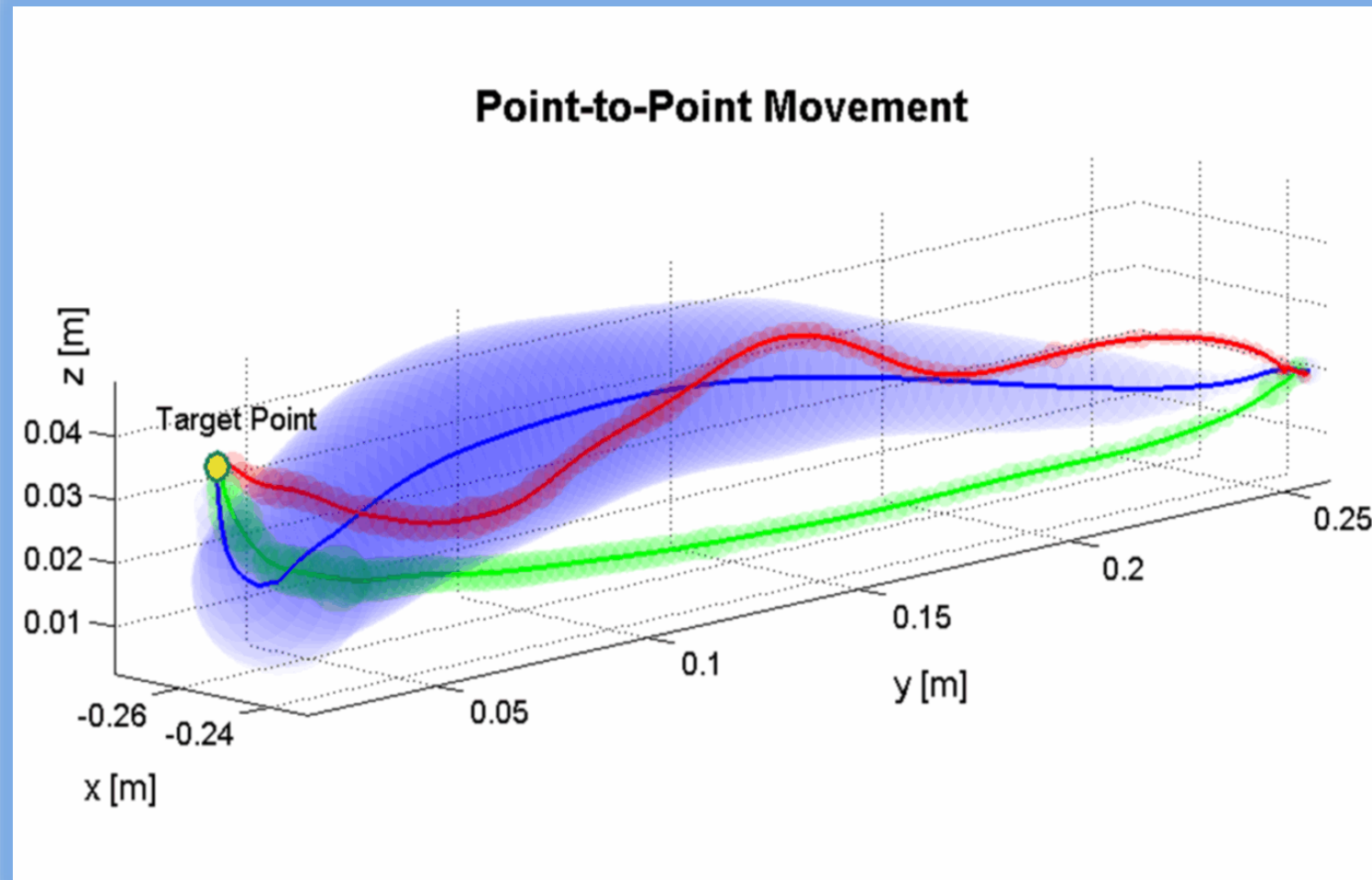
A



B



The Cartesian Controller (4/5)



