OBJECT RECOGNITION USING DIALOGUES AND SEMANTIC ANCHORING

Vamsi Krishna Bandaru and Balasubramanian Rajasekaran

Computer Engineering Programme, 30 higher education credits

Örebro, Sweden, Spring 2013

Examiner: Amy Loutfi
Abstract

This report explains in detail the implemented system containing a robot and a sensor network that is deployed in a test apartment in an elderly residence area. The report focuses on the creation and maintenance (anchoring) of the connection between the semantic information present in the dialog with perceived actual physical objects in the home. Semantic knowledge about concepts and their correlations are retrieved from online resources and ontologies, e.g. Word-Net and sensors information are provided by cameras distributed in the apartment.
Preface

We want to express our gratitude to Pugazhenthhi, John Bosco and SASTRA University for giving us the opportunity to realize this project. We would also like to thank Amy Loutfi for being our supervisor at Örebro University and Andreas Persson for helping us to understand certain key concepts and helping us throughout.
## CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER NO</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Abstract........................................</td>
<td>ii</td>
</tr>
<tr>
<td></td>
<td>Preface..........................................</td>
<td>iii</td>
</tr>
<tr>
<td>1</td>
<td><strong>Introduction</strong>...............................</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1.1 Background..................................</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1.2 Project......................................</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1.3 Objective...................................</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1.4 Requirements................................</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td><strong>The Mobile Robot</strong>............................</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2.1 The hardware of the robot................</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2.2 QBo’s Software..............................</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>2.2.1 Open QBo Distro..........................</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>2.2.2 QBo applications........................</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>2.2.3 QBo drivers..............................</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td><strong>Problem Analysis</strong>...........................</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>3.1 QBo’s SURF based object recognition.....</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>3.1.1 Calibrating QBo’s cameras...............</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>3.1.2 Learning objects........................</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>3.1.3 Recognizing objects.....................</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>3.1.4 Problems in implementing distributed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>object recognition in QBo....................</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>3.2 SIFT based distributed object recognition.</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>3.2.1 SIFT......................................</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>3.2.2 SURF......................................</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>3.2.3 Implementation..........................</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>3.2.4 Shortcomings of SIFT based on distributed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>object recognition............................</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>3.3 Object recognition(SIFT) with dialogues...</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>3.3.1 Dialogue to resolve ambiguity...........</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>3.3.2 Limitations................................</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td><strong>Dialogues with Perceptual Anchoring</strong>.....</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>4.1 Dialogue system............................</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>4.1.1 Google’s API.............................</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>4.1.2 Stanford’s Parser.......................</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>4.1.3 Festival text to Speech..................</td>
<td>12</td>
</tr>
<tr>
<td>Section</td>
<td>Subsection</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
<td>------</td>
</tr>
<tr>
<td>4.2</td>
<td>Intelligence System</td>
<td>13</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Perceptual Anchoring</td>
<td>13</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Ontology</td>
<td>13</td>
</tr>
<tr>
<td>4.2.3</td>
<td>MongoDB</td>
<td>14</td>
</tr>
<tr>
<td>4.3</td>
<td>Implementation</td>
<td>15</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Overview of the dialogue process</td>
<td>15</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Setting up the initial reference database</td>
<td>15</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Searching the ontology</td>
<td>15</td>
</tr>
<tr>
<td>4.3.4</td>
<td>Cost or distance</td>
<td>16</td>
</tr>
</tbody>
</table>

5 Results | 18

6 Conclusion | 20

7 References | 21
1 Introduction

1.1 Background

Robots in combination with smart home environments are increasingly adopted for providing support to elderly and persons with disabilities [1]. The possibility to use ubiquitous robotic systems presents a number of interesting application areas for socially assistive robots that aim to improve quality of life and to allow people to live independently in their own home longer. Such technological solutions can be used to monitor the health conditions via physiological sensors and activities recognition, to remind about medicines and appointments, to raise alarms, and to assist in everyday tasks like finding objects, detect if objects are misplaced, guiding in the execution of tasks.

