BORG

The RoboCup@Home team of the University of Groningen

Team Description Paper

2015

Amirhosein Shantia, Anton Mulder, Ben Wolf, Rik Timmers, Rick van der Mark, Levente Sandor, Luis Knigge, Stef van der Struijk, Gereon Vienken, Francesco Bidoia, and Noel Luneburg

Faculty of Mathematics and Natural Sciences, University of Groningen
Dept. of Artificial Intelligence
Cognitive Robotics Laboratory
Groningen, The Netherlands
http://www.ai.rug.nl
http://www.teamborg.nl

Abstract. This paper provides a description of the BORG team's robotic platform for the competition in the RoboCup@Home league. The robot is being developed at the Artificial Intelligence department of the University of Groningen, The Netherlands. The aim of the current design is to perform general service robot tasks as required by the @Home league of the RoboCup initiative utilizing mainly commercially available hardware components, open source libraries and a framework developed at the Cognitive Robotics Laboratory at the University of Groningen. An overview of the hardware and software specifications is given, with emphasis on the architecture and the methods currently being developed to address regular issues found in today's robotics such as navigation, recognition, manipulation and interaction.

Keywords: Robocup, Robocup@Home, Robotics, Domestic environments, Human Robot Interaction

1 Introduction

The BORG team resides at the Artificial Intelligence department in the faculty of Mathematics and Natural Sciences at the University of Groningen, The Netherlands. The BORG is one of the first Dutch teams in the RoboCup@Home league.

Our current team consists of approximately 10 students and faculty members from the department of Artificial Intelligence and Computer Science. The BORG team is named after the small castles typically built on the hills that surround the Groningen province.

Accomplishing the tasks specified for the RoboCup@Home competition requires the students to be trained in pattern recognition on sensor data, human-robot interaction, actuators control, machine learning, reasoning and language

processing. We use development techniques from eXtreme Programming such as unit-testing and agile design. The robot niche is defined as a stereotypical "home" environment for humans. Our approach towards autonomy is to combine a Volksbot RT-3 robot platform with a robotic arm and several sensors.

This is an excellent combination for a platform where we can do what we are good at: developing Artificial Intelligence software. Nevertheless, the most important aspect of the project is that we do it because it is fun to do!

2 Robot platform

2.1 Hardware architecture

We aim to keep on improving our hardware architecture annually. Our previous robot is shown in figure 1. Our current hardware design provides us with higher load capacity and improved movability.

Our new design is based on a Volksbot RT-3 platform. On top of that we have a frame with a robotic arm and several sensors. For the human-machine interaction we added a display with a human-like face on it.

The Volksbot Rt-3 has been extended with a rich set of additional sensors. This includes two normal HD cameras, a microphone array, sonar sensors, two Kinect (for Xbox 360) cameras, 2 XTion PRO LIVE cameras and 2 laser range finders These additional sensors allows us to accomplish improved performance in object recognition, human robot interaction, manipulation, gesture recognition and navigation.

Extra processing power is provided by the use of additional laptops located just underneath the robotic arm. These laptops and all other systems are interconnected using a local TCP/IP network.

2.2 Software architecture

The software architecture consists of sensor modules providing data about the world, and behavior modules that use this data to perform actions in the world. A separate navigation module uses the sensor information to estimate our location and orientation, and builds up a topological map to be able to reach previously visited destinations.

Each developed sensor module can run in parallel and can have a particular task. Multiple vision modules may for example be responsible for either object recognition, face recognition or even emotion recognition. Furthermore, several modules may be detectors for the same category of objects: in this case, detectinos from multiple systems can be combined to increase realiability.

We reuse existing and freely available software as much as possible. For vision, we use the OpenCV library[1]. The 'libfreenect'[2] and 'openni'[3] libraries are used to

[1] OpenCV is available from `http://opencv.willowgarage.com/wiki/`
[2] libfreenect is available from `http://openkinect.org/wiki/Main_Page`
[3] openni is available from `http://openni.org/`

interface with the Kinect sensor unit. For machine learning we use the 'pyBrain'[4], 'pyFLANN'[5], and 'pyFANN'[6] libraries.

Furthermore, our own custom made software framework is compitable with ROS[7]. This enables us to use third party software components and algorithms, such as AMCL and GMapping for navigation, more easily.

3 Focus of research interests

3.1 Grid Occupancy and Vision based navigation system

The navigation module of our team is now mainly based on grid occupancy methods. The main problem addressed by occupancy grid mapping the problem of generating a consistent metric map from noisy or incomplete sensor data with additional knowledge of robot pose. Even with all these information it is sometimes difficult to say whether a place in the environment is occupied or not, because of ambiguities in the sensor data. Occupancy grid maps solve such problems by generating probabilistic grid maps. These grid maps are usually two-dimensional but nowadays with use of time of flight camera and rotation 2D lasers, 3D grids are also popular. The standard occupancy grid mapping algorithm is a version of Bayes filters, just like any other major mapping algorithm.