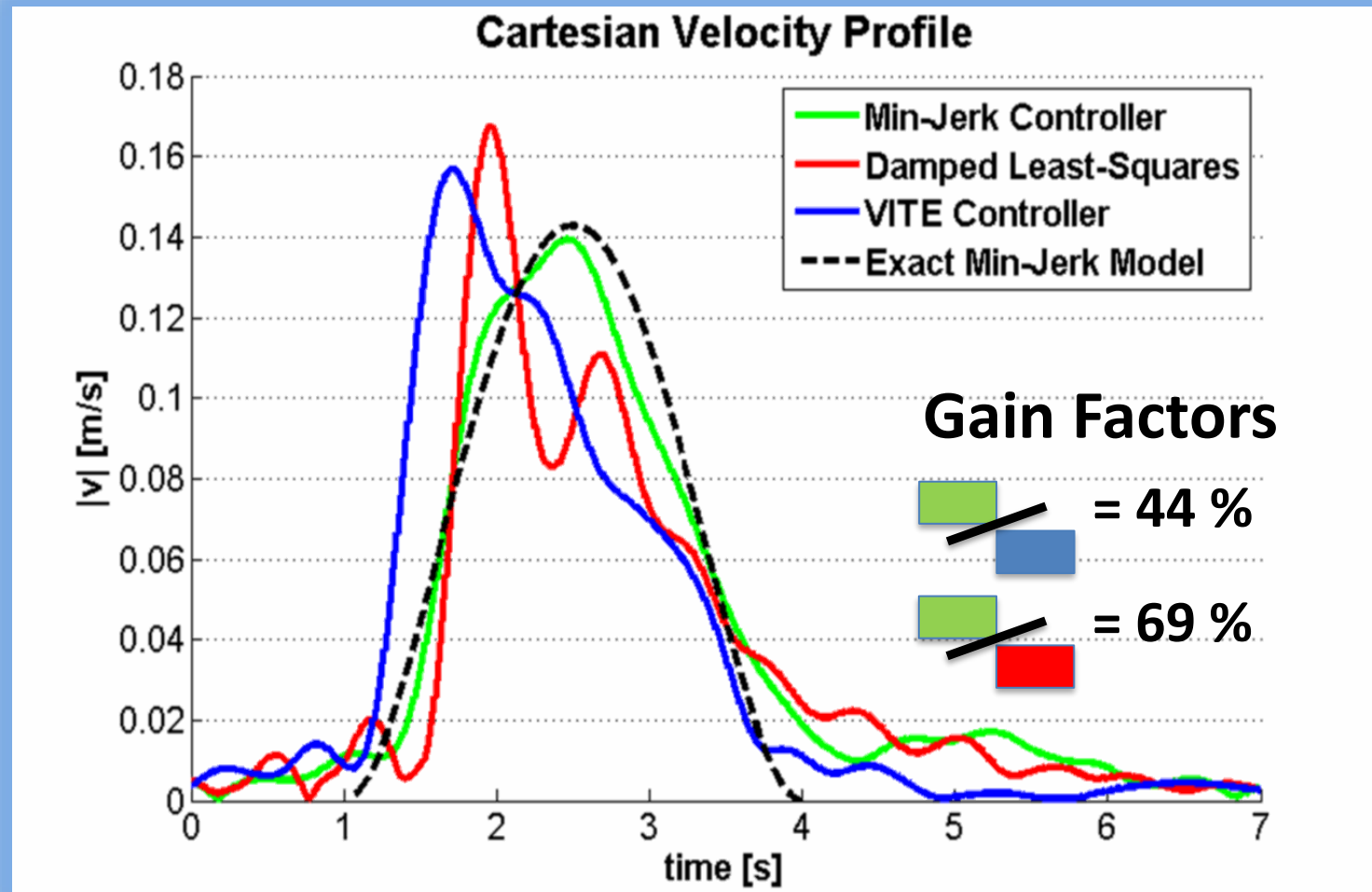
Min-Jerk

DLS

VITE



The Cartesian Controller (5/5)



Min-Jerk

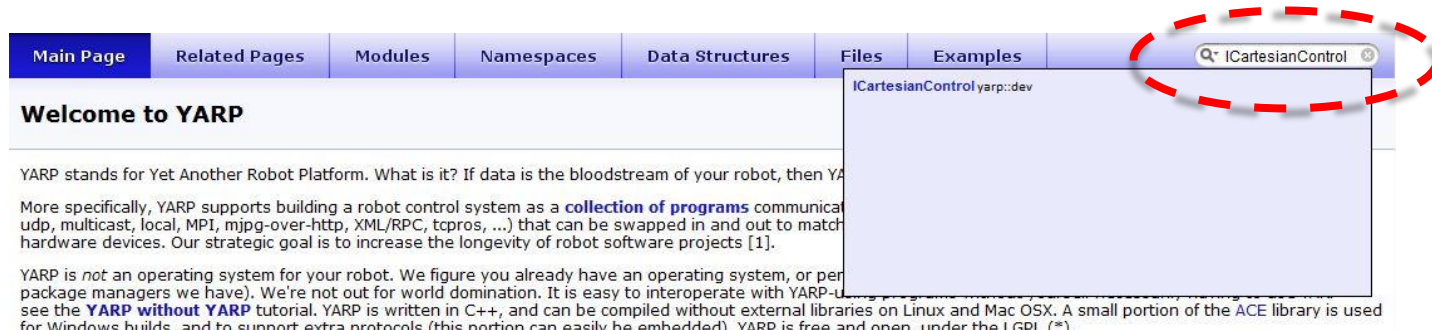
DLS

VITE



Interfaces Documentation

In the search field: type **ICartesianControl/IGazeControl**



The screenshot shows the YARP documentation website. The top navigation bar includes links for Main Page, Related Pages, Modules, Namespaces, Data Structures, Files, and Examples. A search bar in the top right corner contains the text "ICartesianControl". Below the navigation bar, the "Welcome to YARP" section is visible, followed by a description of YARP and its capabilities. The search results for "ICartesianControl" are displayed in a dropdown menu, showing the full path "ICartesianControl yarp::dev".

...