An important fact to many of these applications is the ability to communicate about and interact with objects that are present in the home. An important challenge is therefore to connect the information provided via the dialogue with the sensor information gathered from the robot or other sensor networks. The sensor network is able to assist the robot in performing tasks which are otherwise difficult on its own e.g. localization, while the robot also serves as a useful point of interaction. Studies in Human robot interactions such as [2] have even shown that smart homes with sensor networks are more readily acceptable when a mobile robot is present as an interaction, i.e. receives and relays information to the inhabitants. This challenge has been called the anchoring problem where semantic information (e.g. general concepts and object names) needs to be connected to the sensor data that refers to the object (e.g. feature descriptors). This connection should be first established and then maintained in time. Anchoring has been initially defined in [3] and then extended to consider large knowledge base system like Cyc [4]. The specific challenge of anchoring objects in the context of social robotics in a domestic environment is both the large possibility/variety of objects that need to be anchored and the fact that each object can be referred to in a number of ways in a dialogue. To allow for the possibility to have a dynamic system without the constraints of the dialogue being defined a-priori, open sources of information such as the web can be used. Spoken dialogue is a (for humans) natural and effortless medium of communication, this together with the evolution of speech related on-line services, makes spoken dialogue a more and more prominent solution for human-robot interaction [5]. It has also been proven that a robot which is capable of interacting in natural language would be more convenient to use, even for users without technical experience [6].

1.2 Project

The project consisted of three separate parts, evaluation, implementation and evaluation of the implementation. In the evaluation stage, previous solutions were to be explored, their viability evaluated and any potential obstacles were to be identified. The implementation stage involved choosing one of the solutions and implementing it. In the final stage the performance of the implementation was to be evaluated.
1.3 Objective

The main aim of the project is to develop a system in a real home environment that can establish a dialogue with a human user about objects. The system consists of a mobile robot together with a set of distributed cameras in the home. The robot can accept requests for finding objects via spoken dialogue with a human user and use stored information provided by the cameras about object present in the environment to answer the requests.

1.4 Requirements

At the end of the project the following requirements were to be satisfied:

- The main requirement is to develop a system in a real home environment that can establish a dialogue with a human user about objects.
- The interaction must be using natural language, without the user having to remember a determined set of voice commands.
- The system must not take a significant amount of time to respond to the user’s queries. A delay of more than 5 seconds was considered unacceptable.
2 The Mobile Robot

The chosen platform for the project is Qbo by thecorpora. The main advantages of the platform are that both its hardware and software is open source and it has a community of followers that provide adequate support.

2.1 The Hardware of the Robot

The Q.bo robot needs to interact with its surroundings. Q.bo has some sensors and actuators in order to achieve this goal. The Q.boards have been designed to acquire the sensor data and to make it available for the PC that is embedded in Q.bo. The boards also give the PC the capability to control the actuators of Q.bo.

In order to control the sensors and motors that come with Q.bo, three boards have been designed. Their names are Q.board 1, Q.board 2 and Q.board 3. Q.bo has two additional boards whose names are Q.board 4 and Q.board 5 that serve as an IMU sensor and as the mouth LED matrix respectively.

A small list (non-exhaustive) of key features of QBO’s hardware is presented here.

Sensors:
1. Stereo webcams that forms QBO’s eyes – 2 units
2. Ultrasonic Range Finders (2 units)
3. Proximity Sensors
4. Asus’s Xtion Pro Live

PC Components
1. Intel DQ67EP Mini-ITX Motherboard
2. Intel Core i3-2120T Processor
3. 4GB Kingston DDR3 1333Mhz Non-ECC CL9 DIMM
4. 128Gb Crucial SSD SATA 2.5
5. Intel 6200 IEEE 802.11n Wi-Fi Adapter

Sound:
1. Monacor SP – 6/8 SQ Miniature Speaker
2. Fonestar Microphone 2220

Actuators:
1. EGM30 Motors that drive the robot wheels
2. GWS S125 Servo for head turns.

The fact that the hardware is open source means that the same platform can be used for future projects and extending the platform’s capabilities would be relatively easy. The Q.boards are Arduino compatible, which means that a host of hardware accessories that are designed to work with Arduino would also work with Qbo thereby enhancing its capabilities drastically.
2.2 QBO’s Software

Qbo’s Software can be broken down into three major categories, namely:

2.2.1 OpenQbo Distro:

A complete operating system based on Ubuntu. With this we pretend to have a complete base and a high level development system.