In addition to the popular gird based method, we do research on additional vision based navigation systems. In our visual method, the robot brain organizes a set of visual keywords that describe the robot's perception of the environment similar to that of human topological navigation. The results of its experiences are processed by a model that finds cause and effect relationships between executed actions and changes in the environment. This allows the robot to learn from the consequences of its actions in the real world. The robot is resistant to non-major changes in the environment during training and testing phases. More specific, the robot takes several pictures from the environment with an RGB camera during the training phase. The raw images will be processed using the histogram of oriented gradients method (HoG) to extract salient edges in major directions. By using clustering on HoG results, similar scenes will be clustered based on visual appearances. Furthermore, a world model is made from the observations and actions taken during training. Finally, during testing, the robot selects actions that maximize the probability to reach its goal using model-based reinforcement learning algorithms.

3.2 Online behavior learning

Our behavior modules don't have a strong central controller. Behaviors applicable in a certain context are executed. In each of the tests at the competi-

[4] pyBrain is available from `http://pybrain.org`

[5] pyFLANN is available from `http://www.cs.ubc.ca/~mariusm/index.php/FLANN/FLANN`

[6] pyFANN is available from `http://leenissen.dk/fann/wp/`

[7] ROS (Robot Operating System) is available from `http://www.ros.org`.

tion, a different suite of behaviors and preconditions will be used to ensure a proper behavior for a specific context. There might be different implementations of the same behavior, so different methods to pick up an object might be implemented. The interval estimation algorithm (a machine learning system for real-time robotic learning) is used to select the best-performing implementation of a given behavior. This system is adaptive, so that if circumstances change other behaviors can be chosen. For example, grasping a bottle might use grasp behavior 1, but if the bottle is very different from the previously trained bottle, and it can't grasp it, it starts using the next best behavior.

If a behavior keeps failing, for example selecting similar behaviors with the same post-conditions but running out of time so it gets bored, then the behavior module raises a flag to the reasoning module which checks whether it has a general solution to the problem. An example of a general solution is to go back to the human and tell him/her which behavior failed. This mechanism is integral in our architecture and should solve a lot of the problems of the GPSR test.

If the robot is in the 'playing' mode, the boredom still works, but then it starts adapting its behaviors if the set of implemented behaviors keep failing or only succeed partially, for example by the use of parameter tuning, reinforcement learning or genetic programming.

3.3 Interactive scenario interpretation using scripts

For the general purpose service robot challenge, we are working on a script-based system that tries to find suitable (sequences of) behaviors for a certain scenario. The scripts consists of other scripts and behaviors, with alternative actions. These alternative actions are used to make the system more robust when getting unspecified, general or complex commands. This architecture also allows for failures (like a lower level behavior that is not able to reach its goal) to be handled.

Dialogue with the user is used to gather more information about the scenario, and about the preferences the user has. The system can also learn new scripts, by letting a user explain the steps that a certain complex task consists of, in terms of actions that the system already has scripts or behaviors for. The robot then learns how to execute that task by creating a new script for it.

3.4 Human detection, tracking and recognition

Human detection
The human detection module tries to find humans in front of the robot. The robot has several sensors which it can use to detect humans. In our approach we combine data received from 2 kinects or lasers. Both sensors are placed on different heights. The lowest one tries to detect legs and the top tries to detect bodies.

The data from the sensors is processed and splitted into segments using a jumping distance algorithm mentioned in [Premebida and Nunes, 2005]. With the received data we then try to match the observed legs with a body. This will result in a list with humans.

Human tracking and recognition
Humans are segmented through filtering a human sized 3-dimensional bounding box. This approach relies on the use of a camera with a depth sensor to form pointclouds, containing points consisting of RGB color data and 3D positions. Optionally, to get a more complete picture of the person standing in front of the robot, multiple cameras can be used to merge point clouds.

Using the generated point cloud of the segmented human, the height of this point cloud, and thus the height of the individual, is used as one of the features that are indicative of the human in the scene. Even though the extracted point cloud is not always one hundred percent accurate, the height of this cloud is an effective method to discriminate one individual from another.

Next to the height, other extracted features are the colors of the individuals upper body and legs. The color information is converted to the HSV space to diminish lighting noise. Due to lack of sensor accuracy the colors are grouped into white, grey, black or a combination of four main colors that span the entire hue spectrum. The percentage of points belonging to a specific color group represents one color feature.

The color and height features are combined to form a total of 15 features and are stored as the properties of a specific person (the operator) during a training procedure. The system can then classify any human as either the operator or a non-operator. As this system is used in situations with time constraints, it is important to memorize and classify new individuals within a limited time frame. The classification method used for this task is the k-nearest neighbors (k-NN) algorithm.

3.5 Learning to follow a person using reinforcement learning

The aim of this project is to have a robot learn to follow a person rather than hard coding such a behavior. In our setup, the location of the person in the field of view of the robot and the distance to the person constitute the state space of the reinforcement learning problem. Using Q-Learning [**?**] the agent should find a mapping from these states to suitable acceleration speeds and turning angles. We will compare different grades of discretization for both state and action space to find the settings that will yield a good performance after a feasible learning time.