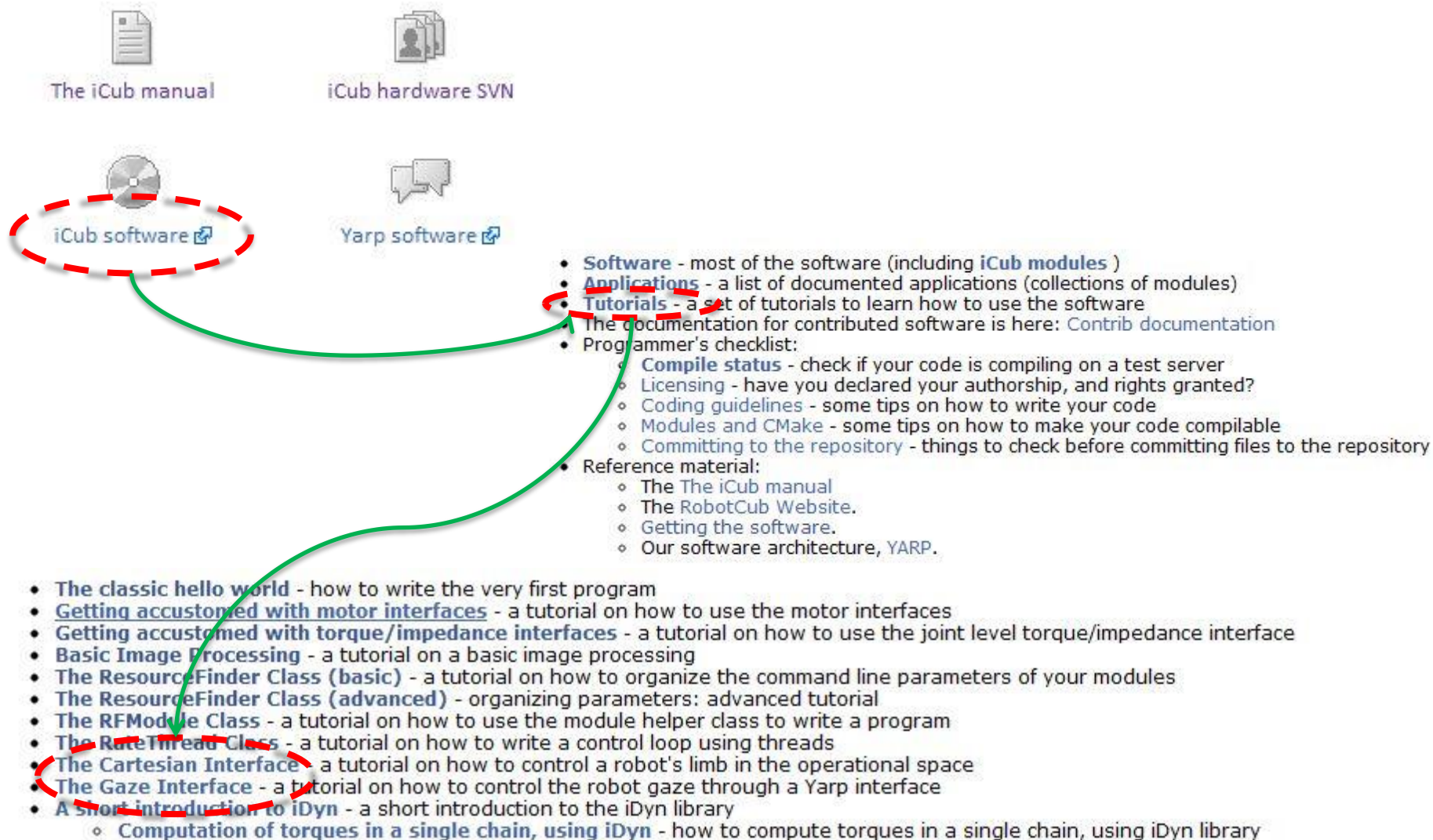
Public Member Functions

virtual	<code>~ICartesianControl()</code>	Destructor.
virtual bool	<code>setTrackingMode(const bool f)=0</code>	Set the controller in tracking or non-tracking mode.
virtual bool	<code>getTrackingMode(bool *f)=0</code>	Get the current controller mode.
virtual bool	<code>getPose(yarp::sig::Vector &x, yarp::sig::Vector &o)=0</code>	Get the current pose of the end-effector.
virtual bool	<code>getPose(const int axis, yarp::sig::Vector &x, yarp::sig::Vector &o)=0</code>	Get the current pose of the specified link belonging to the kinematic chain.
virtual bool	<code>goToPose(const yarp::sig::Vector &xd, const yarp::sig::Vector &od, const double t=0.0)=0</code>	Move the end-effector to a specified pose (position and orientation) in cartesian space.
virtual bool	<code>goToPosition(const yarp::sig::Vector &xd, const double t=0.0)=0</code>	Move the end-effector to a specified position in cartesian space, ignore the orientation.
virtual bool	<code>goToPoseSync(const yarp::sig::Vector &xd, const yarp::sig::Vector &od, const double t=0.0)=0</code>	Move the end-effector to a specified pose (position and orientation) in cartesian space.
virtual bool	<code>goToPositionSync(const yarp::sig::Vector &xd, const double t=0.0)=0</code>	Move the end-effector to a specified position in cartesian space, ignore the orientation.
virtual bool	<code>getDesired(yarp::sig::Vector &xdhat, yarp::sig::Vector &odhat, yarp::sig::Vector &qdhat)=0</code>	Get the actual desired pose and joints configuration as result of kinematic inversion.
virtual bool	<code>askForPose(const yarp::sig::Vector &xd, const yarp::sig::Vector &od, yarp::sig::Vector &xdhat, yarp::sig::Vector &odhat, yarp::sig::Vector &qdhat)=0</code>	Ask for inverting a given pose without actually moving there.
virtual bool	<code>askForPose(const yarp::sig::Vector &q0, const yarp::sig::Vector &xd, const yarp::sig::Vector &od, yarp::sig::Vector &xdhat, yarp::sig::Vector &odhat, yarp::sig::Vector &qdhat)=0</code>	Ask for inverting a given pose without actually moving there.
virtual bool	<code>askForPosition(const yarp::sig::Vector &xd, yarp::sig::Vector &xdhat, yarp::sig::Vector &odhat, yarp::sig::Vector &qdhat)=0</code>	Ask for inverting a given position without actually moving there.

Doxygen Documentation



Interfaces Tutorials





Cartesian Interface (1/9)

OPENING THE CARTESIAN INTERFACE

```
#include <yarp/dev/all.h>
Property option;

option.put("device", "cartesiancontrollerclient");
option.put("remote", "/icub/cartesianController/right_arm");
option.put("local", "/client/right_arm");

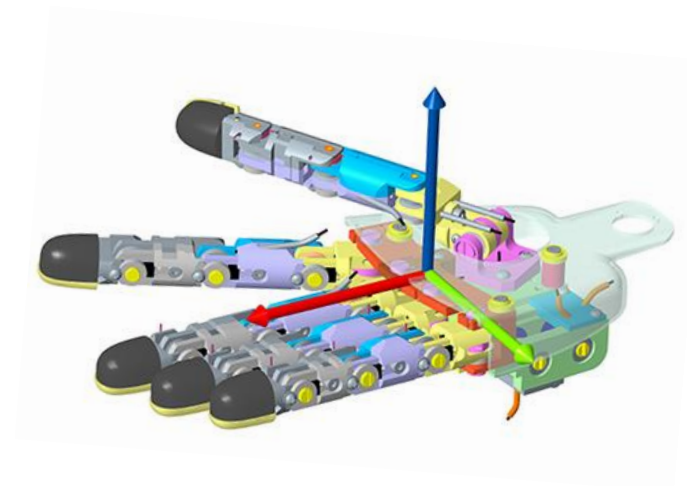
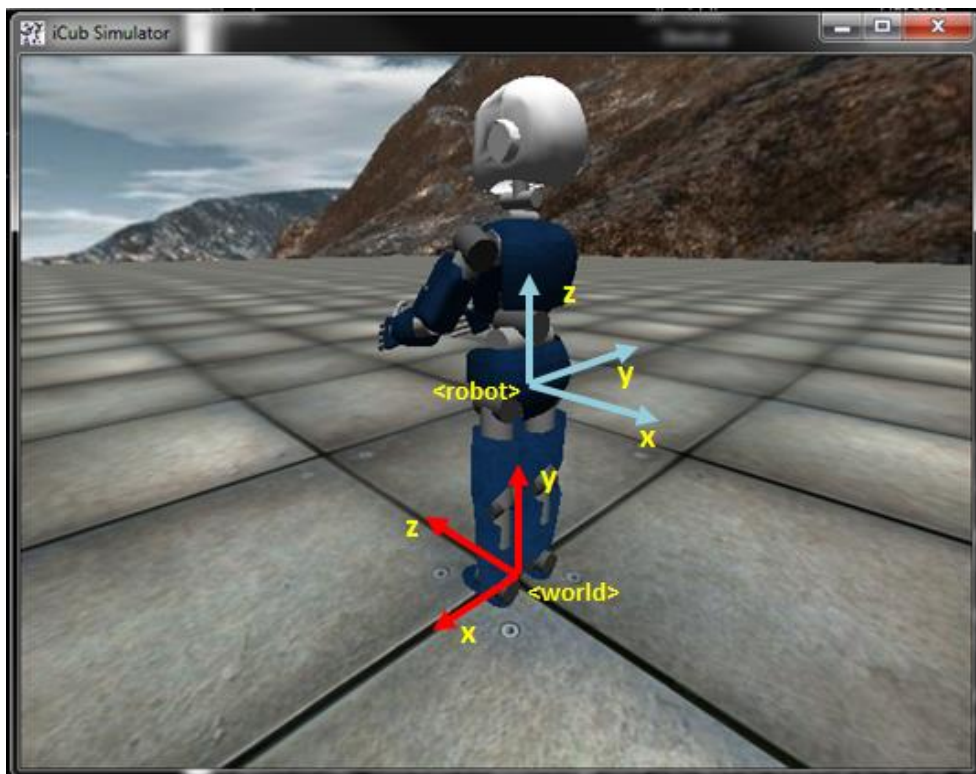
PolyDriver clientCartCtrl(option);

ICartesianControl *icart=NULL;
if (clientCartCtrl.isValid()) {
    clientCartCtrl.view(icart);
}
```




