Voice- and gesture-based user interface

PROJECT PLAN

Master thesis

Project period: Summer Semester, 15.04.2014 – 15.10.2014

Student name: David Itten

Status: 3-rd master semester

E-Mail address: ittend@student.ethz.ch

Supervisor: Dr. Jiwon Shin, Chair of Software Engineering, ETH Zürich
Andrey Rusakov, Chair of Software Engineering, ETH Zürich

1. PROJECT DESCRIPTION

Overview

SmartWalker is a high-tech extension of a walker that aids people of reduced mobility in moving around. Equipped with sensors and actuators, the walker is currently able to move to user-defined destinations. The goal of this project is to develop an algorithm for gesture and voice recognition for the walker, using the walker's PrimeSense 3D Sensor as the input device. This work will enable the walker to interact with its user more naturally and to find its destinations autonomously.

Scope of the work

The SmartWalker, which is equipped with a PrimeSense 3D sensor, should move to the calling person on request (e.g. a hand gesture or a voice command). This should happen in the autonomous mode, which means, that the walker should locate and move to the calling person without direct control commands. During travel, obstacles should be recognized and avoided, furthermore a traversable way should be found to the target person. If there are multiple persons in a room, the person that gave the command should be recognized and located. As soon as the walker arrived at the target person, it should switch to the assistant mode (not part of this work), in which no further autonomous movement is undertaken and the leg movement of its user is sensed to detect the direction and speed of the users movement.

Additionally it is possible that the walker drives back to a predefined parking position after usage. This position could possibly be used to charge the device. If the corresponding command was given to the walker, it switches back to the autonomous mode and moves automatically to the parking position, again avoiding possible obstacles.

Intended results

- Gesture and voice recognition which is needed to give commands to the SmartWalker
- Usage of the laser sensor to recognize obstacles and to avoid them accordingly
- Switching between autonomous and non-autonomous mode (assistant mode)
- Implementation of a control application in Eiffel, using Roboscoop, ROS and pcl (see Method of work)
- Documentation (see Quality management → Documentation)

2. BACKGROUND MATERIAL

Reading list

- Lecture slides explaining Roboscoop: http://se.inf.ethz.ch/courses/2013b_fall/rpl/lectures/02_ROS_Roboscoop.pdf (consulted in March 2014)
- ROS online documentation: http://wiki.ros.org/ (consulted in March 2014)
- pcl project website: http://www.pointclouds.org/ (consulted in March 2014)
- openni_tracker (part of ROS) documentation http://wiki.ros.org/openni_tracker (consulted in March 2014)
- Sushmita Mitra and Tinku Acharya, IEEE: Gesture Recognition: A Survey http://citeseer.ist.psu.edu/viewdoc/download;jsessionid=8E82537D3AB66D3BDC18531B2D7228A9?doi=10.1.1.107.6243&rep=rep1&type=pdf (consulted in March 2014)
- Xi Chen and Markus Koskela, Aalto University School of Science: Online RGB-D Gesture Recognition with Extreme Learning Machines http://users.ics.aalto.fi/chenxi/icmi2013.pdf (consulted in March 2014)

3. PROJECT MANAGEMENT

Objectives and priorities

High priority:

- Fulfilment of the must criteria (steps 1-7) of the master thesis
- High code coverage achieved by the created test classes
- Updated project report with all relevant details

Medium priority:

- Fulfilment of additional feature which is described in step 8

Low priority:

- Fulfilment of additional feature which is described in step 9
- Optional additions to the project

Criteria for success

- At least project steps 1 to 7 fulfilled by the deadline
- Application well documented and tested
- Application tested in a real environment
- Ability to give a live demonstration of the created application

Method of work

The following techniques shall be used in the project:

- **SCOOP (Simple Concurrent Object-Oriented Programming)**

SCOOP is integrated into Eiffel-Studio and allows the object oriented design of multithreaded applications. SCOOP offers easy means to start separate features, to synchronize them and to define critical regions in the program text.

- **ROS (Robot Operating System)**

ROS provides libraries and tools for the creation of robotic applications. It provides easier access to hardware components of the robot by offering device drivers and libraries. Furthermore it defines nodes as individual components of a robot application and the communication among them by using a message passing protocol and a publish/subscribe pattern.

- **openni_tracker**

The openni_tracker is part of ROS. It tracks a person and detects the pose of a human. There are several frame names which can be used to access for example the position of the user's hands or other parts of the body.

- **RoboScoop Framework**

The Roboscoop framework [3] is divided into three layers which are all using SCOOP to handle concurrency. The Roboscoop framework manages the communication between the different layers. The following layers were used:

  - **Control layer**

    Contains stateless feedback control loops to couple sensors to actuators. The control software of the robot is written in this layer, which incorporates all primitive robot behaviour.
  - **Sequencer layer**

    This layer interacts with the control layer to fulfil a task. It is stateful and relies on primitive behaviours from the control layer.
  - **Deliberator layer**

    The Deliberator layer performs time-consuming deliberative computations.

