D:\mk2_ik_getting_offset.cpp

```cpp
#include <ros/ros.h>

// MoveIt!
#include <moveit/robot_model_loader/robot_model_loader.h>
#include <moveit/robot_model/robot_model.h>
#include <moveit/robot_state/robot_state.h>

#include <moveit/planning_interface/planning_interface.h>
#include <moveit/planning_scene/planning_scene.h>
#include <moveit/kinematic_constraints/utils.h>
#include <moveit_msgs/DisplayTrajectory.h>
#include <moveit_msgs/PlanningScene.h>

//#include <ros/ros.h>
#include <sensor_msgs/Joy.h>
#include <eigen_conversions/eigen_msg.h>
#include <sensor_msgs/JointState.h>
#include <geometry_msgs/PoseStamped.h>







class TeleopMK2
{
public:
    TeleopMK2();

private:
    void joyCallback(const sensor_msgs::Joy::ConstPtr& joy);
    void poseCallback(const geometry_msgs::PoseStamped box);
    robot_model::RobotModelPtr kinematic_model;
    ros::NodeHandle nh_;

    ros::Subscriber joy_sub_;
    ros::Subscriber box_pub;
    ros::Publisher joint_pub_;

    geometry_msgs::Pose actual_end_effector_;
    geometry_msgs::Pose desired_end_effector_;
    geometry_msgs::PoseStamped temp_box_pose;
    geometry_msgs::PoseStamped box_pose;
    std::vector<std::string> joint_names_global;

    bool homePose_flag;
};

TeleopMK2::TeleopMK2()
{
    joint_pub_ = nh_.advertise<sensor_msgs::JointState>("/command", 5);
    joy_sub_ = nh_.subscribe<sensor_msgs::Joy>("joy", 10, &TeleopMK2::joyCallback, this);
    box_pub = nh_.subscribe<geometry_msgs::PoseStamped>("/box_pose",10, &
```

```
TeleopMK2::poseCallback, this);

    // Load the robot model
    robot_model_loader::RobotModelLoader robot_model_loader("robot_description");

    // Get a shared pointer to the model
    kinematic_model = robot_model_loader.getModel();

    // Get and print the name of the coordinate frame in which the transforms for this model are computed

    // WORKING WITH THE KINEMATIC STATE
    // Create a kinematic state - this represents the configuration for the robot represented by kinematic_model
    robot_state::RobotStatePtr kinematic_state(new robot_state::RobotState(kinematic_model));

    // Set all joints in this state to their default values
    kinematic_state->setToDefaultValues();

    // Get the configuration for the joints in the right arm of the MK2
    const robot_state::JointModelGroup* joint_state_group = kinematic_state->getJointModelGroup("manipulator");
    //const std::vector<std::string> &joint_names1 = kinematic_state->getJointModelGroup("manipulator1")->getJointModelNames();
    //tutorial2

    // Get the names of the joints in the right_arm
    const std::vector<std::string> &joint_names = joint_state_group->getJointModelNames();
    for (unsigned int i=0; i<joint_names.size(); i++)
    {
        std::cout << joint_names.at(i) << std::endl;
        joint_names_global.push_back(joint_names.at(i));
      //  joint_names_global.push_back(joint_names1.at(i));
        //tutorial2
    }

    // Get Joint Values
    // ^^^^^^^^^^^^^^^^
    // Get the joint states for the right arm
    std::vector<double> joint_values;
    std::vector<double> initial_joint_values;
    kinematic_state->copyJointGroupPositions(joint_state_group,joint_values);
    initial_joint_values = joint_values;

    double p[] = {0.18,0.18,0.15,0.18,0.18,0.18,0.18,0.15,0.18,0.18};
     std::vector<double> vel(p, p+10);
    //double p1[] = {0.18,0.18,0.15,0.18,0.18,0.18,0.18,0.15,0.18,0.18, 0.1, 0.1, 0.1, 0.1};
    //std::vector<double> vel1(p1, p1+14);
     //tutorial2

    sensor_msgs::JointState msg;

    ros::Rate r(50);

    const Eigen::Affine3d &end_effector_state = kinematic_state->getGlobalLinkTransform("link_7");

```

```
105     Eigen::Affine3d new_end_effector_state;
106     tf::poseEigenToMsg(end_effector_state, actual_end_effector_);
107     homePose_flag = false;
108 
109 
110     while(nh_.ok())
111     {
112 
113 
114         if(homePose_flag)
