Introduction to Cognitive Robotics

Module 4: Robot Manipulators

Lecture 5: Implementation of the pick-and-place example for a Lynxmotion AL5D robot arm using the Frame class in C++

www.cognitiverobotics.net

A Simple Pick-and-Place Task Specification

$M0$: Move out of the field of view of the camera

Determine the pose of a block and a suitable grasp point
(possibly using a camera)

$M1$: Move to an approach position above the grasp point

$M2$: Move to the grasp position

Grasp the block

$M3$: Move to the depart position above the grasp point

$M4$: Move to the approach position in above the destination position

$M5$: Move to the destination position

Release the block

$M6$: Move to the depart position above the block

A Simple Robot Programming Language

- This task-level approach to robot programming is typical of many commercial manipulators and they typically provide their own frame-based programming language

- In the following, we show how it can be implemented in C++ by defining a Frame class

  - The assignment operator is overloaded to allow assignment of Frame objects

  - The multiplication operator overloaded so that it effects the concatenation of Frame objects, i.e. homogeneous transformation

Thus, assuming the frames T6, Z, B, G, and E have been declared, and the pose values x, y, theta, and blockHeight have valid values, the transformation equation

$$T6 = Z^{-1} \; B \; G \; E^{-1}$$

can be implemented as

```
Z = trans(0, 0, 0);
B = trans(x, y, 0) * rotz(phi);
G = trans(0, 0, blockHeight/2) * roty(180);
E = trans(0, 0, 100);
T6 = inv(Z) * B  * G * inv(E);
move(T6);
```

- Note: for the sake of clarity, we are adopting the convention that the frame variables are written in upper case

- Normally, in C++, the first character of an object is written in lower case and the first character of a class name in upper case

```
/*******************************************************************************************************************
*   Example pick-and-place program for a LynxMotion AL5D robot arm
*   ---------------------------------------------------------------
*
*   This application implements a simple robot program to grasp a simple object (a block),
*   lift it up, and place it somewhere else.
*
*   The position and orientation (pose) of the object and the goal position are specified in the input file.
*   (The pickAndPlaceVision application uses a camera to determine the object pose.)
*
*   The program uses task-level programming using frames to specify the object, robot, and gripper poses.
*
*   This application reads three lines from an input file pickAndPlace.txt.
*
*   The first line contains a filename of the file with the robot calibration data, i.e. for the inverse kinematic solution.
*   This allows the program to be used with different robots (by specifying the corresponding calibration data file).
*
*   The second line contains the object pose, i.e. the x, y, and z coordinates and the phi angle of the object (i.e. rotation about z).
*
*   The third line contains the destination pose, i.e. the x, y, and z coordinates and the phi angle of the destination (i.e. rotation about z).
*
*   It is assumed that the input file is located in a data directory given by the path ../data/
*   defined relative to the location of executable for this application.
*
*
*   David Vernon, Carnegie Mellon University Africa
*   4 February 2020
*
*   Audit Trail
*   -----------
*   No changes yet
*
*******************************************************************************************************************/
```