Cartesian Interface (2/9)

Coordinate Systems



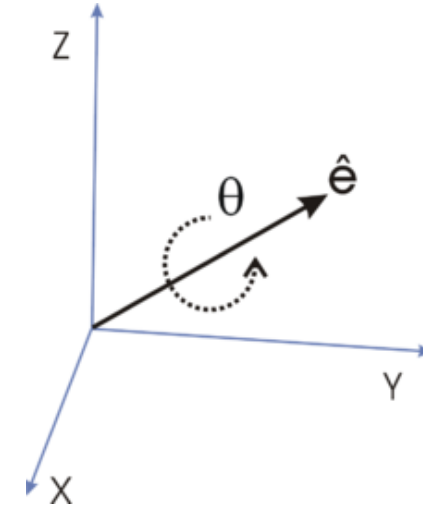


Cartesian Interface (3/9)

Orientation: Axis-Angle

$$r = \begin{bmatrix} e_x & e_y & e_z & \theta \end{bmatrix}$$

$\underbrace{\hspace{1.5cm}}_{\|e\| = 1} \quad \underbrace{\hspace{1.5cm}}_{\text{rad}}$



TARGET ORIENTATION through DIRECTION COSINE MATRIX

```
Matrix R(3,3);  
// pose x-axis  y-axis  z-axis  
R(0,0)= 0.0; R(0,1)= 1.0; R(0,2)= 0.0; // x-coordinate  
R(1,0)= 0.0; R(1,1)= 0.0; R(1,2)=-1.0; // y-coordinate  
R(2,0)=-1.0; R(2,1)= 0.0; R(2,2)= 0.0; // z-coordinate  
  
Vector o=ctrl::dcm2axis(R);
```



Cartesian Interface (4/9)

RETRIEVE CURRENT POSE

```
Vector x,o;  
icart->getPose(x,o);
```

REACH FOR A TARGET POSE (SEND-AND-FORGET)

```
icart->goToPose(xd,od);  
icart->goToPosition(xd);
```

REACH FOR A TARGET POSE (WAIT-FOR-REPLY)

```
icart->goToPoseSync(xd,od);  
icart->goToPositionSync(xd);
```

REACH AND WAIT

```
icart->goToPoseSync(xd,od);  
icart->waitMotionDone();
```



Cartesian Interface (5/9)

ASK FOR A POSE (without moving)

```
Vector xdhat,odhat,qdhat;  
icart->askForPose(xd,xdhat,odhat,qdhat);
```

MOVE FASTER/SLOWER

```
icart->setTrajTime(1.5); // point-to-point trajectory time
```

REACH WITH GIVEN PRECISION

```
icart->setInTargetTol(0.001);
```

KEEP THE POSE ONCE DONE

```
icart->setTrackingMode(true);
```



Cartesian Interface (6/9)

ENABLE/DISABLE DOF

```
Vector curDof;  
icart->getDOF(curDof); // [0 0 0 1 1 1 1 1 1 1]  
  
Vector newDof(3);  
newDof[0]=1; // torso pitch: 1 => enable  
newDof[1]=2; // torso roll: 2 => skip  
newDof[2]=1; // torso yaw: 1 => enable  
icart->setDOF(newDof, curDof);
```

GIVE PRIORITY TO REACHING IN POSITION/ORIENTATION

```
icart->setPosePriority("position"); // default  
icart->setPosePriority("orientation");
```



Cartesian Interface (7/9)

CONTEXT SWITCH

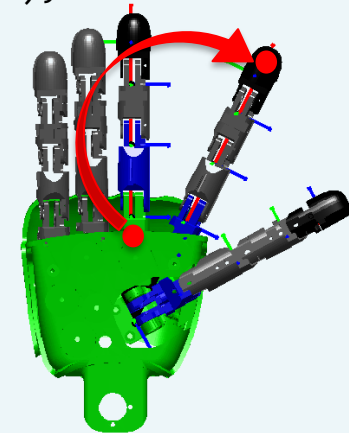
```
icart->setDOF(newDof1,curDof1);    // prepare the context  
icart->setTrackingMode(true);  
  
int context_0;  
icart->storeContext(&context_0);  // latch the context  
  
icart->setDOF(newDof2,curDof2);    // perform some actions  
icart->goToPose(x,o);  
  
icart->restoreContext(context_0);  // retrieve context_0  
icart->goToPose(x,o);             // perform with context_0
```



Cartesian Interface (8/9)

DEFINING A DIFFERENT EFFECTOR

```
iCubFinger finger("right_index");  
Vector encs; iencs->getEncoders(encs.data());  
Vector joints; finger.getChainJoints(encs,joints);  
Matrix tipFrame=finger.getH((M_PI/180.0)*joints);  
  
Vector tip_x=tipFrame.getCol(3);  
Vector tip_o=ctrl::dcm2axis(tipFrame);  
  
icart->attachTipFrame(tip_x,tip_o);  
  
icart->getPose(x,o);  
icart->goToPose(xd,od);  
  
icart->removeTipFrame();
```





Cartesian Interface (9/9)

Find out more (e.g. **Events Callbacks ...**):

http://wiki.icub.org/iCub/main/dox/html/icub_cartesian_interface.html

USING THE INTERFACE ALONG WITH THE SIMULATOR

```
1> iCub_SIM
2> yarprobotinterface --context simCartesianControl
3> iKinCartesianSolver --context simCartesianControl --part left_arm
```