2.2.2 Qbo applications:

These are software applications to control the robot's behaviour. Some of them are implemented using the ROS Software Platform.

Robot Operating System (ROS) is an open-source middle-ware for Robot software development initially developed at Stanford AI laboratory, currently developed at Willow Garage. ROS provides the services one would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes and package management. It also provides tools and libraries for obtaining, building, writing and running code across multiple computers. It is based on a graph architecture where processing takes place in nodes that may receive, post and multiplex sensor, control, state, planning, actuator and other messages. The ROS runtime "graph" is a peer-to-peer network of processes that are loosely coupled using the ROS communication infrastructure. It implements several different styles of communication, including synchronous communication over Services, asynchronous streaming of data over Topics and storage of data on a Parameter Server. It has tools in-built for Simultaneous Localization And Mapping (SLAM) and 2-D map building, visualizing sensor data, robot simulation etc.

Other key applications/packages that come bundled with the distro, apart from ROS Software Platform are,

1. Speech Recognition Software (Julius)

Julius is a high-performance, two-pass large vocabulary continuous speech recognition (LVCSR) decoder software for speech-related researchers and developers. It can perform almost real-time decoding on most current PCs in 60k word dictation task using word 3-gram and context-dependent HMM. Major search techniques are fully incorporated. It is also modularized carefully to be independent from model structures, and various HMM types are supported such as shared-state triphones and tied-mixture models, with any number of mixtures, states, or phones. Standard formats are adopted to cope with other free modelling toolkit. The main platform is Linux and other Unix workstations, and also works on Windows. Julius is open source and distributed with a revised BSD style license.
2. Test to Speech processor (Festival)

It is a general multi-lingual speech synthesis system originally developed by Alan W. Black at Centre for Speech Technology Research (CSTR) at the University of Edinburgh. Substantial contributions have also been provided by Carnegie Mellon University and other sites. It is distributed under a free software license similar to the BSD License.

3. OpenCV library for Computer Vision

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision, developed by Intel, and now supported by Willow Garage and Itseez. It is free for use under the open source BSD license. The library is cross-platform. It focuses mainly on real-time image processing. If the library finds Intel's Integrated Performance Primitives on the system, it will use these proprietary optimized routines to accelerate itself.

4. OpenQbo package repository.

A repository maintained by the OpenQbo community, that contains user developed packages for specific applications.

2.2.3 Qbo drivers:

These are software drivers to interact with the Qbo boards or devices on the operating system level.


3 Problem Analysis

In this section a brief account of the various solutions that we tried to implement and their shortcomings are presented in chronological order

3.1 QBO’s SURF based object Recognition

The first few weeks of our project we focused on learning about QBO and how QBO’s OS is set up. Time was spent on successfully understanding the various nodes that were responsible for QBO performing object recognition. A short description of QBO’s object recognition that is based on SURF is given below.

3.1.1 Calibrating the QBO’s cameras

Initially QBO’s stereoscopic cameras calibrated using a chequered board. QBO stores the images of objects that will act as reference for future recognition tasks in a local folder, which is initially empty.

3.1.2 Learning objects

QBO takes images from left stereoscopic camera that will be used as the raw image and uses the stereoscopic data obtained from calibration to identify the nearest object. Draws a rectangle around the nearest object and it is known as region of interest. Until an object is detected QBO continuously keeps turning its head. On finding an object and thereby successfully drawing a region of interest, he was head finally settles in a singular position and an indication to the user is given with the help of a blue light arising from its nose. Once the user sees the visual cue, he can now ask QBO to either identify the object or learn the object. In order to make QBO learn an object the user has to reach out a voice command in the following format “QBO This is a X” or “This is an X”, where X is the name of the object. In order for QBO to understand X, the word X has to be first loaded into QBO’s dictionary.

On receiving the ‘learn’ command, QBO takes 20 images of the object that is covered by the region of interest and stores it in a folder that is named X. Now QBO will be in a position to recognise X. One can improve the accuracy of QBO’s object recognition by making QBO learn the object in various postures. This means that QBO will record 20 images for each posture learned. The above said procedure is repeated for each new object that QBO has to learn.