3.6 Manipulation

For manipulation we are using the Mico arm from Kinova Robotics. We have implemented a simple grasping method using Cartesian positioning information. The arm is able to pick up objects that are detected with the Kinect. Currently the grasping method does not take into account obstacle detection and avoidance, neither does it know about self collision with other parts of the robot. Therefore the arm needs to start in a safe position from where it cannot collide with other parts of the robot.

The grasping part is currently very hard-coded, the hand will move to a point that is 10cm above the middle of the objects height and 10cm in front of the object. When it is in this position it will open it fingers, move down 10cm and move forwards 10cm, and the fingers will then close. There is no check to make sure if the object was successfully grabbed or not. If the arm cannot reach the object or if the object is too close, the arm will move back to a save position and will report back to the behaviour that it was unable to perform the grasping motion.

In future usage of the arm we are planning on using the MoveIt package. MoveIt uses a 3D model of the robot to plan a path for the arm, making sure that the arm does not collide with other parts of the robot. Also MoveIt is able to get information from sensors, like a Kinect, to create a 3D model of the environment that will prevent collision with obstacles other than the robot.

Currently the firmware of the arm is preventing a correct execution of the path that is calculated by MoveIt, therefore the arm is not able to safely move along the path as planned.

3.7 Human machine interaction

Speech recognition
Audition contributes to both perception and interaction in agents. Robots should have a hearing capability equivalent to ours to realize human-robot communication and social interaction, when they are expected to help us in a daily environment. Robots therefore should be able to recognize speech under various acoustic conditions.

Our setup comprises of a microphone array (from the Microsoft Kinect) and three modules that work together in order to robustly detect and recognize speech in various acoustic conditions.

Hark
The open source software audio processing package Hark[8], developed at the

[8] Hark can be found at `http://hark.jp`

Honda Japan Research Institute, can separate sound sources when using multiple microphones. Using the Blind Source Separation algorithm, Hark is able to filter and track sound sources to be detected later, if configured accordingly.

PocketSphinx
For ASR, we stream the detected sound sources to a python PocketSphinx[9] listener, developed at Carnegie Mellon University, using the general American English Language Model and custom closed grammars. In our module we switch between grammars for each behavior, making sure that the robot only recognizes what it needs to hear at that moment.

Sound Scape Estimator
Hark is a stable framework for silent environments, but needs tuning for specific sound environments or sound scapes. The Sound Scape Estimator characterizes a sonic environment and adjusts Hark accordingly. The SSE is trained on various acoustic environments through reinforcement learning on word recognition rate at the end of the audio processing pipeline.

This setup enables our robot to robustly recognize speech (87% word recognition rate) up to 3 meters in silence and distinguish sound sources on a 5 degree azimuth resolution. It also allows for simultaneous mutli speech recognition, though word recognition rates drop considerably in loud environments.

3.8 Kinect and Xtion Pro Sensor Systems

The depthsensor in the kinect and Xtion Pro sensor systems are used to gather depth information from a scene that enables the robot to gather more information from its surroundings than would be possible using only RGB cameras.

The depth information will be used for segmentation and 3D scene reconstruction for the use of navigation, thereby enabling obstacle avoidance and navigation using one of the freely available SLAM implementations provided by `OpenSLAM.org`.

Other uses include human-computer interaction (HCI), gesture recognition and training by example. The Kinect will also aid in the recognition of persons by their posture.

The libfreenect and OpenNI drivers are used to incorporate these sensors in our current Python software architecture.

4 Relevance

Our approach of combining a Nao and Pioneer to a new robot is easily reproducible, as it uses robots that are typically used for teaching purposes at universities.

[9] `http://cmusphinx.sourceforge.net`

Our setup allows to build a robot that exceeds the capabilities of both Nao and Pioneer alone without having to build a new robot from scratch. Also with the Nao and Pioneer being widely used, both come with libraries (and other conveniences) which allows us to focus on the process of developing novel functionalities and behaviors.

Combining the Nao and the Pioneer integrates the best features of each robot into a more robust "new robot", which allows us to use the Nao's more fine-tuned movement to grab objects (and also gives the Nao a range of operation which is better suited to real world applications), while still being able to provide a speedy locomotion. The additional HD cameras and the two Kinect cameras mounted at a height of approximately 180 cm allow for a good overview over real world scenery.

Apart from the hardware, our software architecture will make it possible for the robot to adapt its behavior to various environments, making it well suitable for applications outside ideal lab environments.

5 Conclusion

While still in the early stages of development, preliminary testing indicates that our platform can perform quite well in most, if not all, of the @Home challenges.

The BORG team aims to explore the realm of general purpose service robotics for ongoing research work in human-robot interaction, computer vision, machine learning and control methods in "real", unconstrained environments. The RoboCup@Home challenges are a fantastic benchmark to test these fields and the scientific validity of the systems presented above to cope with the challenges.

We hope that our main topics of investigation will bring fresh ideas and innovation into the world of service robotics.