```
// main file for atlas control moving right hand to HMI selected 3-D point: wsn  March 6, 2013
// theory of operation:
// callback fnc HMI_callback() subscribes to topic "/userPickedPoint"
// HMI_callback() copies the incoming PointStamped to a global variable, and sets the flag
"HmiPointIsGood"
// This point is interpreted as the desired origin of frame "/right_f1_base" on the palm of the right
hand
// The right-arm Jacobian is computed and dq is computed based on J'*dp, where dp is the hand error
(in pelvis frame)
// Jacobian and dq are recomputed iteratively.  Should result in hand converging on goal

#include "AtlasJointControl.h"
#include "JacobianComputer.h"
#include <geometry_msgs/PointStamped.h> //datatype of published desired hand position from HMI


geometry_msgs::PointStamped g_PickedPoint; //global var to get values from HMI callback
osrf_msgs::JointCommands g_jntCmd;
double jntCmdsFromAtlas[NJoints];

bool HmiPointIsGood = false;

//callback to get hand position commands from HMI
void HMI_callback(const geometry_msgs::PointStamped& pickedPoint) {
  ROS_INFO("got point: x,y,z= %f %f %f",pickedPoint.point.x,pickedPoint.point.y,pickedPoint.point.z);
  g_PickedPoint=pickedPoint;
  HmiPointIsGood=true; //notify "main" that it is OK to use g_PickedPoint
}

//callback to see what was commanded to the robot; odd behavior--does not seem to recognize
// commands since start-up, e.g. neck bend
void atlas_cmds_callback(const osrf_msgs::JointCommands& jntCmds) {
  g_jntCmd= jntCmds; // make info globally available
  for (unsigned int i=0;i<NJoints;i++)
     jntCmdsFromAtlas[i]=jntCmds.position[i];
}

//helper fnc to find max abs val of entries in a vector
double findMaxAbs(double xvec[],int Npts) {
 double maxval;
 maxval=fabs(xvec[0]);
 for (int i=1;i<Npts;i++)
  if (fabs(xvec[i])>maxval)
    maxval=fabs(xvec[i]);
 return (maxval);
}

int main(int argc, char** argv)
{
  double looprateHz = 10.0;
  tf::Vector3 rHandOriginDesired;
  tf::Vector3 rHandOriginActual;
  tf::Vector3 rHandPrev;
  tf::Vector3 drHand;
  tf::Vector3 JdqRHand;
  tf::Vector3 rHandErrorVec;
  double jointSpaceCommand[NJoints];
  double dqVec[NJoints];
  tf::StampedTransform transform;
  double prevAngles[NJoints];
  double currentAngles[NJoints];
  double dq[NJoints];
  geometry_msgs::PointStamped tfPickedPoint;  // HMI point transformed to pelvis frame

  // ROS set-ups:
  ros::init(argc, argv, "hand_follow_hmi"); // call this node "hand_follow_hmi"

  // pointer to node handle, for use by drcsim demo code, now moved to AtlasJointControl constructor
  ros::NodeHandle* rosnode = new ros::NodeHandle(); //note--need to request a nodehandle before
                                                    // ros::Time works (not sure why)
  ros::NodeHandle nh_subCB;                         //
  ros::NodeHandle nh_subHMI;
```

```
  ros::NodeHandle nh_subCmds;

  tf::TransformListener tf_listener;  // create a transform listener

  while (!ros::Time::isValid()) {} // make sure we are receiving valid clock values

  ros::Rate looprate(looprateHz); //will perform sleeps to enforce loop rate of "looprateHz" Hz
  ros::Time startTime= ros::Time::now(); // get the current time, which defines our start
                                         // not really needed, but may come in handy

  //create an instance of an AtlasJointControl...  See AtlasJointControl.cpp
  AtlasJointControl atlasJointControl(rosnode,looprateHz);

  // if want to use method member of class as callback func, can do it this way:
  ROS_INFO("setting up subscriber to joint states");
  ros::Subscriber sub = nh_subCB.subscribe("/atlas/joint_states", 1,
                        &AtlasJointControl::getJointStatesCB, &atlasJointControl);

  // set up subscriber to HMI input:
  ros::Subscriber sub_HMI = nh_subHMI.subscribe("/userPickedPoint", 1,HMI_callback);

  //subscriber to atlas commands:
  ros::Subscriber sub_cmds = nh_subCmds.subscribe("/atlas/joint_commands", 1,atlas_cmds_callback);


  // wait for valid data from joint commands callback--see what is currently commanded to the robot
joints
  jntCmdsFromAtlas[0]=-1000.0; //init w/ impossible value; cycle until value is replaced
  while (jntCmdsFromAtlas[0]<-500.0) {
    ros::spinOnce();
  }

  ROS_INFO("current joint angle commands: ");
  atlasJointControl.displayAngles(jntCmdsFromAtlas);

  //look at the actual joint angles at this moment:
  //test for bad initial value in sensed angles--put there via constructor of atlasJointControl; cycle
until value is replaced
  atlasJointControl.getCurrentAngles(currentAngles);
  while (currentAngles[0]<-500.0) {
    ros::spinOnce();
    atlasJointControl.getCurrentAngles(currentAngles);
  }
  ROS_INFO("current angles:");  //debug: display the received joint angles; should all be ~0.0
                                // for Atlas in start-up pose
  atlasJointControl.displayCurrentAngles();

  JacobianComputer jacobianComputer; // create a Jacobian computer object; might bury this inside
another object...