3.1.3 Recognizing objects

In order for QBO to recognise an object, the object first has to be within the vicinity of QBO’s eyes (stereo cameras). On receiving the visual cue, that is the blue light on QBO’s nose which indicates that QBO has spotted an object in front of its eyes, the user now can issue the object recognition command – “QBO what is this?” Or “what is this?”

On receiving the command, QBO immediately starts performing SURF between the region of interest obtained by cropping the live feed obtained from studios left stereoscopic camera using the stereoscopic calibration data and the various images stored inside folders whose names represent objects that QBO has previously learnt. Out of the scores of images present,
if a predefined number of images belonging to a particular folder (a folder represents an object) results in matches whose percentage value is about a predefined threshold, then the name of the folder is announced as what QBO perceives to be the object in question.

### 3.1.4 Problems in implementing distributed object recognition in QBO

1. QBO’s inbuilt object recognition is set up in such a way that it needs a large number of reference images of a particular object in order to make up for the deficiencies in the matching algorithm. This means that more often than 100 images (five orientations) of each object is necessary for QBO to identify the object with acceptable accuracy.

2. QBO’s stereo cameras are of very poor quality. It takes images of resolution 360 x 240 pixels and then cropping it to the region of interest means that, even smaller images are stored as reference images.

3. Large number of images means that the time taken to recognise an object increases.

4. Since QBO only compared images within the region of interest, this methodology would be ineffective if the live feed were to be obtained from a ceiling camera or a surveillance camera that is usually kept overhead. In QBO’s methodology both the reference image and the live feed’s region of interest would only carry the object and very little background information. Whereas, the images obtained from a surveillance camera, will predominantly have objects that are not of interest. In other words, objects of interest would occupy only a fraction of the entire image.

5. Stereo calibration data was a key component in QBO’s object recognition algorithm. Since ceiling cameras are only mono cameras, there was no stereo data to work with.
3.2 SIFT based distributed object recognition

Firstly, it was decided to continue to use QBO’s method of learning (storing reference images) as it would be much convenient for the end user to issue voice commands and make QBO learn rather than to manually placed images in a specified folder in QBO. Hence our aim was to use an image recognition technique which would use QBO’s reference images to identify an object in an image taken from a ceiling camera.

3.2.1 SIFT

The Scale Invariant Feature Transform [1] is a method to detect distinctive, invariant image feature points, which easily can be matched between images to perform tasks such as object detection and recognition, or to compute geometrical transformations between images. Matching SIFT features between images can be performed using kd-trees. SIFT has been proven to be the most robust local invariant feature descriptor. SIFT isn’t just scale invariant. It gives good results for changes in the following parameters, namely

- Scaling
- Rotation
- Illumination
- Viewpoint

SIFT matching is performed between the features of the images from the cameras and the features from the reference images in the image database. Based on the matching, the presence of the object can be determined and based on the camera from which the image was taken the location of the object can be determined.

3.2.2 SURF

SURF (Speeded Up Robust Features) is a robust local feature detector, first presented by Herbert Bay et al. in 2006, that can be used in computer vision tasks like object recognition or 3D reconstruction. It is partly inspired by the SIFT descriptor. SURF is based on sums of 2D Haar wavelet responses and makes an efficient use of integral images.

It uses an integer approximation to the determinant of Hessian blob detector, which can be computed extremely quickly with an integral image (3 integer operations). For features, it uses the sum of the Haar wavelet response around the point of interest. Again, these can be computed with the aid of the integral image.

3.2.3 Implementation

Experiments showed that SIFT gave better results when compared to SURF for this particular application. SIFT algorithm along with KD tree matching (algorithm to match the key points) was used. The original program was written by Rob W Hess and was named Open SIFT. The source code modified so that, the various ceiling cameras can be accessed, images could be obtained from each of them, SIFT algorithm and KD tree matching were performed and finally the result was announced to the user.
3.2.4 Shortcomings of sift based distributed object recognition

1. Limited vocabulary. The object names have to be fed into the dictionary beforehand. Otherwise, the Julius speech engine would fail to recognise a particular word.

2. If the reference image and the live stream image where larger (more pixels) then the results would have been even more accurate.

