CS36010- Robotic Applications
Report tob31

Robot Strategy

Introduction

For this robotic assignment the task was to program the fetch robot to navigate the set obstacles in the robot's way, go towards the one of the red blocks (perhaps manipulating the head), use the robot's arm to pick up the red block, take the red block back to the green bin, repeat for the other red block and then finish. The fetch robot must be loaded into the set environment "room22".

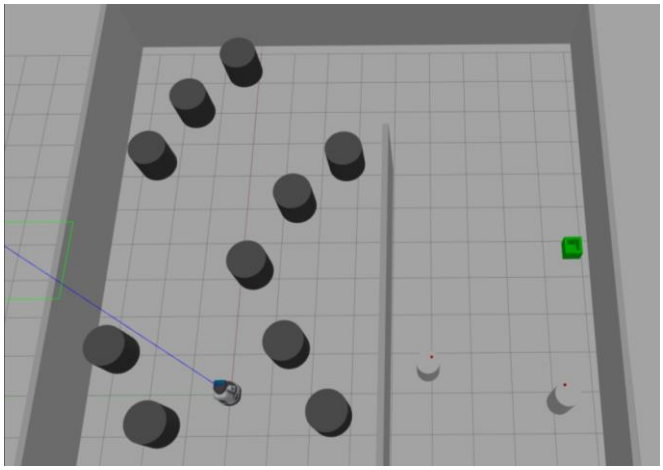
For context the fetch robot is a wheeled robot that has a 7DoF arm with a gripper attached to the body. There are additional joints to lift up the torso and for pan-tilt of the head.

In addition to the 2D laser scanner mounted in the base, sensors include an RGB-D camera mounted in the head and 2 IMUs, one in the base, the other in the gripper.

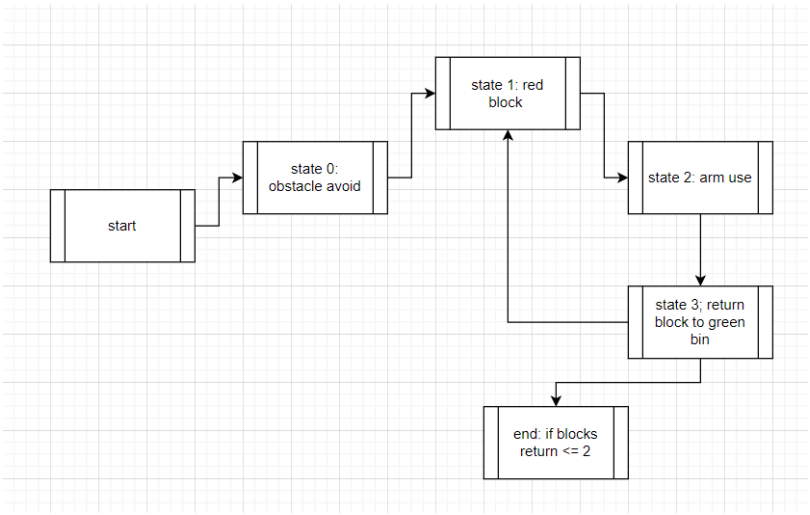
The maximum height with torso raised is 1.49m, the arm length is 94cm and the maximum opening size of the gripper is 100mm.

The robot can drive around at a maximum speed of 1.0m/s, although in simulation it can go faster.

The environment has black cylinders as obstacles in the first section of the maze-like structure, and red blocks on white cylinders at the end of the second sections. Here is a picture of the environment for reference. And one of the fetch robot.



Modelling



State 0 is for avoiding the obstacles it uses the 2D laser scanner mounted in the base to. To get the minimum distance from three sets of different range positions from the sensors. One straight in front. At roughly 50degrees to the left and at 50 degrees to the right. It also gets the max distance. Once it sees a red object it'll go to state 1.

