PeopleBot

Intelligent Robotics – I Project Report

BY

RASHMI DUBEY

PORTLAND STATE UNIVERSITY

Guidance

Prof. Marek A. Perkowski

FALL 2010

Preface

Intelligent Robotics, an engineering discipline, is the amalgamation of Artificial Intelligence and Robotics. It is concerned with providing automatic agent (a Robot) the ability to perform complex tasks with human-like intelligence. Intelligent Robotics is growing by leaps and bounds greatly benefitted by the advent of faster processors and more sophisticated machinery. There are many exciting applications including underwater and space exploration, remote surgery, research, etc.

PeopleBot or Pioneer-2 is research robot. It is differential-drive robot for service and human-robot interaction (HRI) projects. It can be used for object and people recognition and tracking, or other robot vision tasks. It runs best on hard surfaces. It can traverse low sills and household power cords and climb most wheelchair ramps.

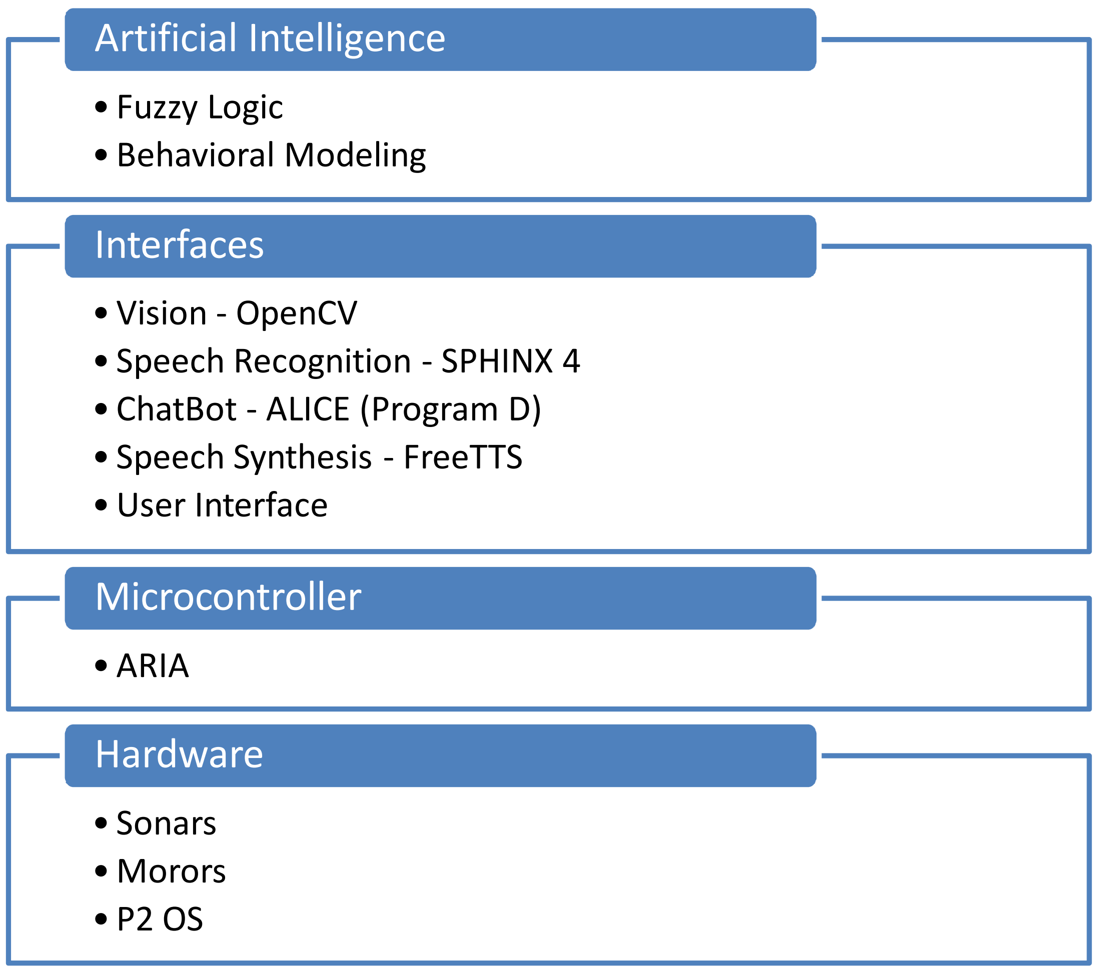
The goal of this project was to resurrect PeopleBot and incorporate it with cognitive abilities. Bringing the robot back to operational state involved doing mechanical assembly and modification of the original software. A laptop is attached to the robot body which provides it with cognitive features like Vision, Speech Recognition and Synthesis, Chatting, and Decision making. The robot is programmed such that it can be controlled both locally and remotely.