3. The accuracy of the recognition depended significantly on the quality of reference images. QBO makes a mistake or happens to draw a larger area of interest, say for instance if the region of interest also accommodates the user’s hands/fingers this additional information distorts the sift algorithm to work properly. Even though the inherent nature of this would actually overcome this one has to remember that the size of the reference image is not more than 150 X 200 pixels in most cases.

4. There was no way to resolve ambiguity. In other words if two objects had similar features, then it would be very difficult for the user to know which object matches his requirements without actually visiting the places in which the objects were found.

5. The time taken for QBO to complete the entire process rights receiving the users command till the point it gives out the required response, it took more than 10 seconds just when there were only five or six objects that QBO had learned.

6. There is no guarantee that the first 20 images that QBO takes a particular object will be sufficient for it to recognise objects at various illumination levels. Even randomly picking, images within the events course of images for each object would not guarantee the same level of accuracy as to combating all the objects that have been recorded as reference images.
3.3 Object Recognition (SIFT) with Dialogues

In order to overcome the various problems faced by SIFT based distributed object recognition, we were told to work in tandem with Andreas. Andreas was working on an algorithm that would be able to identify objects with sufficient accuracy in an image taken from a ceiling camera. The reference database had thousands of objects. This means that his algorithm at the potential to identify more than a few thousand objects.

We were asked to develop a dialogue between QBO and the user that would help the user to resolve an ambiguity that might arise when using Andreas’ algorithm to find objects.

3.3.1 Dialogue to resolve ambiguity

There are two broad categories based on which the ambiguity could be resolved. They were the colour of the object and the location where it was present.

In a couple of weeks we were able to develop a program that was able to help the user to narrow down the search results and find out the exact location of the object that he was looking for. The dialogue was also built in such a way, that the user to communicate in a natural way with the robot as he would do with the human. For instance, if the user knew that the object was looking for was red in colour, then he could directly ask “where are the red instances of the object found?”

3.3.2 Limitations of the above mentioned dialogue system

1. There was a particular set of sentences that would work in QBO. If the user wish to ask the question in a different manner or if he framed the question in the manner to which QBO was unfamiliar (which means the Julius speech engine’s config file did not carry such a syntax) when QBO would fail to recognise the question

2. Also as previously stated, Julius speech engine’s vocabulary (vocabulary inside the config file) is initially very limited and has to be updated if the user wants to use new words while talking to QBO. This applies to even object names.

3. The user has no option but to ask for specific objects. He cannot use more common names like for instance, he cannot ask “can you find me something to eat?”, for such a question does not carry the name of the object.
4 Dialogues with Perceptual Anchoring

The probable solution to the problem at hand required a dialogue method where the user can speak in natural language with the robot (QBO) and that QBO had the intelligence to understand the user’s requests and has the knowledge and capability to effectively accomplish the user’s requests. Thus a system that made use of tools like Google’s speech to text API, Stanford’s Parser[22] (a product of Stanford’s core NLP suite), a high level AIT concept namely, Perceptual Anchoring and a database management system based on MongoDB.

![Figure 4.1 Framework](image)

### 4.1 Dialogue system

With the evolution of speech related services, it is imperative that speech will be the medium of communication with robots in the future too, as it makes robot human interaction as natural and as smooth as interaction between humans[11]. In such a scenario, asking robots to find objects (object recognition) would be a common task that one may perform repeatedly [12]. If the robot can understand natural language and reply back to the user, it would be more convenient to use, for even users who have not been exposed much to technology [16].

#### 4.1.1 Google’s API

Google provides a speech to text API that is targeted for Chrome and Android users. However, the API does work for custom programs that are not built for Chrome or Android. An independent developer had created IRIS, a voice assistant similar to that of Apple’s SIRI using Google’s API. The main advantage of using Google’s API is that the processing is done in Google’s servers. In other words, a piece of recoded speech is sent to the Google’s server and then with a very short delay, a response containing the text equivalent of the recorded
speech is received. The other advantage is that there is no limitation on the vocabulary, as long as the language is English.