State 1 is for finding and moving towards to the red blocks, when it sees them on the moments array, The head camera will move so that it keeps the red object visible by adjusting is up or down based on how far away it is from the centre of the screen, the robot will move towards it, it will linearly move forward at a value of 0.35, and the angular will change based on how far away the block is from the centre of the screen, and stop at a set distance from it and go to state 2.

State 2: this is where the sensors locate the red blocks and uses the arm to extend and open the griper to pick it up, the fetch robot will then do a 180 and go to state 3.

State 3: this is where the robot will locate the green bin, stop close to it, drop the red box into it, do a 180, then BlocksReturned will increment by 1 and go to state 1. If blocksreturned is equal to 2 the robot will stop.

Implementation

These are the headers for the files

```
#include <ros/ros.h>
#include <geometry_msgs/Twist.h>// For geometry_msgs::Twist
#include <opencv_apps/MomentArrayStamped.h>
#include <stdlib.h> // For rand() and RAND_MAX
#include <ros/ros.h>
#include <cv_bridge/cv_bridge.h>
#include <sensor_msgs/image_encodings.h>
#include <image_transport/image_transport.h>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <tf2/LinearMath/Quaternion.h>
#include <tf2_ros/transform_broadcaster.h>
#include <geometry_msgs/TransformStamped.h>
#include <sensor_msgs/LaserScan.h>
#include <actionlib/client/simple_action_client.h>
#include <control_msgs/FollowJointTrajectoryAction.h>
#include <exception>
#include <string>
#include <tf2/LinearMath/Quaternion.h>
```

These are the global variables I've used.

```
float xCo;
float yCo;
float xAr;
static const std::string OPENCV_WINDOW = "Image window";
//sensors
float minfront;
float minright;
float minleft;
float maxfront;
float maxright;
float maxleft;
float colourdetector;
bool colour = false;
int state = 0;
float tiltval;
float y;
int BlocksReturned = 0;
```

This is the function of getting the values from the moments array sensory that corresponds to the red blocks location.

```
void poseMessageReceived(const opencv_apps::MomentArrayStamped& msg) {
    //colourdetector = msg.moments.size();
    if (msg.moments.size() > 0){
        xCo = msg.moments[0].center.x;
        yCo = msg.moments[0].center.y;
        xAr = msg.moments[0].area;
        colour = true;
    }
```

This is the function of getting the minimum distances from the ranges. Just collecting the maxes, it's not used but it thought it would be useful.

```
void laserScanReceived(const sensor_msgs::LaserScan& msg) {
    minfront = 25.0;
    maxfront = 0;
    for(int i = 310; i<=350;i++){
        if (minfront > msg.ranges[i]){
            minfront = msg.ranges[i];
        }
        if(maxfront < msg.ranges[i]){// just collecting the
            maxfront = msg.ranges[i];
        }
    }
    minright = 25.0;
    maxright =0;
    for(int i = 510; i<=550;i++){
        if (minright > msg.ranges[i]){
            minright= msg.ranges[i];
        }
        if(maxright < msg.ranges[i]){
            maxright= msg.ranges[i];
        }
    }

    minleft = 25.0;
    maxleft = 0.0;
    for(int i = 120; i<=160;i++){
        if (minleft > msg.ranges[i]){
            minleft= msg.ranges[i];
        }
        if(maxleft < msg.ranges[i]){
            maxleft= msg.ranges[i];
        }
```