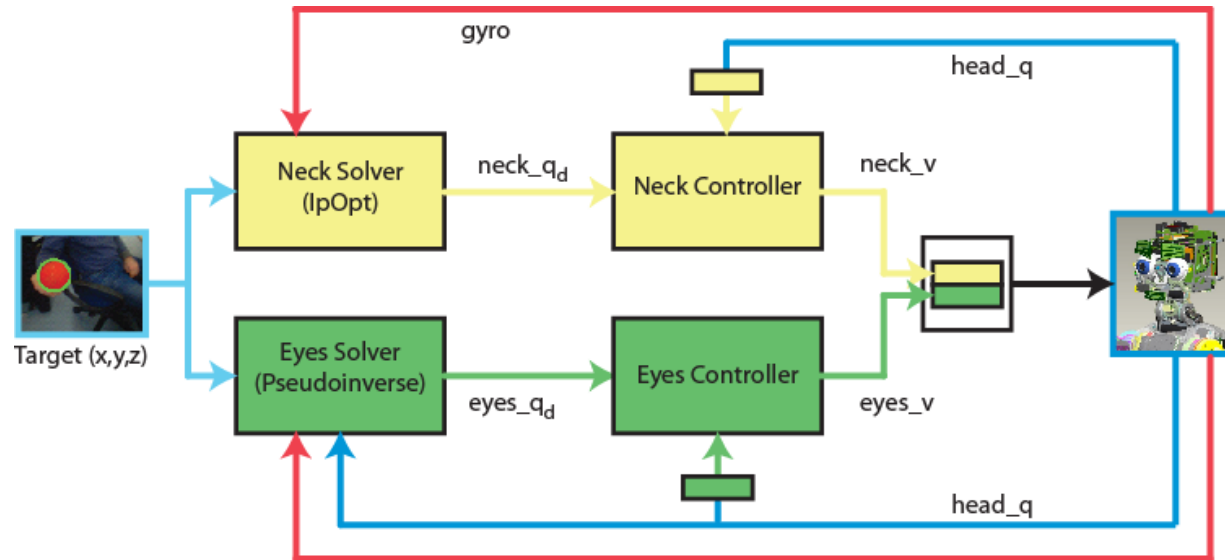
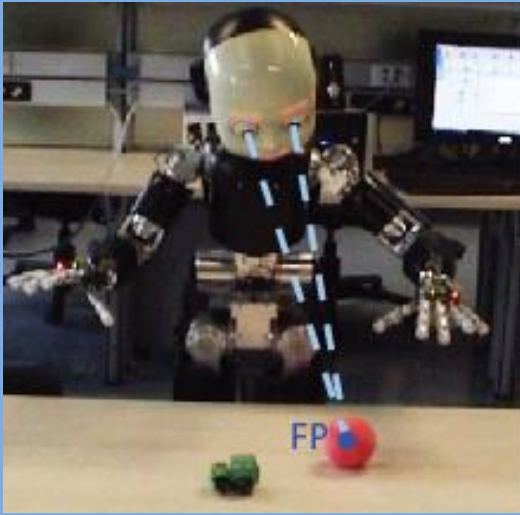
```
option.put("device","cartesiancontrollerclient");
option.put("remote","/iCubSim/cartesianController/left_arm");
option.put("local","/client/right_arm");
```




Tutorial Time !



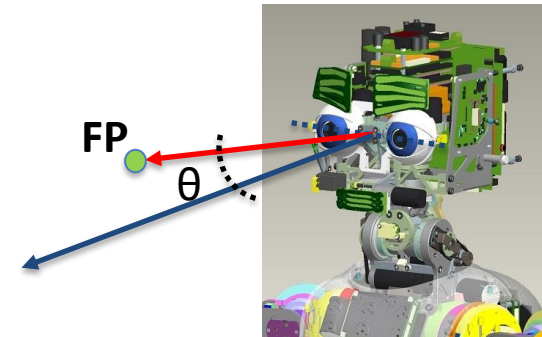
The Gaze Controller (1/6)



Yet another Cartesian Controller: reuse ideas ...

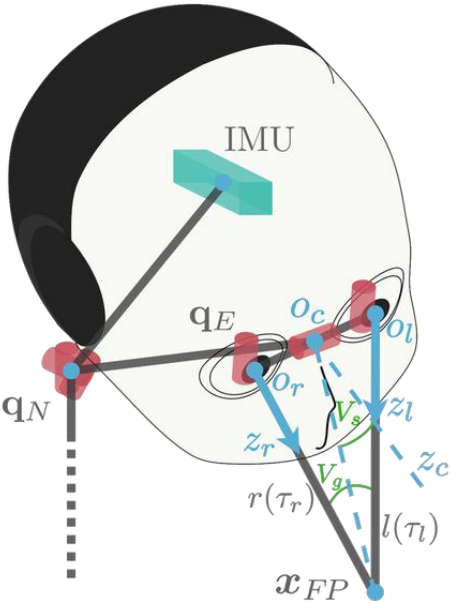
Then, apply easy transformations from Cartesian to ...

1. Egocentric angular space
2. Image planes (mono and stereo)





The Gaze Controller (2/6)