4.1.2 Stanford’s Parser

A natural language parser is a program that works out the grammatical structure of sentences, for instance, which groups of words go together (as "phrases") and which words are the subject or object of a verb. Stanford’s Parser[22] package is a Java implementation of probabilistic natural language parsers, both highly optimized PCFG and lexicalized dependency parsers, and a lexicalized PCFG parser. The original version of this parser was mainly written by Dan Klein, with support code and linguistic grammar development by Christopher Manning.

4.1.3 Festival text to speech

The Festival Speech Synthesis System is a general multi-lingual speech synthesis system originally developed by Alan W. Black at Centre for Speech Technology Research (CSTR) at the University of Edinburgh. Substantial contributions have also been provided by Carnegie Mellon University and other sites. It is distributed under a free software license similar to the BSD License.

It offers a full text to speech system with various APIs, as well as an environment for development and research of speech synthesis techniques. It is written in C++ with a Scheme-like command interpreter for general customization and extension.
4.2 Intelligence System

The intelligence system consists of three core components namely the Perceptual Anchor creating system, Ontology, MongoDB a database management system.

4.2.1 Perceptual Anchoring

Anchoring [8, 9, 10] is the process of creating and maintaining the correspondence between symbols and percepts that refer to the same physical objects. With the increasing abilities of mobile robots, anchoring is becoming a major research issue to make robots perform their task in unstructured and partially unknown environments. The environment tends to change with time. Objects may be removed or added to the environment (apartment rooms). The agents or robots should be aware of these changes. Hence, an Anchor space is maintained. Anchors are physical objects that are present in the apartments at that very instant of time. The Anchors are created and updated periodically in the Anchor space. The anchor model includes image features (symbols) and semantics (percepts) that refer to the same physical object.

Figure 4.2 Anchoring

4.2.2 Ontology

To support the sharing and reuse of formally represented knowledge among AI systems, it is useful to define the common vocabulary in which shared knowledge is represented. A specification of a representational vocabulary for a shared domain of discourse — definitions of classes, relations, functions, and other objects — is called an ontology. In the context of
In computer and information sciences, an ontology defines a set of representational primitives with which to model a domain of knowledge or discourse. The representational primitives are typically classes (or sets), attributes (or properties), and relationships (or relations among class members). The definitions of the representational primitives include information about their meaning and constraints on their logically consistent application. In the context of database systems, ontology can be viewed as a level of abstraction of data models, analogous to hierarchical and relational models, but intended for modelling knowledge about individuals, their attributes, and their relationships to other individuals. Ontologies are typically specified in languages that allow abstraction away from data structures and implementation strategies; in practice, the languages of ontologies are closer in expressive power to first-order logic than languages used to model databases. For this reason, ontologies are said to be at the "semantic" level, whereas database schema are models of data at the "logical" or "physical" level. Due to their independence from lower level data models, ontologies are used for integrating heterogeneous databases, enabling interoperability among disparate systems, and specifying interfaces to independent, knowledge-based services. In the technology stack of the Semantic Web standards [15], ontologies are called out as an explicit layer. The ontology consists of semantic relationships such as synonyms, hyponyms and hypernyms[14].

Figure 4.3
4.2.3 MongoDB

MongoDB is a cross-platform document-oriented database system. Classified as a "NoSQL" database, MongoDB eschews the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas (MongoDB calls the format BSON), making the integration of data in certain types of applications easier and faster. Released under a combination of the GNU Affero General Public License and the Apache License, MongoDB is free and open source software.

4.3 Implementation

4.3.1 Overview of the Dialogue Process

The overview of the process can be seen in the flow chart (fig 1). The user can initiate the search using a voice command. The voice will be recorded and sent to Google servers to be converted to text. The resultant text is then parsed using state of the art Stanford parser. The result of the parser is then processed by the agent. Here, the robot tries to understand the meaning and context of the user’s request. Once the agent confirms that the user has asked for something to be searched, then it accesses the most recent updated anchor database form the central server. Then, depending on the user’s request, the agent searches for possible candidates in the database. If a direct match for the request made by the user is unavailable then the agent uses a semantic ontology to find out possible candidates.

Semantic descriptions of the objects are fairly specific. For instance, pasta and sandwich are some of the semantics used. If the user asks the robot to search for some food, then the robot will be able to map food items like pasta to the word food, using the semantic ontology. This increases the scope of the dialogue that the user can have with the robot as the user is not limited to the vocabulary that he can use. Since this approach also includes a parser, the type of sentences that the user can use is not limited either.

