Cognitive Interaction with Robots

Safety & Control Improvement in Human-Robot Interaction

Authors:
Alex Pardo
Marc Bolaños
Pablo Pardo

Professors:
Cecilio Angulo
Marta Díaz

January 9, 2015

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Contents

1 Introduction

This project is comprehended within the teleoperated robots area. A broad topic recently being pulled further given the increasing need of elderly attention.

The robot we were working with was MASHI, which is a teleoperated robot with the purpose of guiding in "La Bóbila" social centre in L'Hospitalet del Llobregat. The root consists in an Arduino board embedded to a wheeled structure that resembles the human figure.

The environment the robot is going to work is placed in the main entrance of the social centre, where some temporary expositions are hold. MASHI is supposed to welcome the visitors and guide them through the exposition being able to communicate with the visitors.

Our project is focused on two different but necessary aspects:

1. **Collisions Avoidance**: We developed a simple but efficient way to avoid collisions using an RGB-D camera attached in the neck of MASHI. Our method tackles the so called *Freezing Robot Problem*.

2. **Gesture Control**: Facilitate the teleoperator control of the robot by allowing him/her to use the hands to move the robot. This feature will allow the teleoperator to manage more complex movements such as control head, arms, hands and so on. We use a Microsoft Kinect to capture the depth image of the teleoperator and be able to recognise the gestures.

The Section 2 is a brief compilation of related works and common problems in the area. In the Section 4 we explain with all the detail how was carried out the implementation part. The Section 5 contains the experiments with users done in order to test the acceptance and efficiency of out work. In the last section (Section 6) we expose our conclusions of this project and the possible future work.

2 Related Work

2.1 The Freezing Robot Problem

The *Freezing Robot Problem* is a classic problem in robotics and consists of finding a balance between being conservative in terms of movement and safety and reach efficiently the goal. If the robot is too conservative and the environment too complex could end up freezing because none of the possible movements would be considered to be safe enough.

A classical approach to this problem is to use a planner and try to predict the movement of the surrounding in order to decide the next move, however these approaches usually fail to find a path when the environment is too complex. There are other approaches that tackle the problem from different perspectives. Peter Trautman and Andreas Krause [3] studied and proposed a non-parametric statistical model based on dependent output Gaussian processes that can estimate crowd interaction from data. Chung-Che Yu and Chieh-Chih Wang [4] proposed a Learning from Demonstration (LfD) approach proving that these kind of methods are efficient to both avoid collisions and freezing free navigation.

2.2 Gesture Recognition

Gesture Recognition is an important topic in computer vision and an active line of research for lots of companies. The reason behind this interest in the area is because of the multiple and revolutionary applications to the world of technology, it changes completely the conventional machine input mechanisms (mouse, keyboard and even touch-screens).

This technology has being successfully applied in the game industry, changing the gaming experience to a whole new level. Its also has being applied to device control such as tv or intelligent home systems improving the user experience.

One of the greatest advances in the topic was the improvement and large scale production of RGB-D cameras which started with the Kinect camera developed by *Microsoft* for the gaming industry. This busted the research in the area. There are countless examples of papers studying the area, a good example of them was presented by Lin Song1, Ruimin Hu, Yulian Xiao and Liyu Gong [2] in 2013 proposing a method for hand segmentation and gesture recognition in real time. Another interesting example was given by Yi Li [1] with a method capable of recognizing 9 different gestures with an accuracy between 84% and 99%.

3 Analysis

In this section we will analyse each one of the two main objectives of this project in order to explain the main problems we would find and the possible solutions to them.

3.1 Gesture control

After testing the MASHI robot by using the web interface already developed we realized that controlling it is not an easy task since we are not only able to move the robot itself but also the head, so we have nine different buttons to press (five for motion and four for head movement). The team in charge of developing MASHI plans to add two arms two the robot, so the number of keys would be increased. For this reason we decided to move some of the controls to a more intuitive gesture-based system. For doing so we would use RGB-D images in which we are able to get the skeleton of the teleoperator in order to easily recognize gestures.

Instead of moving all the controls, we decided to first implement the motion gestures and keep the head movement in the keyboard. This means the teleoperator would use the keyboard while performing gestures. The problem is that the system should be able to track the skeleton of a seated person. Microsoft Kinect SDK allows to track a person even if he/she is seated.

3.2 Collision avoidance

Also in the test we performed with the robot, we discovered that the communication between teleoperator and MASHI is not perfect and some orders are missed over the channel. We have to take into account that every wireless communication system is affected by interferences and errors that causes missing packets. So, if the robot is about to collide with an obstacle we could miss it because a camera problem (we are not able to see it) or because the *STOP* order is not received by the robot. In order to solve this conflicting situations we decided to add to the robot the ability of detecting obstacles in front of it. We would use a RGB-D camera in order to track distances and be able to tell the robot that we are approaching something that would block the trajectory.

4 Implementation

4.1 Collision Avoidance

We use the depth map given by the RGB-D camera from the robot to analyse whether there is collision risk. We use a sliding window to compute the minimum average distance among the patches (Eq. 1) and get the approximate distance to the closest visible object to the robot.

$$\min \frac{\sum_{w,h}^{i,j} depth(p_{i,j})}{w * h} < threshold \ \ \forall p \in Img \tag{1}$$

By setting two thresholds the robot sends a warning to the teleoperator, when there is an object close enough, or stops, when there is an object too close. Once the robot is freeze because of an obstacle the teleoperator can still move backwards (at its own risk) or turn around.