115         {
116             tf::poseEigenToMsg(end_effector_state, actual_end_effector_);
117             msg.header.stamp = ros::Time::now();
118             msg.name = joint_names;
119             msg.position = initial_joint_values;
120 
121             msg.velocity = vel;
122             joint_pub_.publish(msg);
123         }
124         //finding setfromik, changing jointstategroup
125         else
126         {
127             tf::poseMsgToEigen(actual_end_effector_, new_end_effector_state);
128 
129             bool found_ik = kinematic_state->setFromIK(joint_state_group,
new_end_effector_state, 10, 0.1);
130 
131             if (found_ik) //sets up the motion
132 {
133                 kinematic_state->copyJointGroupPositions(joint_state_group,joint_values);
134 
135                 msg.header.stamp = ros::Time::now();
136                 msg.name = joint_names;
137                 msg.position = joint_values;
138                 msg.velocity = vel;
139                 joint_pub_.publish(msg);
140 
141 
142 
143                 std::cout<<"\n Each joint target: \n";
144                 std::cout<<"joint: ";
145                 std::copy(joint_values.begin(), joint_values.end(), std::ostream_iterator<
double>(std::cout, "\n joint: "));
146                 std::cout<<"\n";
147             }
148             else
149             {
150                 ROS_INFO("Did not find IK solution");
151             }
152 
153         }
154         std::cout<<"Hi I`m Min \n";
155         float ad_x = 0.295;
156         //Y&Z is opposite direction
157         float ad_y = -0.249;
158         float ad_z = -0.5;
159         //float joint_8 = -0.5;
```

```
160
161
162
163
164
165         homePose_flag = false;
166         std::cin >> ad_x;
167         std::cin >> ad_y;
168         std::cin >> ad_z;
169
170
171         std::cout << " final point: "<<ad_x<<","<<ad_y<<","<<ad_z<<"\n";
172
173         if(ad_x== 0)
174         {
175             homePose_flag = true;
176             ROS_INFO("If you entered z < 0.28, plz enter z > 0.28");
177
178             /*kinematic_state->copyJointGroupPositions(joint_state_group,joint_values);
179                                 joint_values.push_back(0.2);  // index
180                                 joint_values.push_back(0.2); //ring yaw
181                                 joint_values.push_back(-0.3);//angle thumb
182                                 joint_values.push_back(0.2); //thumb
183
184                                 msg.header.stamp = ros::Time::now();
185                                 msg.name = joint_names1;
186
187                                 msg.position = joint_values;
188                                 msg.velocity = vel1;
189                                 msg.header.stamp = ros::Time::now();
190                                 msg.name = joint_names1;
191                                 msg.name.push_back("right_thumb_roll_joint");
192                                 msg.name.push_back("right_thumb_pitch_joint");
193                                 msg.name.push_back("right_index_yaw_joint");
194                                 msg.name.push_back("right_ring_yaw_joint");
195                                 joint_pub_.publish(msg);
196           }
197       else
198         {  homePose_flag = false;
199
200                 kinematic_state->copyJointGroupPositions(joint_state_group,joint_values);
201
202                 joint_values.push_back(1.5);  // index
203                 joint_values.push_back(1.5); //ring yaw
204                 joint_values.push_back(-1.5);//angle thumb
205                 joint_values.push_back(1.2); //thumb
206
207                 msg.header.stamp = ros::Time::now();
208                 msg.name = joint_names1;
209
210                 msg.position = joint_values;
211                 msg.velocity = vel1;