This is the code for the head movement.

```
void imageCb(const sensor_msgs::ImageConstPtr& msg) {
    //This function uses x and y coordinates from the contour moments
    cv_bridge::CvImagePtr cv_ptr;
    try {
        cv_ptr = cv_bridge::toCvCopy(msg,sensor_msgs::image_encodings::TYPE_32FC1);
    }
    catch (cv_bridge::Exception& e) {
        ROS_ERROR("cv_bridge exception: %s", e.what());
        return;
    }
    float depth = cv_ptr->image.at<float>(yCo,xCo);
    // for verification purposes:
    // draw circle of radius 10 around the xy point
    cv::circle(cv_ptr->image, cv::Point(xCo, yCo), 10, CV_RGB(255,0,0));
    // Update GUI Window
    cv::imshow(OPENCV_WINDOW, cv_ptr->image);
    cv::waitKey(3);
    //needs work
    //waitForMessage (const std::string sensor_msgs/CameraInfo&, NodeHandle &nh, ros
    static tf2_ros::TransformBroadcaster br;
    float f = 554.254691191187;
    geometry_msgs::TransformStamped transformStamped;
    transformStamped.header.stamp = ros::Time::now();
    transformStamped.header.frame_id = "head_camera_depth_frame";
    transformStamped.child_frame_id = "target_object";
    transformStamped.transform.translation.x = depth;
    //height 480
    //weight 640
    transformStamped.transform.translation.y = -(xCo-(640/2)/f)*depth;
    transformStamped.transform.translation.z = -(yCo-(480/2)/f)*depth;
```

```
    tf2::Quaternion q;
    //setRPY(roll, pitch, yaw) calculates quaternion rotation for transform
    q.setRPY(0, 0, 0);
    transformStamped.transform.rotation.x = q.x();
    transformStamped.transform.rotation.y = q.y();
    transformStamped.transform.rotation.z = q.z();
    transformStamped.transform.rotation.w = q.w();
    br.sendTransform(transformStamped);
    tiltval = depth;
```

These are the head positions. I used these briefly in the code but then I stopped using them. I decided to keep them in case I would need to use them in the future. Since the head movement wasn't finished.

```
// Our Action interface type for moving Fetch's head, provided as a typedef for convenience
typedef actionlib::SimpleActionClient<control_msgs::FollowJointTrajectoryAction> head_control_client;

std::string head_joint_names[2] = {[0] "head_pan_joint", [1] "head_tilt_joint"};

float head_joint_positions[3][2] = {//these are the head positions from the prac there not used i this
        [0] { [0] -1.57, [1] -0.76} ,
        [1] { [0] 1.57, [1] 1.45} ,
        [2] { [0] 0.0, [1] 0.0}
};
```

This is the function I wrote to reset the head but I couldn't get it to work so I left it out

```
/* this commented out code was going to be a function that resets the head
void HeadReset(control_msgs::FollowJointTrajectoryGoal head_goal){
    //reset head
    head_goal.trajectory.joint_names.push_back(head_joint_names[0]);
    head_goal.trajectory.joint_names.push_back(head_joint_names[1]);
    head_goal.trajectory.points.resize(1);
    head_goal.trajectory.points[0].positions.resize(1);
    head_goal.trajectory.points[0].positions[1] = 0;
    head_goal.trajectory.points[0].positions[0] = 0;
    head_goal.trajectory.points[0].velocities.resize(1);
    head_goal.trajectory.points[0].velocities[0] = 1;
    head_goal.trajectory.points[0].velocities[1] = 1;
    head_goal.trajectory.points[0].time_from_start = ros::Duration(5.0);
    head_goal.trajectory.header.stamp = ros::Time::now() + ros::Duration(1.0);
    head_client.sendGoal(head_goal);
    // Wait for trajectory execution
    head_client.waitForResult(ros::Duration(2.0));
}*/
```

This is the start of the main function, I subscribe to the base scan for the laser scanner and to the contour moments for the red blocks here, also for the green bin. Also publishing to the /cmd_vel

```
int main(int argc, char **argv) {
  // Initialize the ROS system and become a node.
  ros::init(argc, argv, "publish_velocity");
  ros::NodeHandle nh;
  ros::NodeHandle tb;

  // Create a publisher object.
  ros::Publisher pub = tb.advertise<geometry_msgs::Twist>("/cmd_vel", 10);
  // sub
  ros::Subscriber base = tb.subscribe("/base_scan", 1000,&laserScanReceived);

  ros::Subscriber sub1 = tb.subscribe("/contour_moments_red/moments", 1000, &poseMessageReceived);//used to be nh.
  //ros::Subscriber sub2 = tb.subscribe("/contour_moments_green/moments", 1000, &poseMessageReceived); //this is for the green object
```