My contribution to the project was in designing and developing the core Artificial Intelligence module. This multi-threaded module is implemented in JAVA programming language. The work involved integrating speech recognition and synthesis, and chatting interfaces in the robot. A UI was developed for accessing the robot and controlling its various features. Fuzzy Logic is implemented to impart human-like decision making faculty. To integrate AI module with modules developed by other team members, multi-threaded sockets were developed. A review was carried out for the softwares available for Natural Language Processing:

- SPHINX 4 from Carnegie Mellon University (CMU) was selected for speech recognition.
  Two speech recognizers (one using JSGF Grammar and another one using N-Gram Grammar) were configured to run on same machine.
- FreeTTS from SUN Microsystems is used for speech synthesis.
- A.L.I.C.E. - Program D version is integrated for chatting.
  A comprehensive review of the software was carried out. The original software was modified to provide learning ability to the robot, a feature only available in paid versions of ALICE - Program Z. There are some bugs in the online version of the software which were fixed.

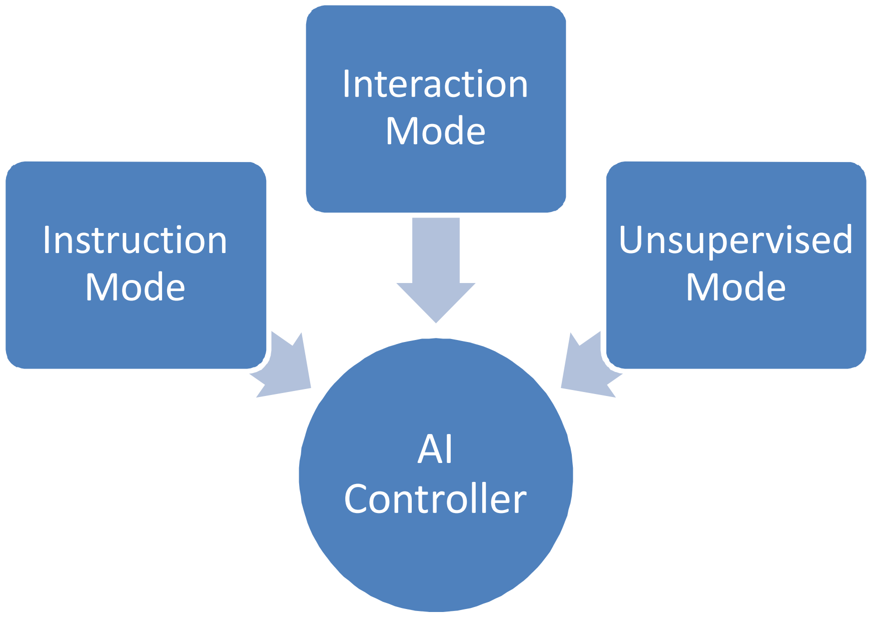
Contents

PeopleBot Architecture



This diagram shows the architecture of PeopleBot. The softwares used and the AI design are explained in details in later sections.