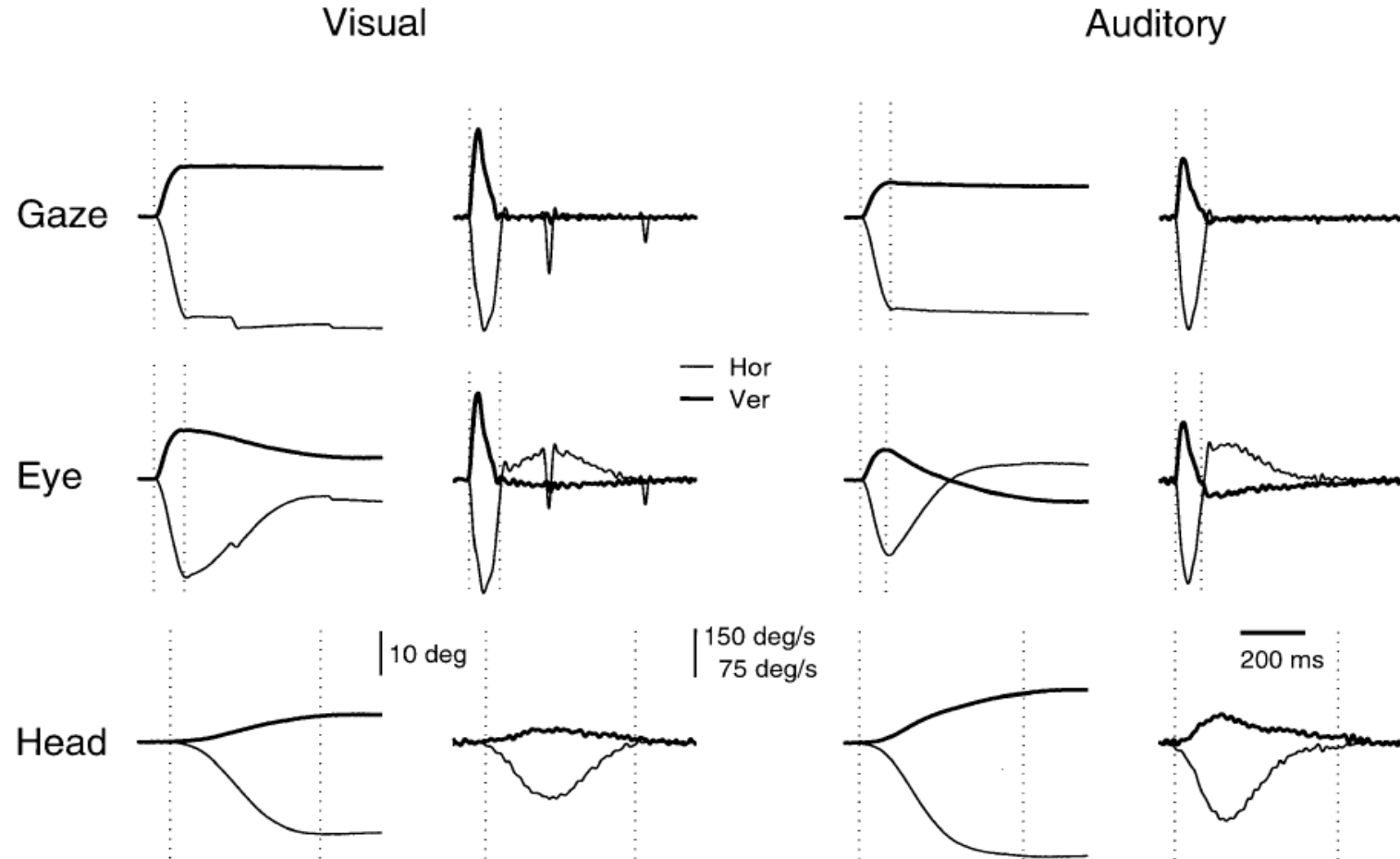


Joint #	Part	Joint Name	Range	Unit
0	Neck	Pitch	+/-	[deg]
1	Neck	Roll	+/-	[deg]
2	Neck	Yaw	+/-	[deg]
3	Eyes	Tilt	+/-	[deg]
4	Eyes	Version	+/-	[deg]
5	Eyes	Vergence	≥ 0	[deg]



The Gaze Controller (4/6)

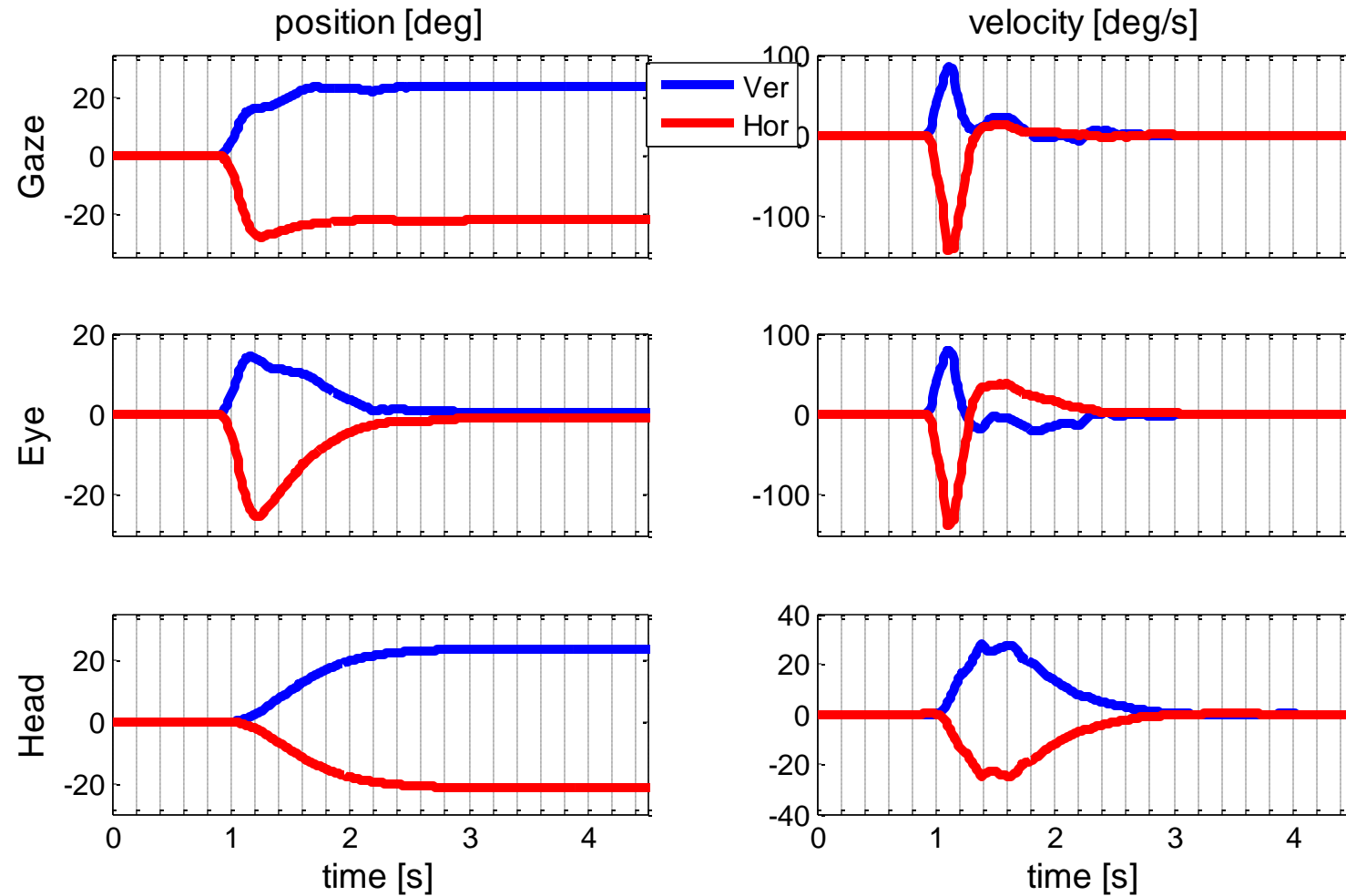
Studies on humans ...