This code here resets the head once the robot loads up since is at the start of the main. It's a lot of boilerplate code to do a simple motion.

```
control_msgs::FollowJointTrajectoryGoal head_goal;
//reset head
head_goal.trajectory.joint_names.push_back(head_joint_names[0]);
head_goal.trajectory.joint_names.push_back(head_joint_names[1]);
head_goal.trajectory.points.resize(1);
head_goal.trajectory.points[0].positions.resize(1);

head_goal.trajectory.points[0].positions[1] = 0;
head_goal.trajectory.points[0].positions[0] = 0;
head_goal.trajectory.points[0].velocities.resize(1);
head_goal.trajectory.points[0].velocities[0] = 1;
head_goal.trajectory.points[0].velocities[1] = 1;
head_goal.trajectory.points[0].time_from_start = ros::Duration(5.0);
head_goal.trajectory.header.stamp = ros::Time::now() + ros::Duration(1.0);
head_client.sendGoal(head_goal);
// Wait for trajectory execution
head_client.waitForResult(ros::Duration(2.0));
```

This is the start of the switch statement that controls the current state of the robot. Here you can see the code for state 0, it uses the laser scanner values collected from the function to avoid the obstacles by going in the opposite direction to where there is a close obstacle. And will try to centre itself between obstacles moving itself forward. I'm particularly proud of this implementation of the switch statement, here its very clean, the only thing that would make it cleaner is to have the states in separate functions potentially.

```
float x;
switch(state){
    case 0:
        msg.linear.x = 0.08*minfront;
        msg.angular.z = 0.1*(minleft-minright);
        if(colour){
            state = 1;
            base.shutdown();
        }
        break;
    case 1:
        //rest head
        //ros::Subscriber sub1 = tb.subscribe("/contour_moments_red/moments",
        x = 320-xCo;
        y = 240-yCo;
        if(xAr < 8300){
            msg.linear.x = 0.35;
            msg.angular.z = x*0.005;
        }
        else if(xAr > 8300){
            msg.linear.x = 0;
            msg.angular.z = 0;
            state = 2;
            //sub1..shutdown();
        }
        if(y < -220.0){
            head_goal.trajectory.joint_names.push_back(head_joint_names[0]);
            head_goal.trajectory.joint_names.push_back(head_joint_names[1]);
            head_goal.trajectory.points.resize(1);
            head_goal.trajectory.points[0].positions.resize(1);
```

This is where state 1 starts. It uses the contour moments values collected to slowly shift the robot on the angular to look for and centre the robots towards the red block, then it will move forwards. It should also tilt the head to constantly look at the red objects but I haven't been able to work it out yet.

```
x = 320-xCo;
y = 240-yCo;
if(xAr < 8300){
    msg.linear.x = 0.35;
    msg.angular.z = x*0.005;
}
else if(xAr > 8300){
    msg.linear.x = 0;
    msg.angular.z = 0;
    state = 2;
    //sub1..shutdown();
}
if(y < -220.0){
    head_goal.trajectory.joint_names.push_back(head_joint_names[0]);
    head_goal.trajectory.joint_names.push_back(head_joint_names[1]);
    head_goal.trajectory.points.resize(1);
    head_goal.trajectory.points[0].positions.resize(1);

    head_goal.trajectory.points[0].positions[1] = 1.1;
    head_goal.trajectory.points[0].positions[0] = 0;
    head_goal.trajectory.points[0].velocities.resize(2);
    head_goal.trajectory.points[0].velocities[0] = 1;
    head_goal.trajectory.points[0].velocities[1] = 1;
    head_goal.trajectory.points[0].time_from_start = ros::Duration(5.0);
    head_goal.trajectory.header.stamp = ros::Time::now() + ros::Duration(1.0);
    head_client.sendGoal(head_goal);
    // Wait for trajectory execution
    head_client.waitForResult(ros::Duration(2.0));
}
break;
```