Behavioral Modeling



PeopleBot is designed as a Finite State Machine and as the above diagram shows, it has an AI Controller (a start state) and 3 stable states:

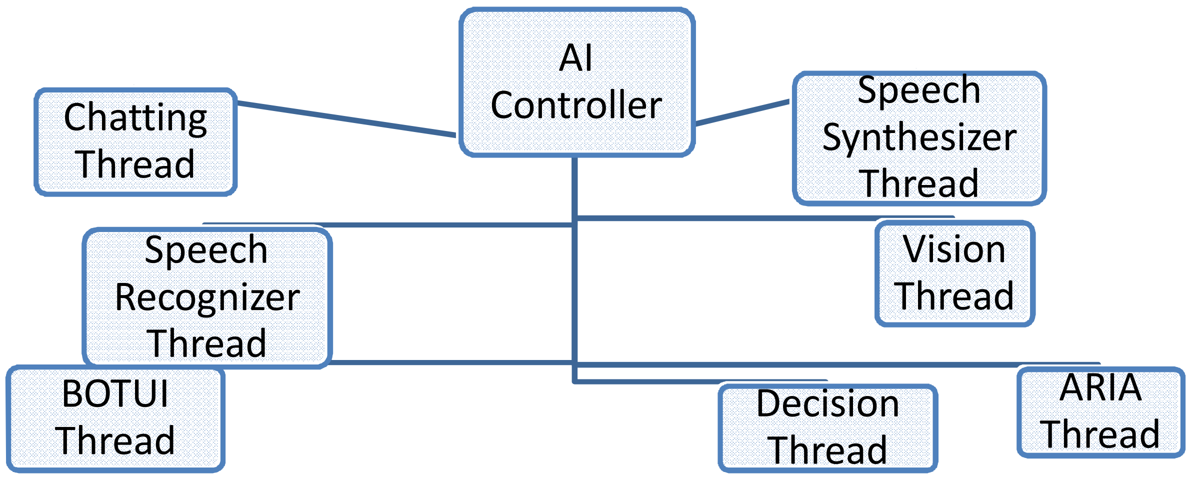
AI Controller – This can be considered as Start state. PeopleBot does not remain in this state for long. This state is to ensure that all the hardware and software parts of the robot are working fine. Once everything is ascertained, AI software kicks in and robot is moved to one of the following three stable states.

- Unsupervised Mode – This state is not supervised. The robot will wander on its own in the space avoiding obstacles.
- Instruction Mode – This is administrator controlled state. The robot moves only as directed.
- Interaction Mode – In this state, the robot stops moving and starts interacting with people.

Switching between states can be achieved by many means. The Artificial Intelligence part (Fuzzy Logic) programmatically takes the decision of the robot state and does the switching if required. Speech and User interfaces are provided to explicitly switch states. Instruction Mode is controlled by an administrator and hence switching to this state requires necessary login credentials.

AI Software Design

The main Artificial Intelligence module is designed in JAVA programming language. This module is exhaustively multi-threaded. The following diagram presents the overall picture of the module:



All the threads communicate with their respective applications via threaded sockets.

- ARIA Thread runs Server Socket to communicate with ARIA software.
- BOTUI Thread provides user interface for accessing PeopleBot and speechless interaction.
- Chatting Thread runs threaded Program D and communicate, via Server Sockets, with another JVM process running n-gram speech recognizer.
- Decision Thread starts with the AI Controller and remains active in the program's lifecycle. It does not directly communicate with any program or thread but uses data fetched by other threads for intelligent decision making.
- Speech Recognizer Thread runs speech recognizer program created using sphinx-4 libraries and assigns different JSGF grammar to the recognizer based on state of the robot.
- Speech Synthesizer Thread runs speech synthesizer program created using freeTTS libraries.
- Vision Thread runs Server Socket which listens to Open CV software on same or different machine.

Natural Language Processing (NLP)

Natural Language Processing is best described as intersection of Artificial Intelligence and Computational Linguistics. It plays an important role in interactions between computers and humans through natural language. NLP uses various computational techniques to formalize elements of human language (linguist analysis) for the sole purpose of achieving human like language processing skill.

Many approaches have been proposed for Natural Language Processing from time to time, but the recent approach of using machine learning (pattern recognition) is the most promising one. The basis of this confidence is in the learning algorithms used which make systems automatically learn rules through the analysis of real-world training data. Hence such a system has better ability to deal with unseen data than traditional language processing techniques.