4.3.2 SETTING UP THE INITIAL REFERENCE DATABASE

In order to create the Anchor database, reference database is needed. An online semantic repository named Wordnet is used based on which manually an offline database is created. A program has been developed that would create a database containing all the semantics that have the word food as its hyponym directly or indirectly. In other words, all the hyponyms of food are explored recursively till there are no more nodes to be added to the database. The ontology database is represented as shown in Table 1.

4.3.3 SEARCHING THE ONTOLOGY

The ontology is in the form of search trees in a finite search space. The trees are represented in the form of tables. The relation between the various nodes is in the form of parent child relation. A field in the table contains a sequential list of all the nodes that has to be traversed in order to reach a given node of interest from the top of the tree. MongoDB is the preferred database structure.
<table>
<thead>
<tr>
<th>Node No.</th>
<th>Semantic</th>
<th>Parents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Eat</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>Food</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Pasta</td>
<td>0, 1</td>
</tr>
<tr>
<td>3</td>
<td>Baked food</td>
<td>0, 1</td>
</tr>
<tr>
<td>4</td>
<td>Diary food</td>
<td>0, 1</td>
</tr>
<tr>
<td>5</td>
<td>Spaghetti</td>
<td>0, 1, 2</td>
</tr>
<tr>
<td>6</td>
<td>Macroni</td>
<td>0, 1, 2</td>
</tr>
<tr>
<td>7</td>
<td>Tortillini</td>
<td>0, 1, 2</td>
</tr>
<tr>
<td>8</td>
<td>Biscuit</td>
<td>0, 1, 3</td>
</tr>
<tr>
<td>9</td>
<td>Cake</td>
<td>0, 1, 3</td>
</tr>
<tr>
<td>10</td>
<td>Butter</td>
<td>0, 1, 4</td>
</tr>
<tr>
<td>11</td>
<td>Cheese</td>
<td>0, 1, 4</td>
</tr>
<tr>
<td>12</td>
<td>Sponge cake</td>
<td>0, 1, 3, 9</td>
</tr>
<tr>
<td>13</td>
<td>Chocolate cake</td>
<td>0, 1, 3, 9</td>
</tr>
<tr>
<td>14</td>
<td>Coconut cake</td>
<td>0, 1, 3, 9</td>
</tr>
<tr>
<td>15</td>
<td>Honey cake</td>
<td>0, 1, 3, 9</td>
</tr>
<tr>
<td>16</td>
<td>Cheese cake</td>
<td>0, 1, 3, 9</td>
</tr>
<tr>
<td>17</td>
<td>Yak butter</td>
<td>0, 1, 4, 10</td>
</tr>
<tr>
<td>18</td>
<td>Brown butter</td>
<td>0, 1, 4, 10</td>
</tr>
<tr>
<td>19</td>
<td>Soft cheese</td>
<td>0, 1, 4, 11</td>
</tr>
<tr>
<td>20</td>
<td>Cheddar</td>
<td>0, 1, 4, 11</td>
</tr>
<tr>
<td>.......</td>
<td>...........</td>
<td>...........</td>
</tr>
<tr>
<td>.......</td>
<td>...........</td>
<td>...........</td>
</tr>
<tr>
<td>63</td>
<td>Chilli cheese</td>
<td>0, 1, 4,11 , 19, 29, 47</td>
</tr>
</tbody>
</table>

Table 4.1
4.3.4 COST OR DISTANCE

The number of elements in the parents field for each node (semantic) provides the cost or distance of each node from the starting node. The cost or distance from any random node (semantic) will be given by the number of elements after that random node's number.

Example:

In Table 1, the cost or distance of node 'Butter' from 'Eat' is given by the number of elements in the corresponding parent field, which is equal to 3.

Similarly, the cost from 'Food' to 'Cheese' is given by the number of elements after the node number of 'Food' i.e '0' So, the cost or distance is 2.

The cost is represented as shown in Fig 4.4.
5 Results

The above described setup was implemented at Ängen research apartment. A few scenarios depicting the results obtained by implementing the system and testing it is described below.