- **Point Cloud Library (**PCL**)**

Library for 2D/3D image and point cloud processing

Project Setup

To facilitate testing and because the SmartWalker has to be updated to the newest Roboscoop framework and the PrimeSensor has to be mounted, the development is first done on a smaller Thymio II robot. In a later stage of the project, the software will be tested and adapted to the SmartWalker. The following setup will be used for developing the software:

- PrimeSense Sensor
- Thymio II Robot
- 2 USB extension cables
- 1 USB to MicroUSB cable

Quality management

Documentation

The documentation should at least consist of the following parts:

- Methods used for gesture and voice recognition
- Methods used for person tracking
- Methods used for object recognition and avoidance
- Design choices
- Evaluation and results
- User documentation for the created application

Validation steps

Throughout the software development process a set of test classes shall be created to ensure the correct functioning of the individual software components. For testing, the built in AutoTest[4] function of Eiffel Studio is used. The testing should mainly rely on manually created test classes. To test the effectiveness of the gesture recognition, person tracking and the automatic movement, tests in a real environment should be done with the Thymio II robot. Furthermore the software should be created according to the design by contract paradigm. Which means that pre- and post-conditions as well as class invariants should be used.

4. PLAN WITH MILESTONES

Project steps

1. **Learn ROS and Roboscoop**
   - Learn the usage of ROS together with the Roboscoop framework. Work through the tutorials on the ROS website.
2. **Setup of work environment, connection to actuators and sensors**
   - Installation of the necessary tools and libraries to connect to the sensors and actuators of the SmartWalker. Create an application that allows controlling the actuators and reading out all sensor values.
3. **Locating the Person that gave a command to the SmartWalker**
   - Literature review

- Locate the person that gave a command to the SmartWalker by moving the mounted PrimeSense 3D sensor and recognizing the person that gave the command.

4. **Gesture recognition that allows the user to give commands to the SmartWalker**
   - Find an appropriate way to give commands to the SmartWalker by either gestures or voice commands. Creation of a node that recognizes the commands of the user and executes the appropriate actions.
5. **Direct, autonomous movement to the target person without obstacles**
   - Autonomous movement of the robot to the calling person. With the following restrictions:
     - A single person is in the room (or view of camera)
     - Person is in the range of the camera
     - The SmartWalker can use the direct way between its stand-by location and the calling user (there are no obstacles).
   - Autonomous movement back to the predefined parking position.
   - To following commands can be given to the robot by gestures or voice commands:
     - Come here
     - Stop
     - Go to the charging station
6. **Autonomous movement to the target person (short distance) with obstacles**
   - Recognition of obstacles in front of the SmartWalker. Measure size and distances of obstacles in order to avoid them when moving.
   - Autonomous movement of the robot to the calling person. With the following restrictions:
     - A single person is in the room
     - The person is in the range of the camera
   - Test the created solution on the tablet computer which is used as the control device of the SmartWalker (which has less performance than the notebook which is used during development).
7. **Autonomous movement to the target person with 3 or more persons in the room**
   - Autonomous movement of the robot to the calling person, when at least three persons are in the same room. The SmartWalker has to recognize and navigate to the calling person.
8. **Autonomous movement to the target person (long distance) with obstacles**
   - Autonomous movement of the robot to the calling person. If the person is not in range of the camera, find the direction from which the command was given. Move until the person is in the visible range and go to the target person.
9. **Adding additional commands to the SmartWalker**
   - E.g. a command to turn the walker by 180 degrees and probably other useful and more sophisticated commands.

Deadline

15. October 2014

Tentative schedule

1. **Learn ROS and Roboscoop**
   - 2 Weeks
2. **Setup of work environment, connection to actuators and sensors**
   - 2 Weeks
3. **Locating the Person that gave a command to the SmartWalker**
   - 1.5 Weeks
4. **Gesture recognition that allows the user to give commands to the SmartWalker**
   - 4-5 weeks
5. **Direct, autonomous movement to the target person without obstacles**
   - 1 week
6. **Autonomous movement to the target person (short distance) with obstacles**
   - 3 weeks
7. **Autonomous movement to the target person with 3 or more persons in the room**
   - 2 weeks
8. **Autonomous movement to the target person (long distance) with obstacles**
   - 3 weeks
9. **Adding additional commands to the SmartWalker**
   - If time allows
10. **Literature review, data collection and analysis, evaluation and finishing report (Draft)**
    - 4 weeks
11. **Corrections, final version of report**
    - 1 week

REFERENCES

[1] Chair of Software Engineering: *Semester-/Diplomarbeiten*; Online at: http://se.inf.ethz.ch/projects/index.html, consulted in October 2002.

[2] Bertrand Meyer: *Object-Oriented Software Construction, 2nd edition*, Prentice Hall, 1997.

[3] Chair of Software Engineering: Roboscoop project description; Online at: http://se.inf.ethz.ch/research/roboscoop/roboscoop.pdf, consulted in March 2014.

[4] Eiffel Software: Testing: Background and basics; Online at: http://docs.eiffel.com/book/eiffelstudio/testing-background-and-basics, consulted in March 2014.