Natural Language Processing provides both theory and implementations for a wide range of applications like 'Text Summarization', 'Text-to-Speech', 'Speech Recognition', 'Optical character recognition', 'Human-Computer Interaction', etc.

PeopleBot uses NLP for its speaking, listening and chatting features. Many types of softwares are available for implementing these sensory inputs. A review was carried out for the softwares available for Natural Language Processing:

- SPHINX 4 from Carnegie Mellon University (CMU) was selected for speech recognition.
- FreeTTS from SUN Microsystems is used for speech synthesis.
- A.L.I.C.E. - Program D version is integrated for chatting.

The choice of software is based on its availability (Open Source) and ease of adaptability for experimentation to aid learning. The respective software was modified in some cases to better suit the project. Tailoring the software for the project has a two-fold benefit. Firstly, the software is customized for the project. Secondly, it provided an opportunity to gain knowledge of NLP in the short time frame of a quarter.

The following three sections give a brief overview of the NLP softwares used, how they are implemented in the project and the customization done, if any. For more details about the original software, see Reference Section.

ChatBot: A.L.I.C.E. (Program D)

What is a ChatBot?

Natural Language Processing's most fundamental application is ChatBot. ChatBot is a machine designed to accomplish Turing test, i.e., converse intelligently with humans via auditory or textual methods. In a nutshell, chat bots mimics human conversation. They are based on the recognition of keywords or phrases given in input and on a set of corresponding pre-arranged and pre-programmed answers in output, so that the conversation could be considered intelligent. Chat bots are used exhaustively in commercial applications like entertainment, education, information interfaces and more.

ALICE (Program D) - Used in PeopleBot

The classical ELIZA, written by Joseph Weizenbaum, is the pioneer amongst chatter bots. ELIZA inspired creation of many chat bots of which A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) or Alice, designed by Richard Wallace, has been the most successful one. It has been awarded the coveted Loebner Prize three times. The success of Alice over other chat bots lies in its heuristic conversation rules. Instead of searching a mammoth database, Alice uses simple AIML (Artificial Intelligence Markup Language, an extension of XML) files.
In PeopleBot Program D version of Alice, written in JAVA, is used. It is an open source software.

How AIML works?

The basic unit of the knowledge-base (KB) of a chatbot conversational agent (AIML) is 'category'. It is made of a pair of pattern and template, which can be linked together semantically and/or recursively by means of srai connections. The pattern is a string representing user question and the template is string representing chat bot response.

Simple pattern matching example:

```
<category>
        <pattern> HOW ARE YOU DOING</pattern>
        <template> I am doing perfectly fine. Thank you.</template>
</category>
```

To achieve human like response, several AIML objects are used; the frequently used ones are 'that', 'think', 'random', 'star' and 'topic'. When aiml files are loaded, all pattern strings along with 'that' and 'topic' strings are stored as keys in a HashMap variable. The corresponding template is stored as value of the key. A unique processor is assigned for each of the AIML objects. *The pattern matching algorithm is restricted version of **depth first search***.

When user input is given, the longest pattern is matched and respective template is generated as the bot response. In the following example, for input "DO YOU KNOW BASEBALL", the response will only be 'Yes, but I cannot play'.

```
 <category>
            <pattern> DO YOU KNOW</pattern>
            <template> I am not sure.</template>
</category

<category>
            <pattern> DO YOU KNOW BASEBALL</pattern>
            <template> Yes, but I cannot play</template>
</category
```

Since it would be difficult to have patterns for each and every expected question, wildcard character '*' is used. In this example, bot response will be the same from all of these questions:

- DO YOU KNOW WHERE IS POLAND
- DO YOU KNOW WHAT IS POLAND
- DO YOU KNOW POLAND

```
<category>
            <pattern> DO YOU KNOW * POLAND</pattern>
            <template> Poland is a large country in Central Europe. Poland has a long and interesting history. The
country has been divided and its borders shifted many times.</template>
</category>
```

Another interesting situation is when the bot has different responses to same query or has the same response to different queries. The respective AIML constructed for these two cases are listed below.