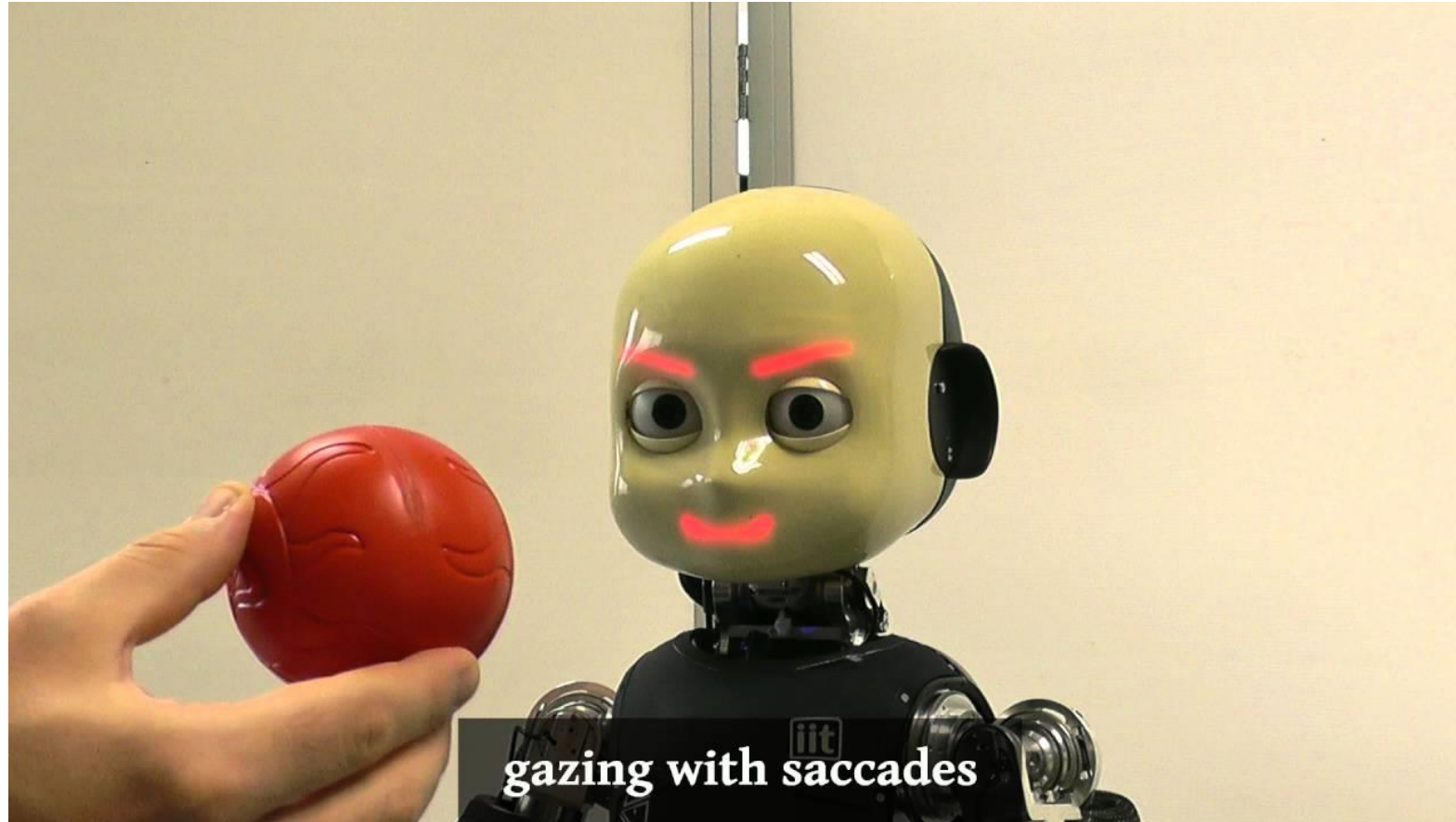
The Gaze Controller (5/6)

Results on iCub ...





The Gaze Controller (6/6)



<http://y2u.be/I4ZKfAvs1y0>



Gaze Interface (1/7)

OPENING THE GAZE INTERFACE

```
#include <yarp/dev/all.h>
Property option;

option.put("device", "gazecontrollerclient");
option.put("remote", "/iKinGazeCtrl");
option.put("local", "/client/gaze");

PolyDriver clientGazeCtrl(option);

IGazeControl *igaze=NULL;
if (clientGazeCtrl.isValid()) {
    clientGazeCtrl.view(igaze);
}
```




Gaze Interface (2/7)

GET CURRENT FIXATION POINT IN CARTESIAN DOMAIN

```
Vector x;  
igaze->getFixationPoint(x);
```

GET CURRENT FIXATION POINT IN ANGULAR DOMAIN

```
Vector ang;  
igaze->getAngles(ang);  
// ang[0] => azimuth [deg]  
// ang[1] => elevation [deg]  
// ang[2] => vergence [deg]
```