212                 msg.header.stamp = ros::Time::now();
213                 msg.name = joint_names1;
214                 msg.name.push_back("right_thumb_roll_joint");
215                 msg.name.push_back("right_thumb_pitch_joint");
```

```
                msg.name.push_back("right_index_yaw_joint");
                msg.name.push_back("right_ring_yaw_joint");
  joint_pub_.publish(msg);
          }
          */

     /*if(ad_x== 0.5)
          {
              homePose_flag = false;
              //ROS_INFO("If you entered z < 0.28, plz enter z > 0.28");
              kinematic_state->copyJointGroupPositions(joint_state_group,joint_values);
                                  joint_values.push_back(0.2);  // index
                                  joint_values.push_back(0.2); //ring yaw
                                  joint_values.push_back(-0.3);//angle thumb
                                  joint_values.push_back(0.2); //thumb

                                  msg.header.stamp = ros::Time::now();
                                  msg.name = joint_names1;

                                  msg.position = joint_values;
                                  msg.velocity = vel1;
                                  msg.header.stamp = ros::Time::now();
                                  msg.name = joint_names1;
                                  msg.name.push_back("right_thumb_roll_joint");
                                  msg.name.push_back("right_thumb_pitch_joint");
                                  msg.name.push_back("right_index_yaw_joint");
                                  msg.name.push_back("right_ring_yaw_joint");
                                  joint_pub_.publish(msg);
          }


 */
              //tutorial2



          ad_x += (-0.2);
          ad_y += (0);
          ad_z += (0);
          //'joint_6' += (0);
          //change Y&Z aixs direction
          ad_y = (-ad_y);
          ad_z = (-ad_z);
          //'joint_6' = (-'joint_6');
          //std::cout<< x_ad;


          actual_end_effector_.position.x =ad_x;
          actual_end_effector_.position.y =ad_y;
          actual_end_effector_.position.z =ad_z;

          actual_end_effector_.orientation.x = 0.0;
          actual_end_effector_.orientation.y = -0.7071068;
          actual_end_effector_.orientation.z = 0.0;
          actual_end_effector_.orientation.w = 0.707;

```

```
272
273
274
275
276
277
278         ros::spinOnce();
279         r.sleep();
280 }
281     }
282
283
284     void TeleopMK2::poseCallback(const geometry_msgs::PoseStamped box)
285     {
286         temp_box_pose = box;
287     }
288
289     void TeleopMK2::joyCallback(const sensor_msgs::Joy::ConstPtr& joy)
290     {
291
292
293
294         if (joy->buttons[7]) //     button 8 : Go home position
295         {
296             homePose_flag = true;
297         }
298         else if(joy->axes[0] != 0 || joy->axes[1] != 0 || joy->axes[3] != 0)
299         {
300             homePose_flag = false;
301
302             actual_end_effector_.position.x += -0.005 * joy->axes[1];
303             actual_end_effector_.position.y += 0.005 * joy->axes[0];
304             actual_end_effector_.position.z += -0.005 * joy->axes[3];
305
306             //actual_end_effector_.position.'joint_6' += -0.005 * joy->axes[5];
307         }
308
309         ROS_INFO_STREAM ("Pos:"<<actual_end_effector_.position.x <<"," <<
    actual_end_effector_.position.y <<","
310                          <<actual_end_effector_.position.z <<"; Ori:"<<
    actual_end_effector_.orientation.x <<","
311                          <<actual_end_effector_.orientation.y <<"," <<
    actual_end_effector_.orientation.z <<","
312                          <<actual_end_effector_.orientation.w);
313     }
314
315     int main(int argc, char** argv)
316     {
317         ros::init(argc, argv, "teleop_mk2");
318         TeleopMK2 teleop_mk2;
319     }
320
321
```