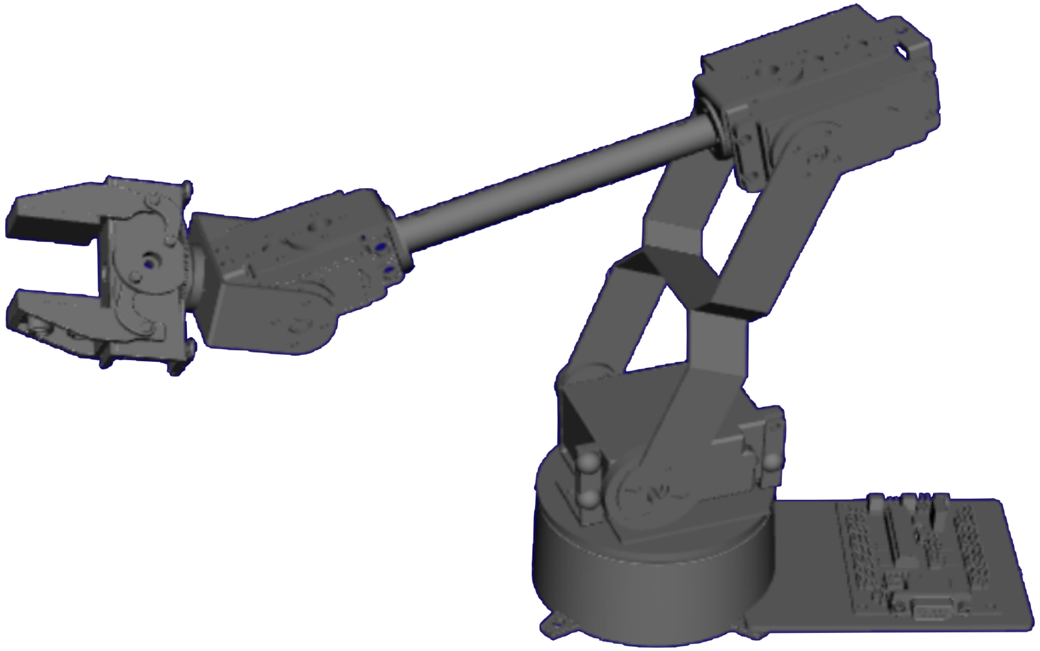
```
#include "pickAndPlace.h"

int main(int argc, char ** argv) {

   extern robotConfigurationDataType robotConfigurationData;
   bool debug = true;
   FILE *fp_in;                    // pickAndPlace input file
   int  end_of_file;
   char robot_configuration_filename[MAX_FILENAME_LENGTH];

   /* Frame objects */

   Frame E;
   Frame Z;
   Frame T6;
   Frame block;
   Frame grasp;
   Frame approach;
   Frame destination;

   /* data variables */

   float effector_length;           // this is initialized from robot configuration file

   float object_x          = -40;   // default values; actual values are read from the input file
   float object_y          = 150;   //
   float object_z          =   0;   //
   float object_phi        = -90;   // rotation in degrees about the z (vertical) axis

   float destination_x     =  40;   // default values; actual values are read from the input file
   float destination_y     = 150;
   float destination_z     =   0;
   float destination_phi   = -90;   // rotation in degrees about the z (vertical) axis

   float grasp_x           =   0;   // grasp pose relative to object and destination poses
   float grasp_y           =   0;
   float grasp_z           =  10;
   float grasp_theta       = 180;   // rotation in degrees about the y axis

   float approach_distance = 100;   // approach and departure distance from grasp pose in -z direction
```

```
/* open the input file */
/* ------------------- */

if ((fp_in = fopen("../data/pickAndPlaceInput.txt","r")) == 0) {
   printf("Error can't open input pickAndPlaceInput.txt\n");
   prompt_and_exit(0);
}


/* get the robot configuration data */
/* -------------------------------- */

end_of_file = fscanf(fp_in, "%s", robot_configuration_filename); // read the configuration filename
if (end_of_file == EOF) {
   printf("Fatal error: unable to read the robot configuration filename\n");
   prompt_and_exit(1);
}

readRobotConfigurationData(robot_configuration_filename);


/* get the object pose data */
/* ------------------------ */

end_of_file = fscanf(fp_in, "%f %f %f %f", &object_x, &object_y, &object_z, &object_phi);
if (end_of_file == EOF) {
   printf("Fatal error: unable to read the object position and orientation\n");
   prompt_and_exit(1);
}


/* get the destination pose data */
/* ----------------------------- */

end_of_file = fscanf(fp_in, "%f %f %f %f", &destination_x, &destination_y, &destination_z, &destination_phi);
if (end_of_file == EOF) {
   printf("Fatal error: unable to read the destination position and orientation\n");
   prompt_and_exit(1);
}
```

```
/* now start the pick and place task */
/* --------------------------------- */

effector_length = (float) robotConfigurationData.effector_z; // initialized from robot configuration data

E            = trans(0.0, 0.0, effector_length);                                          // end-effector (gripper) frame
Z            = trans(0.0 ,0.0, 0.0);                                                       // robot base frame
object       = trans(object_x,      object_y,      object_z)      * rotz(object_phi);      // object pose
destination  = trans(destination_x, destination_y, destination_z) * rotz(destination_phi); // destination pose
grasp        = trans(grasp_x,       grasp_y,       grasp_z)       * roty(grasp_theta);     // grasp frame w.r.t. object & destination frames
approach     = trans(0,0,-approach_distance);                                              // frame defined w.r.t. grasp frame

/* close the gripper */
/* ----------------- */

setGripper(GRIPPER_OPEN);
wait(1000);  //  1 second

/* move to initial approach pose */
/* ----------------------------- */

T6 = inv(Z) * object * grasp * approach * inv(E);

if (move(T6) == false)
   display_error_and_exit("move error ... quitting\n");;
wait(4000);  // 2 seconds

/* move to the grasp pose */
/* ---------------------- */

T6 = inv(Z) * object * grasp * inv(E);
if (move(T6) == false)
   display_error_and_exit("move error ... quitting\n");
wait(2000);   // 2 seconds

/* close the gripper */
/* ----------------- */

setGripper(GRIPPER_CLOSED);
wait(2000);
```

```
   /* move back to initial approach pose */
   /* ---------------------------------- */

   T6 = inv(Z) * object * grasp * approach * inv(E);
   if (move(T6) == false)
      display_error_and_exit("move error ... quitting\n");
   wait(3000);  // 3 seconds

   /* move to destination approach pose */
   /* --------------------------------- */

   T6 = inv(Z) * destination * grasp * approach * inv(E);
   if (move(T6) == false)
      display_error_and_exit("move error ... quitting\n");
   wait(3000); // 2 seconds

   /* move to the destination pose */
   /* ---------------------------- */

   T6 = inv(Z) * destination * grasp * inv(E);
   if (move(T6) == false)
      display_error_and_exit("move error ... quitting\n");;
   wait(2000);  // 2 seconds

   /* open the gripper */
   /* ---------------- */

   setGripper(GRIPPER_OPEN);
   wait(2000);  // 2 seconds

   /* move back to initial approach pose */
   /* ---------------------------------- */

   T6 = inv(Z) * destination * grasp * approach * inv(E);
   if (move(T6) == false)
      display_error_and_exit("move error ... quitting\n");;
   wait(3000);  // 2 seconds

   goHome(); // this returns the robot to the home position; could also do this with a move() as shown above
   return 0;
}
```

Lynxmotion AL5D Robotic Arm with serial interface



Robot Arms
Lynxmotion AL5D Simulator