Different responses to same query:

```
<category>
            <pattern>THREE HUNDRED *</pattern>
            <template> <random> <li>That is quite a lot.</li> <li>That much.</li> <li>That is a
            lot.</li></random></template>
</category>
```

Same response to different queries:

```
<category>
            <pattern>WHO IS XYZ</pattern>
            <template> XYZ is a nice person. He is my master</template>
</category>

<category>
            <pattern>WHAT ABOUT XYZ</pattern>
            <template> <srai>WHO IS XYZ</srai></template>
</category>
```

The ‘star’ element imparts an intelligent touch to the responses. The following example shows that the chatbot picks whatever is said between FOR and YEARS and uses it in its response.

```
<category>
            <pattern>FOR * YEARS</pattern>
            <template><li>A lot can happen in <star/> years.</li></template>
</category>
```

The ‘think’ element does not produce a response but it might do the processing. If any response is generated, it is stored in user predicate map (a log file locally stored on user machine containing details about conversation) and can be used later, if required.

ChatBot learning

Making the chatbot learn new stuff is a step towards accomplishing Turing Test. Learning feature is available only in the paid ALICE version Pandorabot (written in LISP). To implement learning in PeopleBot running on Program D, a new AIML object ‘learn’ is created and a processor is developed for the same. Following example demonstrates how learning is implemented:

```
<category>
  <pattern>LEARN * IS *</pattern>
  <template>
        Ok I will learn that<star index="1"/>is<star index="2"/>.
   <srai>XEDUCATE WHAT IS XSPLIT <star index="1"/> XSPLIT <star index="2"/></srai>
   <srai>XEDUCATE WHERE IS XSPLIT <star index="1"/> XSPLIT <star index="2"/></srai>
   <srai>XEDUCATE WHO IS XSPLIT <star index="1"/> XSPLIT <star index="2"/></srai>
  </template>
</category>

<category>
  <pattern>XEDUCATE * XSPLIT * XSPLIT *</pattern>
   <template>
      <learn>
       <category>
          <pattern>
             <eval><uppercase><star index="1"/> <star index="3"/></uppercase></eval>
          </pattern>
          <template>
             <eval><star index="2"/></eval>
          </template>
       </category>
      </learn>
      <learn>
       <category>
          <pattern>
```

```
            <eval><uppercase><star index="1"/> <star index="2"/></uppercase></eval>
          </pattern>
          <template>
            <eval><star index="3"/></eval>
          </template>
        </category>
      </learn>
    </template>
</category>
```

Hence when the chatbot is given input : 'LEARN APPLE IS A FRUIT', it will give user the response: Ok, I will learn that apple is a fruit. Internally it will create the following pattern-template pair and will load it as it would normally do for any aiml file.

```
<category>
            <pattern>WHAT IS A FRUIT</pattern>
            <template>apple</template>
</category>
<category>
            <pattern>WHAT IS APPLE</pattern>
            <template>a fruit</template>
</category>
<category>
            <pattern>WHERE IS A FRUIT</pattern>
            <template>apple</template>
</category>
```

This learning can be short term (for the current conversation);
it can be saved for a user profile; it can be saved for the chatbot profile; it can be saved as a universal truth and all chatbots will have this knowledge.

Apart from adding learning, there were some minor issues in Program D online version which were fixed, such as unloading of an aiml file is supported but it was not implemented in the code.

Future Improvements

Learning is in rudimentary stage currently. It works only when user input starts with 'learn' word. Next quarter's goal is to improve and expand learning.

Speech Recognizer: SPHINX 4

What is a Speech Recognizer?

Speech Recognizers aid in conversion of spoken words to text. It is another application of Natural Language Processing. Speech recognition engine requires two types of files to recognize speech. The first one is acoustic model, which is created by taking audio recordings of speech and compiling them into statistical representations of the sounds that make up each word (through a process called 'training'). The second one is a language model or a grammar file. A language model is a file containing the probabilities of sequences of words. A grammar is a much smaller file containing sets of predefined combinations of words. The modern speech recognizers are based on Hidden Markov Model. Speech recognizers are used in applications such as Health Care, Telephony, Military, assisting People with disabilities, etc.