This is the start of code of state 2 and 3, there are unfinished but you can see the logic that I was going to implement given more time.

```
case 2:
    msg.linear.x = 0;//just used to stop the robot and indicate its switched states
    msg.angular.z = 0;
    //arm code goes here
    //once the arm has picked up the red block it'll turn 180
    //state = 3;

    break;
case 3:
    //take red block to green object
    //reset head
    //ros::Subscriber sub2 = tb.subscribe("/contour_moments_green/moments", 1000, &poseMessageReceived);
    x = 320-xCo;
    y = 240-yCo;
    if(xAr < 8300){
        msg.linear.x = 0.35;
        msg.angular.z = x*0.005;
    }
    else if(xAr > 8300){
        msg.linear.x = 0;
        msg.angular.z = 0;
        state = 2;
        //sub2..shutdown();
        //once the arm lets go of the redbox turn 180
        //BlocksReturned + 1;
        //state = 1
    }
    //head movement will need to go here
    break;
```

This last section of code is about publishing and displaying information in the terminal.

```
// Publish the message.
pub.publish(msg);

// Send a message to rosout with the details.
ROS_INFO_STREAM("Sending velocity command:"
  << " linear=" << msg.linear.x
  << " angular=" << msg.angular.z
  << " state=" << state
  << " xAR=" << xAr
  << " y" << y);

  // Subscribe to input image topic using image transport
  image_transport::ImageTransport it(nh);
  image_transport::Subscriber depth_sub =
          it.subscribe("/head_camera/depth_registered/image_raw", 1, imageCb);
  cv::namedWindow(OPENCV_WINDOW);
  //ros::Subscriber sub = nh.subscribe("/head_camera/depth_registered/image_raw", 1, &imageCb);
  //ros::init(argc, argv, "tf2_broadcaster");

// Wait until it's time for another iteration.
ros::spinOnce();
rate.sleep();
```

Discussion and conclusion

I believe that all though I didn't finish the project, my implementation is quite robust and would be very flexible if the variations in the scenario where different, nothing in the program that is

specifically hard coded for the robots' motions so it should work, in any similar scenario. It makes use of the ros middleware well however I could use some more development on the head movement and arm movement, however once that would be completed the assignment would be perfectly completed.

Ethical Considerations Article

First full-time robot teacher employed

Introduction

For the first time a full-time robot teacher has been employed/implemented at a school in London. Should this robotic teacher be successful, they can be implemented around the country and possibly the world. To help the tackle current lack of teachers in the schools, and perhaps increase the potentially taught knowledge base to their students. Since robots are computational machines, they will not tire like biological humanoid teachers will, therefore could be more consistent at teaching at a high level. However human attitudes towards robotic teachers may become a concern.

Ethical considerations

So, lets begin to look at some of the ethical considerations of implementing robotic teachers, lets look at the negative's ethical implications first, For the robot to be able to teach students effectively they will need to be able to adapt their teaching methods to each of their students, to do this they will need to collect and store data about each individual student, this could potentially be a bad idea should this info be exposed.

Furthermore, will this robot be able to effectively adapt their teaching methods to not only different pupils but pupils with disabilities and learning difficulties, also how will pupils with disabilities and learning difficulties react and communicate with these robotic teachers.

Young students without strong parental figures often look towards teachers a role models, they would not be able to do this for robotic teachers as they would not be able to relate to them and subconsciously models themselves after them. Similarly, a robotic teacher may not be able to empathise with is students, and understand them their emotional reactions/changes

What if there's a medical emergency, can the robot react appropriately to a the said medical emergency or will it be unable to do anything, potentially leaving a young pupil in harms way.

In terms of social development, Pupils without classmates may miss out on developing valuable person to person interaction skills, this could promote antisocial behaviour in young children.