LOOK AT 3D POINT

```
igaze->lookAtFixationPoint(xd);
```

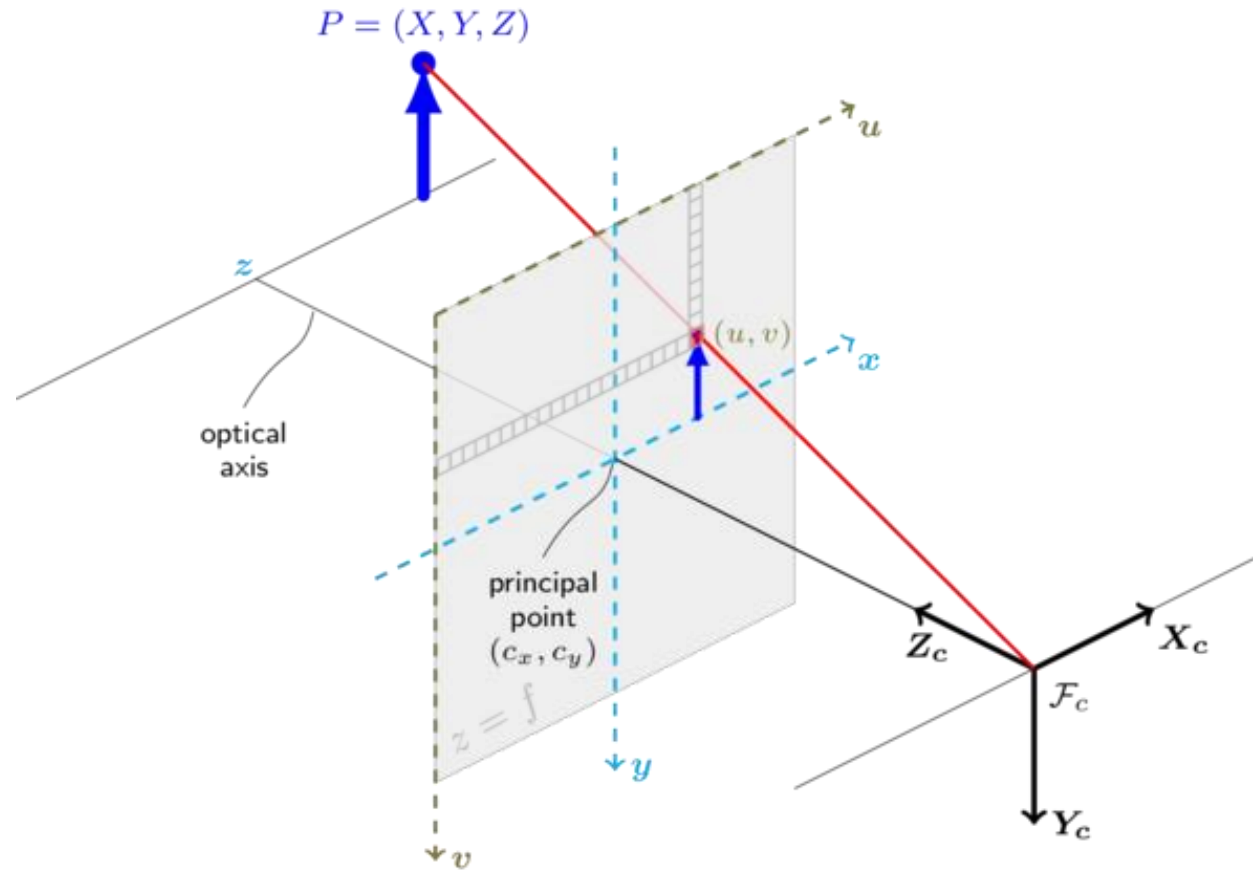
... IN ANGULAR DOMAIN

```
igaze->lookAtAbsAngles(ang);  
igaze->lookAtRelAngles(ang);
```



Gaze Interface (3/7)

LOOK AT POINT IN IMAGE DOMAIN





Gaze Interface (4/7)

LOOK AT POINT IN IMAGE DOMAIN

```
int camSel=0; // 0 => left, 1 => right
Vector px(2);
px[0]=100;
px[1]=50;
double z=1.0;

igaze->lookAtMonoPixel(camSel,px,z);
```



... EQUIVALENT TO

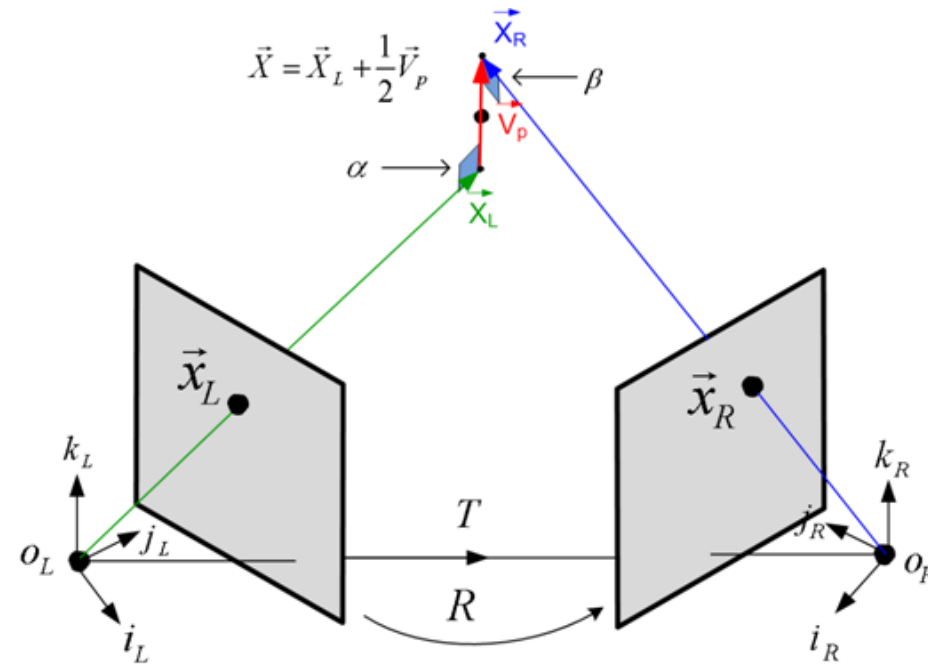
```
Vector x;
igaze->get3DPoint(camSel,px,z,x);
igaze->lookAtFixationPoint(x);
```



Gaze Interface (5/7)

GEOMETRY OF PIXELS

```
Vector x;  
igaze->get3DPointOnPlane(camSel,px,plane,x);  
igaze->get3DPointFromAngles(mode,ang,x);  
igaze->triangulate3DPoint(px1,pxr,x);
```





Gaze Interface (6/7)

GEOMETRY OF PIXELS

```
Vector x;  
igaze->get3DPointOnPlane(camSel,px,plane,x);  
igaze->get3DPointFromAngles(mode,ang,x);  
igaze->triangulate3DPoint(px1,pxr,x);
```





Gaze Interface (7/7)

Find out more (e.g. **Events Callbacks**, **Fast Saccadic Mode ...**):

http://wiki.icub.org/iCub/main/dox/html/icub_gaze_interface.html

USING THE INTERFACE ALONG WITH THE SIMULATOR

```
1> iCub_SIM
2> iKinGazeCtrl --from configSim.ini

option.put("device","gazecontrollerclient");
option.put("remote","/iKinGazeCtrl");
option.put("local","/client/right_arm");
```



Tutorial Time !