SPHINX-4 – Speech Recognizer used in PeopleBot

The speech recognition software used in this project is open source SPHINX-4. It was created via a joint collaboration between the Sphinx group at Carnegie Mellon University, Sun Microsystems Laboratories, Mitsubishi Electric Research Labs (MERL), and Hewlett Packard (HP), with contributions from the University of California at Santa Cruz (UCSC) and the Massachusetts Institute of Technology (MIT). Sphinx-4 is in the Java^{TM} programming language, making it available to a large variety of development platforms.

Grammars in Recognizer

Speech recognition systems provide computers with the ability to listen to user speech and determine what is said. In order to achieve reasonable recognition accuracy and response time, most speech recognizers constrain what they listen for by using *grammars*. Sphinx-4 supports Word List, JSGF and N-Gram grammars. In this project N-Gram and JSGF grammars are used.

JSGF Grammar: JAVA Speech Grammar Format is a platform-independent, vendor-independent textual representation of grammars for use in speech recognition. It is developed by SUN and hence adopts the style and conventions of the Java programming language. This type of grammar file has '.gram' extension. JSGF gives 80% accuracy for smaller grammar. PeopleBot uses JSGF grammar for all transitions states, including AI Controller.

N-Gram Language Model: It is a subsequence of *n* items (phonemes, syllables, letters, words or base pairs) from a given sequence. An ***n*-gram model** is a type of probabilistic model for predicting the next item in such a sequence. N-grams used in sphinx-4 are files with '.lm' extension. The language model file is generated by a tool (web based) provided by sphinx-4 [http://www.speech.cs.cmu.edu/tools/lmtool-new.html]. N-gram is good for large grammars as compared to JSGF, but it performs poorly for small grammars. In this project N-Gram is used for Interaction mode.

Challenges Faced

A recognizer, once created, is attached to a specific grammar and cannot be changed in the lifetime of the program. Such a recognizer will not recognize sentences from other type of grammars. Also, it is not possible to have two recognizers running on same JVM. Except Interaction Mode, PeopleBot uses very small grammar, hence only JSGF would suit other modes. In Interaction mode, if only JSGF is used, speech recognition is practically restricted. To overcome this, both N-gram and JSGF grammars are used. N-gram grammar is activated only in Interaction mode and runs in a separate JVM, but both grammars run on same machine. In interaction mode, two recognizers are running and will report the recognized text; to check duplicate processing, PeopleBot is designed to respond to only one type of grammar, either JSGF or n-gram, for each recognized text.

Future Improvements

Speech Recognition accuracy is approximately 60% in n-gram. In next quarter, it is planned to improve n-gram as well as JSGF grammar's recognition accuracy. Machine learning will be incorporated to train the system for an individual's voice and accent.

Speech Synthesizer: FreeTTS

What is Speech Synthesizer?

Speech synthesis is an application of Natural Language Processing. It converts text or symbolic linguistic representation (phonetic transcription) into speech. Speech synthesis systems maintain a database of recorded speech either in phonetic form or in the form of entire words or sentences. These recorded pieces are concatenated to produce synthesized speech. Speech Synthesizers are employed to aid people with visual or reading disabilities, telephony, etc. The famous scientist Stephen Hawking uses speech synthesizer to communicate.

What is a TTS engine?

A text-to-speech engine is composed of front-end and back-end. Front-end converts raw text of numbers and abbreviations into written words. This process is known as text normalization or tokenization. It then assigns each token a phonetic transcription. The back end, known as synthesizer, converts the symbolic linguistic representation into sound waves. Incorporating vocal tract model and other human voice characteristics are done in this stage.

The current text normalization challenge is to design heuristic techniques to guess disambiguate homographs. For example, in sentences ‘Have you read the book’ and ‘read the book’, the word ‘read’ is pronounced differently. Thus synthesizer technologies are emphasizing their efforts on developing naturalness and intelligibility in synthesis engines.

