A vocal interface to control a mobile robot

Roberto Gretter, Maurizio Omologo, Luca Cristoforetti, Piergiorgio Svaizer
FBK, Povo (Trento)

Introduction

In human-machine communications, one of the most common dreams is about talking with a robot; still, working systems are few. To our opinion, the best way to control a robot should be a mix of voice and gesture, with few linguistic restrictions and from a reasonable distance, until about 5 meters. Language skills of the robot have to cover at least sentences needed to specify the commands it can execute. We are building a system which uses distant speech to control a mobile robot; communication with the robot is always active in some subdomains (navigation commands, agenda, control of parameters, teaching, ...) and the system is able to reject unknown sentences.

Architecture

Our robot is based on the TurtleBot 2 platform. The basic structure is a Kobuki moving base, a Microsoft Kinect device and an entry-level laptop. The Kobuki base is equipped with two motors, bumpers, encoders on the wheels and a gyroscope. To this standard setup we added eight digital MEMS microphones, some LEDs and an Arduino-based LCD screen.

The software architecture is based on ROS, an open source platform for Ubuntu that helps in building robot applications. The main concept is based on software nodes that collaborate, supervised by a master. Nodes can run on different machines to obtain a distributed architecture.

We added new nodes to handle multichannel audio acquisition, beamforming, sound source localization, Arduino LCD. In addition, other specific nodes interact with the speech recognizer and the dialogue framework.

Commands

The language that the robot can understand covers some subdomains, each modeled by hand-defined grammars – also stochastic grammars can be used if a corpus is available - that allow to express the possible commands with relatively complex sentences in a variety of ways. The lexicon is composed of about 5000 words. When a command is incomplete, a mixed-initiative dialogue helps to get the missing parameters. The most important command types are:

- basic navigation commands (*go backward one meter and a half*; *turn right 90 degrees*);
- multimodal commands, using different input modalities:
  - ***come here*** needs to localize the position of the speaker using his voice - he could be behind the robot;
  - ***go there + gesture*** combines speech and gesture, seen by the robot via Kinect - user must be in front of the robot;
  - ***follow me*** combines the two: based on localization via speech, first the robot will move to the speaker and then will use Kinect to follow him;
- information commands: questions about the capabilities of the robot; get/set values for some parameters (speed, etc.);

- teaching commands: to provide new information, e.g., labeling a given position (*learn, this is the entrance door*);
- mid level navigation commands: (*go to the entrance door*) that use ROS primitives to reach the position;
- compound commands: up to four navigation commands that will be executed in sequence;
- agenda (*set a new appointment for next Tuesday: phone call with John at 3PM*; *tell me what I have to do this afternoon*).

The system is also able to solve pronoun resolution, useful in dialogues like: User: *robot, how much is your speed?* Robot: *speed is 0.4.* User: *raise **it** by 0.2*. Robot: *changing speed from 0.4 to 0.6.*

Data acquisition

To test the system we acquired data in an apartment: 7 speakers in 3 sessions had to drive the robot to execute 14 tasks in different conditions (close to the robot or not – respectively < 2 meters or > 2.5 meters, turned toward the robot or not); about 30 minutes were needed on average to perform all the tasks (min 21, max 40). Entire speech data sequences (total duration 3h 19m) were acquired and processed to get time markers and ASR output. Manual correction was then performed to obtain a reliable transcription. Also a semantic representation was derived from the transcriptions.

Results

Speech segments with manual time boundaries were divided into dev and test sets. Preliminary results in terms of Word Accuracy (WA) and Semantic Accuracy (SA) are 77.96% (WA) and 81.55% (SA) for the dev set, 81.73% (WA) and 85.28% (SA) for the test set. In the paper more results will be given.