Object Localization and Reaching on iCub

# Task Description

1) Creation of a blue ball in the iCub\_SIM simulator  
2) Acquisition of the image stream from the eyes  
3) Color-based retrieval and ball centroid computation  
4) Computation of the 3D position via triangulation  
5) Motion of the robot gaze for centering the target object  
7) Reaching of the object for grasping by either of the iCub robotic arm

D:\Documenti\Universita\Dottorato\Ricerca\Software\Assignments\Assignment 2 - iCub tutorials\BallReaching\Documentation\DataFlowDiagram.emf

Figure 1: Data Flow Diagram of the reaching application

# Instructions to run the program on iCubSim

**1- Launch the Cartesian Interface**

In order to use the Cartesian Interface, make sure that the following steps are done. [Note that the term cluster refers to the set of computers directly connected to the robot network, whereas PC104 indicates the hub board mounted on the robot]

**1.1** Update YARP and iCub repositories.

**1.2** Compile YARP (always a good practice).

**1.3** Install Ipopt on the cluster.

**1.4** On the cluster: compile the repository with the switch ***ENABLE\_icubmod\_cartesiancontrollerclient enabled***. This will make the client part of the interface available.

**1.5** On PC104: compile the repository with the switch ENABLE\_icubmod\_cartesiancontrollerserver enabled. This will make the server part of the interface available on the hub.

Henceforth rely on the installed copy of ***$ICUB\_ROOT/main/app/iCubStartup/scripts/iCubStartup.xml.template***  
application to launch both the cartesian solvers (and other useful tools as well).

Make sure that the machines where the solvers will be running are configured properly to work with the specific robot (e.g. the YARP\_ROBOT\_NAME correctly points to the name of the robot) in order to load at start-up the robot dependent kinematics.

**NOTE**: Once everything is launched from within the application, if the user accidentally stops one of these modules, it turns to be mandatory to restart all of them from the beginning; this comes from the requirement to keep the communication protocol as light as possible, avoiding requests for integrity check and reinitialization.

Then:

**a.** Launch the iCub Simulator:  
***iCub\_SIM***  
  
**b.** Launch the simCartesianControl module.

Parameters:

--robot name // specifies the simulated robot name to connect to.  
--no\_legs // disable the control of the robot legs.

Other options are available but their default values should be fine for normal use. If you are really curious then get into the short code :)  
  
e.g. in the most basic case on the simulator:  
***simCartesianControl --robot icubSim --no\_legs***

**c.** Launch the Cartesian Solvers for the required limbs: have a look to the template located in the directory $ICUB\_ROOT/main/app/simCartesianControl/scripts .  
  
e.g. for left arm:   
***iKinCartesianSolver --context simCartesianControl --part left\_arm***

**2- Launch the gaze interface**, selecting the proper configuration file:  
***iKinGazeCtrl --from configSim.ini***

**3- Launch 2 "findBallLocation" modules** with "--camSide left" and "camSide --right", respectively  
***findBallLocation --camSide left  
findBallLocation --camSide right***

**4- Connect** iCubSim eye image output stream port “/icubSim/cam/${camSide}” with "/findBallLocation/image/${camSide}/in", for both sides.  
***yarp connect /icubSim/cam/left /findBallLocation/image/left/in   
yarp connect /icubSim/cam/right /findBallLocation/image/right/in***

**5- Launch "compute3DLookReach**"  
***compute3DLookReach***

**6- Connect** "/findBallLocation/centroid/${camSide}/out" with "/compute3DLookReach/centroid/${camSide}/in", for both sides:  
***yarp connect /findBallLocation/centroid/left/out /compute3DLookReach/centroid/left/in  
yarp connect /findBallLocation/centroid/right/out /compute3DLookReach/centroid/right/in***