In order to tackle the **Freezing Robot Problem** we use a time window to store the closest distance and check if it disappears in time, i.e. if the sum of the distances in that window frame is larger that certain threshold, then we consider that the object is a danger, otherwise the obstacle is just passing by and so not a real danger.

Since stop the robot abruptly might scare or disturb the people around we decided to **reduce the speed** of the robot once it enters to a warning zone inverse proportionally to the closest distance until it reaches the stop zone where it stops. We also setted the backwards speed to half the forward speed since there is no camera in the back and the teleoperator will not be aware of any obstacle.

4.2 Gesture Robot Control

For the *Gesture Robot Control* we use a Kinect camera setted in the teleoperator side. It has two main phases, *depth image processing* and *gesture recognition*.

- **Depth Image Processing**: We use OpenNI sdk to extract the skeleton of the teleoperator. OpenNI skeleton tracking algorithm is not easy to find, however the Microsoft approach is to use a Random Forest classifier trained using a labelled dataset of depth images.

  The result is an efficient algorithm that works in real time.

- **Gesture Recognition**: We use a gesture recognition system based on right hand positions in the space (3D) and its variation over time, i.e. three axis and its two possible directions, which do 6 different gestures.

The movement is controlled by an empirically tested threshold.

Moving the hand in the depth axis will command to the robot to move forward or backward depending on the direction of the hand's movement, moving the hand in the horizontal axis will send to the robot the order of turning right or left and finally the vertical axis will send an stop signal to the robot.

4.3 Communication

All the connections on the system are shown in Figure 1, boxes in white represent those parts that are new. Next we detail how we added the two new modules:

- The **Anti-collision system** code runs on the MASHI side since it should be able to stop it even if the communication channel is broken. This part of the system runs on the Robot controller (which is a Server). The distance obtained is broadcasted over the network in order to allow everyone to know it. This part of the system also controls the speed reduction when approaching an obstacle as described before.

- On the other side, we have the **Gestural Control** which runs on the teleoperator machine. The application uses Openni and Nite libraries so it must be written in C++ while all the system is implemented in node-js. In order to enable the communication between languages, we defined a Client-Server protocol inside Kinect Server, which is getting data from the gesture recognition system (Gestures Camera in the figure) and sending it to the teleoperator application. Then, this will send the proper commands to the Server. The C++ - node js transition is made inside Kinect Server which is implemented using C++ but binded on node js using node-gyp (i.e. C++ code is executed from node js)

This complex structure allows us to maintain as much as possible the original communication since we are adding the newest modules (Gesture recognition) at the client side, making it transparent for the server placed on the robot. The collision avoidance system is set on the robot server because it does not require any communication with the clients.

images/communication.png

Figure 1: System communication diagram

5 Test with Users

5.1 Experiment Design

The experiments we performed have two parts:

- **Gesture Control and Collision Avoidance**: To test this part the tester has to teleoperate MASHI using our gesture control system and perform some simple tasks, going out of the laboratory room and coming back. The testers are also asked to try to collide and interact with the environment so they could test the effectiveness of the collision system. After the test the tester has to answer a short survey about the user friendliness and the efficiency of our system.
- **Vital Space**: In the second and last part of the test we focus on the personal distance the users feel good interacting with the robot. We ask the users what is the distance to the robot they feel comfortable with and try approaching the robot to them.

The small survey consists of the following 10 questions:

1. The system is easy to use. (Disagree 1 to 5 Totally Agree)
2. The system is intuitive. (Disagree 1 to 5 Totally Agree)
3. The system responds accordingly to your expectations. (Disagree 1 to 5 Totally Agree)
4. What would you improve from the system?
5. The gesture commands are easy and intuitive. (Disagree 1 to 5 Totally Agree)
6. Which gesture is more complicated to detect?
7. Which improvement do you think would be interesting for the gesture control?
8. The anti-collision system is useful. (Disagree 1 to 5 Totally Agree)
9. Do you think the speed reduction is useful?
10. After the test, at which distance you consider the robot must stop in order to avoid a collision with a human?

5.2 Experiment Results

6 Conclusions

In this project we improved the MASHI control system in order to help the teleoperator to operate the robot and also the safety which is really important in the environment the robot is placed. We used a User-Centered Design methodology in order to develop the proposed system. We completed a full cycle and set the things for the next iteration.

The cycle started with a proposal of the system, went through a phase of analysing the state of the art, then we designed the system according to the information we had about the robot, the environment and the related literature. Once we implemented the proposed design we tested it with potential users in order to retrieve information and improve the design.

Taking into account the problems described in Section 5 we define the following improvements as future work:

•

References

[1] Yi Li. Hand gesture recognition using kinect. In *Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference on*, pages 196–199, June 2012.

[2] Yulian Xiao Liyu Gong Lin Song, Ruimin Hu. Real-time 3d hand gesture recognition from depth image. ICSEM, 2013.

[3] Peter Trautman and Andreas Krause. Unfreezing the robot: Navigation in dense, interacting crowds. In *IROS*, pages 797–803. IEEE, 2010.

[4] Chung-Che Yu and Chieh-Chih Wang. Collision- and freezing-free navigation in dynamic environments using learning to search. In *Technologies and Applications of Artificial Intelligence (TAAI), 2012 Conference on*, pages 151–156, Nov 2012.