FreeTTS-Software used in PeopleBot

PeopleBot uses text-to-speech synthesizer; hence FreeTTS (an open source text-to-speech synthesizer), written in JAVA^{TM} programming language, was an apt choice. It is based upon Flite, a small run-time speech synthesis engine developed at Carnegie Mellon University and is built by the Speech Integration Group of Sun Microsystems Laboratories. It uses Hidden Markov Model (HMM) for synthesis. It provides partial support for the Java Speech API (JSAPI). PeopleBot uses male US English voice.

Future Improvements

FreeTTS is used as is in PeopleBot. The goal of the project in next quarter is to employ machine learning to train a voice to produce sound with emotions.

AI Implementation: Fuzzy Logic

What is Fuzzy Logic?

Fuzzy Logic is a multi-valued logic derived from fuzzy set theory. It deals with approximate values (e.g. a proposition can be almost false, may be true or very little true) rather than accurate ones (a proposition is either true or false). Fuzzy Logic uses complex algorithms to perform data evaluation, derive conclusion and make decisions. It is easy to understand, tolerant of imprecise data, flexible and is based on natural language. Hence its decision making process is similar to human thought process.

Implementation in PeopleBot

The main processing part of PeopleBot uses Fuzzy Logic for making complex decisions. The fuzzy logic programming thread starts along with the main Artificial Intelligence program and runs till end. It requests various types of data from other threads at regular intervals; For example, ARIA Thread is queried for battery voltage, current speed, current angle, etc. Vision Thread is queried if any face or gesture is seen; Listening (Speech Recognition) Thread is asked if it heard anything. After fetching data from all threads, it makes new decisions every minute.

The heuristics used to construct fuzzy logic are developed by processing the combined knowledge of acquired and current information. The following examples give brief description of few of the heuristics used:

- ➢ Making decisions for State Transition :
  - IF battery is full (>80%) AND detects a face looking at it, it will move towards the face at a certain speed and wait to switch mode till it hears a greeting message (thus transitioning from Unsupervised Mode to Interaction Mode).
  - IF battery is 80%-60% AND detects a face, then PeopleBot will move only if it hears a greeting message (this behavior preserves energy).
  - IF battery is below 30%, PeopleBot will not move at all and will start flashing alert message on UI and will also start beeping.

- ➢ Making decisions for speed of locomotion :
  - Speed is directly proportional to battery. Hence when PeopleBot has to decide on a speed, it will use current battery voltage to determine its speed.
  - If user explicitly sets speed even when battery is low, PeopleBot will check if the requested speed value suits its battery status. If not, it will re-calculate a speed value and will move with the new speed.

- Making decisions for changing direction (angle) :
  - Changing direction consumes great amount of energy. Hence when battery is running low, changing direction is avoided.
  - If user requests PeopleBot to change direction while battery is low, it will do so, but to preserve energy will reduce speed and take some other measures.

- Making decisions for preserving energy :
  - Apart from above listed energy saving measures, PeopleBot can take decision to stop listening, speaking or going into Interaction Mode at all to save energy.

References

PeopleBot:
http://www.mobilerobots.com/ResearchRobots/ResearchRobots/PeopleBot.aspx

Natural Language Processing:
http://en.wikipedia.org/wiki/Natural_language_processing

ChatBot:
http://en.wikipedia.org/wiki/Artificial_Linguistic_Internet_Computer_Entity
http://www.alicebot.org/
http://www.aitools.org/Main_Page

Speech Recognizer:
http://en.wikipedia.org/wiki/Speech_recognition
http://cmusphinx.sourceforge.net/sphinx4/doc/Sphinx4Whitepaper.pdf
http://www.speech.cs.cmu.edu/
http://java.sun.com/products/java-media/speech/forDevelopers/JSGF/

Speech Synthesizer:
http://en.wikipedia.org/wiki/Speech_synthesis
http://freetts.sourceforge.net/docs/index.php
http://labs.oracle.com/speech/index.html

Fuzzy Logic:
http://www.xpertrule.com/pages/fuzzy.htm
http://en.wikipedia.org/wiki/Fuzzy_logic