  //where is the hand now?  (w/rt pelvis frame)
  bool tf_not_ready=true;
  int ntries=0;
  while(tf_not_ready) {
    try {
          tf_not_ready=false;
          tf_listener.lookupTransform("/pelvis", "/right_f1_base", ros::Time(0), transform);
    }
    catch (tf::TransformException ex) { //do nothing
      tf_not_ready=true;
      ntries++;
      ROS_INFO("waiting for right_f1_base frame; ntries = %d",ntries);
      ros::Duration(0.3).sleep();
    }
  }

  rHandOriginActual= transform.getOrigin(); // extract the origin of r_hand frame, relative to pelvis
  ROS_INFO("initial hand pos = %f %f %f",rHandOriginActual[0],rHandOriginActual[1],rHandOriginActual
[2]);

  // for smooth start-up, set hand goal coincident with current, actual hand position
  tfPickedPoint.point.x= rHandOriginActual[0];  //init right-hand goal w/rt pelvis
```

```
  tfPickedPoint.point.y= rHandOriginActual[1];
  tfPickedPoint.point.z= rHandOriginActual[2];

  double goalAngles[NJoints];
  double moveDuration; //can specify speed of move via this

  //to start, command robot to go to its current pose...should stand still
  atlasJointControl.getCurrentAngles(currentAngles);

  //may initialize move command to current angles:
  //better is to use current commands...having trouble with neck command snapping back to zero
  //atlasJointControl.setNewJointSpaceGoal(jntCmdsFromAtlas,jntCmdsFromAtlas, 1.0);// move time=1Sec
  atlasJointControl.setNewJointSpaceGoal(currentAngles,currentAngles, 1.0);// move time=1Sec

  // define a right-hand goal position: initially, same as current hand position
  rHandOriginDesired[0]= tfPickedPoint.point.x;
  rHandOriginDesired[1]= tfPickedPoint.point.y;
  rHandOriginDesired[2]= tfPickedPoint.point.z;


  //ROS_ERROR("false alarm");

  ROS_INFO("starting main loop...");
  while(ros::ok())  // main loop
  {
   double qmax = 0.001;
   if(atlasJointControl.isMoveDone()) { // test if time to set a new move
       // get the current hand position:
       tf_listener.lookupTransform("/pelvis", "/right_f1_base", ros::Time(0), transform);
       rHandOriginActual= transform.getOrigin(); // extract the origin of r_hand frame, relative to
pelvis

       ROS_INFO("hand pos = %f %f %f",rHandOriginActual[0],rHandOriginActual[1],rHandOriginActual[2]);

       // get desired hand position from HMI...transform this point to the pelvis frame:
       // input PickedPoint (updated by callback) and transform to tfPickedPoint
       if (HmiPointIsGood) {
        try{
         tf_listener.transformPoint("/pelvis", g_PickedPoint, tfPickedPoint); //tfPickedPoint is goal
in pelvis frame
         //ROS_INFO("main: HMI orig = %f %f %
f",g_PickedPoint.point.x,g_PickedPoint.point.y,g_PickedPoint.point.z);
         ROS_INFO(" ");
         ROS_INFO("desired hand pose...");
         ROS_INFO("main: HMI xfmd = %f %f %
f",tfPickedPoint.point.x,tfPickedPoint.point.y,tfPickedPoint.point.z);
         rHandOriginDesired[0]= tfPickedPoint.point.x;
         rHandOriginDesired[1]= tfPickedPoint.point.y;
         rHandOriginDesired[2]= tfPickedPoint.point.z;
          }
        catch(tf::TransformException ex)
         {
           // transform will fail if point's timestamp is too old
           ROS_ERROR("HMI catch: failed to do point transform");
         }
         HmiPointIsGood=false; //only transform new points
       }
       else
         ROS_INFO("no new HMI point");

       rHandErrorVec= rHandOriginDesired-rHandOriginActual;
       ROS_INFO("hand err = %f %f %f",rHandErrorVec[0],rHandErrorVec[1],rHandErrorVec[2]);


       jacobianComputer.computeRightHandJacobians(tf_listener); //  compute a right-hand Jacobian

       //compute dqVec for dp step towards goal as dqVec= J'dp
       jacobianComputer.computeRightHandJTransposeDp(rHandErrorVec,dqVec);

       // make sure dqVec is not too large: find max value
       qmax = findMaxAbs(dqVec,NJoints);
       if (qmax>0.05)
```

```
        for (unsigned int i=0;i<NJoints;i++)
          dqVec[i]*=(0.05/qmax); // scale entire vec so max increment does not exceed 0.05 rad in move
duration


       ROS_INFO("J'dp[22-25]= %f %f %f %f",dqVec[22],dqVec[23],dqVec[24],dqVec[25]);


       atlasJointControl.getCurrentGoalJointAngles(jointSpaceCommand); //look up previous joint-space
cmd vec

       for (unsigned int i=22;i<26;i++) //only increment right-arm angles (not back joints)
         jointSpaceCommand[i]+=dqVec[i];  //increment the previous joint-space command by vector dq

       ROS_INFO("cmd[22-25]= %f %f %f %f",jointSpaceCommand[22],jointSpaceCommand[23],
                                          jointSpaceCommand[24],jointSpaceCommand[25]);

       moveDuration=0.3; // this is slow; could speed up w/ shorter moveDuration and/or larger max dq
per iteration
       atlasJointControl.setNewJointSpaceGoal(jointSpaceCommand, moveDuration);
     }


   // next line commands a joint-space incremental move--invoked at frequency "looprate"
   atlasJointControl.update();  //compute/publish incremental motion commands towards current goal

   looprate.sleep();
   ros::spinOnce();
  }
  return 0;
}
```