As described in the above process, the user interface to the robot consists of a speech based application, where the user can speak sentences in simple English. The voice is converted into sentence using Google Speech Service. The sentence is analysed by a recursive descent parser and translated into a symbolic description.

The grammar defined allows commands of the form “find ...” followed by a description of the object. The description consists of a main part and can be followed by sub-clauses describing objects that are spatially related to that object. The main part and each of the sub-clauses can be either a definite or indefinite description, indicated by the article “a” or “the”, and includes the object’s class, for example “cup”, and optionally its colour. The colour of an object is inferred from the clause “with ...” following the object’s class. The derived symbolic description is used to construct a query for the KB. The main functionality is realised by a FIND-ANCHOR routine, which collects candidates from the KB that match the description.

Example Run:

Having a scenario in mind where the user asks Q.bo to find some object. The camera identified the objects and they were correctly anchored with the respective information created in the KB.

The following objects were placed in various rooms in the apartment at Ängen.

- Biscuit pack - 3
- Orange Juice -1
- Wine bottle -1
- Pasta -1
- Chips -1

Example 1: (Object specific request)

User – U ; Robot – R

U: “Find me some biscuit.”

In a first step a query for "biscuits" returned the three candidates from the search space, and there is an ambiguity as both the objects were biscuits.

R: “Three instances of biscuit are found.”
U: “Where are they?”
R: "Of the Three instances, one is in the Kitchen, one is in the Dining room and one is in the Living room.”
U: “What are they?”
R: "Two Red biscuit, One Blue biscuit."  
U: "Where is the Blue biscuit?"
Example 2: (Category specific request)

U: “Find some beverage.”

As mentioned earlier, the Search space consists of anchors and their hypernyms from the ontology space. The query for "beverage" returned the two candidates,

R: “Two instances of beverage are found.”
U: “What are they?”
R: "One instance of Wine, One instance of Orange Juice."
U: “Where is the Orange Juice?”
R: "Orange Juice is in the Kitchen."

Example 3: (Purpose specific request)

U: Find me something to crunch.”

As mentioned earlier, the Search space consists of anchors and their hypernyms from the ontology space. In a first step a query for "crunch" returned the root verb "eat", and in the second step the query for "eat" returned five candidates.

R: “Five instances are found.”
U: “What are they?”
R: "Three instances of Biscuits, One instance of Pasta, One instance of Chips."
U: “Where is the Chips?”
R: "Chips is in the Kitchen."
6 Conclusion

There are great expectations that technology will contribute to an improved quality of life of elderly and persons with disability allowing them to live independently in their own homes. However this will only be possible if robust and flexible systems are developed that are suitable for supporting everyday tasks.

A key capability of such systems is the possibility to speak about physical objects present in the home using a natural vocabulary. In this project we have presented a distributed perceptual anchoring framework in a smart home where an ontology can be extended based on requests given by human users and through crawling on-line repositories. This provides for not only an enriched dialogue, but also enhances the semantic symbols anchored to perceptual data in a anchoring system.

The overall framework presented here was trained a-priori and hence the system had some basic knowledge first hand with which it tacked the problem. It would be a more complex and challenging problem to involve the user in a dialogue in order to learn perceived objects.

Another aspect that need further attention is the user experience and what would be acceptable delays in a natural language dialogue. Execution time of ‘crawling’ an on-line repository is highly dependent on recursiveness in the search and bandwidth. In the framework presented in this paper, results of crawling are stored in a knowledge base for faster off-line retrieval, which on the other hand consumes disk storage space.
REFERENCES


[9] Silvia Coradeschi and Alessandro Saffiotti, "Perceptual Anchoring of Symbols for Action"

[10] Daoutis, Marios (School of Science and Technology, Örebro University, Sweden) "Knowledge based perceptual anchoring: grounding percepts to concepts in cognitive robots", Doctoral thesis.(2013)


[14] E. Mena, V. Kashyap, A. Illarramendi, A. Sheth, "Domain Specific Ontologies for Semantic Information Brokering on the Global Information Infrastructure"


[19] www.ros.org

[20] www.answers.ros.org

[21] www.openqbo.org

[22] nlp.stanford.edu/software/lex-parser.shtml