It has been proven time and time again that ai can be corrupted when given miss information what kind of measures would be taken to stop this happening, for example creating a racist robot ai.

Students answers if correct but not the same as the set answer how will the robot ai, react and give feedback, human teachers will recognise what a young pupil (that can't quite explain themselves corrects) is trying to say.

Now, let's have a look at the benefits of having robotic teachers, the very distinct lack of teachers and/or competent teachers in schools is significantly low in this country, these robotic

teachers could be mass produced and implemented in schools around the world to fill these large gaps and properly teach our young students which would greatly benefit our society.

These robotic teachers could potentially be far more effective teachers than biological human ones, these teachers could be programmed with the adequate academic knowledge to teach at a high level, potentially teaching these students better.

Discussion and conclusion

Let's begin to evaluate some of this points, for the robot collecting data about the students and building up a profile about them, this data could be leaked and break data protection laws, however this data will most likely be heavily protected, when compared to human teachers the only projection is the trust they won't spill the info themselves, this means the information collected by the teachers is safer with the robotic ones when compared to the human ones.

Robotic teachers could be developed/tailored to teach a specific set of pupils, for example those with disability's and/or learning difficulties. Potentially better than a teacher not trained to teach those pupils, machine learning techniques could be used to allow this to happen and refine the robots teaching process, these robots could be programmed to get around teaching difficulties with some pupils, for example robotic hand movements into sign language for deaf students, and visual audio representation for blind pupils.

Unfortunately, when it comes to the social development of students/children where there the only student, they most likely won't be able to get this from robotic teachers, potentially developing anti-social behaviour however it must be noted that in the similar situation with a human teacher this could also occur since they are the only student being taught. It just increases the probability of it happening with a robotic teacher. However, in some cases where there isn't a teacher available a robotic teacher would be able to fill this role, for example the student being in an isolated location away from normal civilisation or perhaps in quarantine.

Similarly, the young students would not be able to use these robots as role models in their lives should they be without strong parental figures, this could become a major issue for some students, however it must be recognised that the root of the problem here is with the lack of strong parental figures in their lives, which Is a separated issue. Although, it could be possible in the future to develop robotic ai teachers who could become strong role models for young children but currently this is only very hypothetical and very advanced.

Should a medical emergency occur it is very unlikely that a robotic teacher would be very helpful dealing with this situation. The child could be left in harms way. However, it is possible for the robotic teacher to be programmed to detect a medical emergency and immediately call for help, this is very similar to what untrained teachers would in this situation, is it unlikely that they would be able to provide any actual first aid training,

However, there is the possibility of in the future the robotic ai teacher being programmed to recognise what the medical problem, inform the human medical team, and potentially give first aid assistance. for example, mouth to mouth by blowing air into their mouth or chest compressions. But this must be stressed that the current advancements in robotic technology would most likely not allow these to be possible.

In terms of the robot using wrong data, supervised learning is used for the robotic teacher to stop it from becoming “evil” or “racist” like so have in the past. This machine learning to allow the robot to recognise current/ partially correct answers from pupils.

When it comes to the issue of the lack of capable teachers in this country, it could be fixed by implementing robotic teachers in place of them. However, this could also take away jobs from current teachers stopping them from teaching and putting them out of work.

The robotic teachers would be more effective than the teachers but wouldn’t have programmed with any actual practical knowledge or life experience to be taught to the students only theoretical knowledge taught on the syllabus.

Overall robotic teachers would be a great addition to the work force but should only be used when it is needed and should never replace human teachers.

Self-assessment

I believe that for this assignment I would score 65% although I could not fully finish the work, it is clear to see what is need to be done, and the logic is clear, its just a matter of implementing a working solutions, apart from that I believe I have worked very well on this assignment,

References

Newton D. P., Newton L. D., Humanoid Robots as Teachers and a Proposed Code of Practice, Durham University, (2019).