Project

Prepared for: Alexander Stoychev

HCI 585x Developmental Robotics

Iowa State University

Prepared by: Julie Welsh

March 6, 2011

Grounding via Embedded Cues and Affordances Utilizing an Android Phone as a Low-Cost Robotics Platform

Introduction

Housework who needs it?! For legions of housework haters, a personal domestic robot, like those portrayed in science fiction, movies and cartoons would be a dream come true. However, before this dream can become a reality, there are a number of problems to solve, not the least of which is the Symbol Grounding Problem (Harnad 1999). The Symbol Grounding Problem is concerned with understanding the process of how words are associated with their meanings. In other words how are the symbols in our heads (words) connected to tangible objects in the physical world?

In order to function flexibly and adaptively within a natural environment, an embodied agent or robot must be grounded, able to make this connection between concrete instances of objects 'in the world' and its internal data structures (knowledge 'in the head'). Grounding is a prerequisite for communicating with humans. Household service robots must be able to respond to words and must associate those words with the objects in its operating environment.

Affordances, coined by Gibson (1977) and expanded upon by Norman (1988), are those properties of an object that suggest potential for action. Affordances are dependent upon the specific capabilities of the perceiver. Gibson's definition of an affordance was purely rooted in the environment and involved direct perception by the actor. Gibson defined affordances as action possibilities inherent in the environment in relation to the actor, but not dependent upon the actor's ability to discern of these possibilities (or form a mental model). Donald Norman, wrote about perceived affordances, which took the viewpoint of the individual and their understanding of how a particular object might be used. Normans perspective requires a mental representation based on experience, that is referenced with respect to how an entity might be utilized (in that regard affordances are learned).

Affordances are important in robotics because they suggest how an object might be used. If robots are able to correctly interpret the affordances of an object, they are more likely to be able to select appropriate behaviors and to interact with it successfully. (Of course this assumes that the objects are designed with 'good' affordances: with visual cues or other information that suggests how the object is to be used. There sufficient variation in the details of how everyday objects are used to cause difficulties for robots (and even some people-at times the fault of the designer, not the users). Simple objects such as doors, light switches, or faucets cause problems for humans on a regular basis.

As a usability person, it occurs to me that we are asking a robot to function in an environment designed for humans, not robots. Robots have very different sensing modalities than we do. If we expect them to function in our households, it is reasonable to modify the environment to support

and take best advantage of their unique capabilities. Most homes are designed for the average human and cause difficulty for humans who are not within the norm. We make accommodations for the elderly, children, and individuals with disabilities. Why not do the same for robots? Affordances (or perceived affordances per Norman) pertain to a perceived relationship between our bodies and the environment. For a robot affordances that we perceive don't always map to the capabilities inherent to their makeup. Because robots don't have the same sensing equipment that we humans do, they are not receiving the same perceptual information that we are.

This project will explore the problem of grounding and affordances for a hypothetical robot operating in the home. It will describe a system for grounding objects in the physical world by encoding data about objects in the environment that the robot will be able to detect. The "robot" will store sensory information in conjunction with object data as it is encountered in the environment. The perceptual memory store is intended to be accumulated for use with learning algorithms. The robot will have ready access to a persistent common knowledge base, which can be used for to provide additional context.

The 'robot' platform used for this project will be an Android phone. The Android has many attributes that make it attractive for use as a robotics platform, as will be described below.

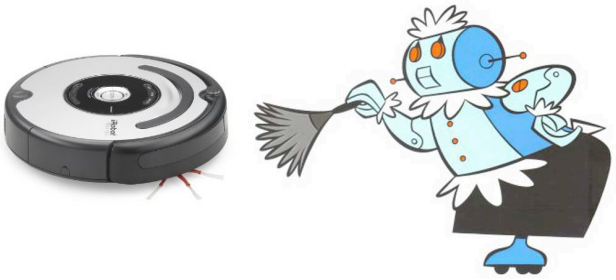
It should be emphasized that the system described herein is intended to serve as a complement to, not a substitute for developmental learning algorithms. The embedded data will serve as 'training wheels' to bootstrap the system. The stored perceptual, bottom-up, data linked to top-down semantic data provides a foundation that can be used to train the robot to recognize new objects based on prior stored perceptual data and the associated semantic data. In a home, it is not realistic to assume that experimenting with the good china is fair game for a robot. Techniques such as behavioral babbling may be appropriate in the lab, but are not necessarily desirable in the home. By encoding information about object properties, the robot will know to 'handle with care'.

Target Audience

There are three main target segments for this technology. This system is intended for people who don't have the time, ability, or desire to do housework. This technology should be attractive to anyone who would like assistance with household chores. This could range from a stay at home parent to a busy executive or elderly couple. Basically anyone who doesn't care to do housework or who needs extra support in the home.

Motivation and Need

For individuals employed outside the home and working long hours, household chores are a burden, stealing precious leisure time that could be spent on more meaningful pursuits. Many people simply detest housework. A third group are not able to be independent due to disability and need support to stay in their homes. Any advancements in household automation would be a huge boon to all three of these audiences. In order to evolve from Roomba to Rosie, it will be necessary to develop systems that are capable of associating meaningful information about their physical environment with an internal system representation.

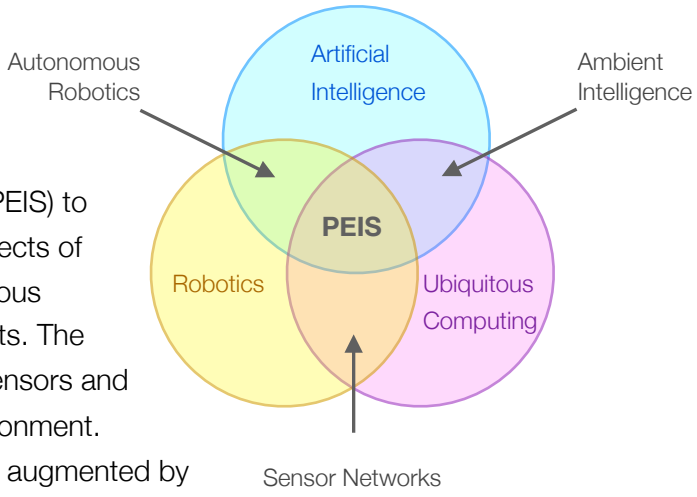


Getting from Roomba to Rosie will require robots that can make sense of their environment

In the not so distant future, many homes will be equipped with smart systems that could include a multitude of sensors. Today most households are equipped with wireless networks, and ubiquitous computing is becoming a reality through the pervasive use of location aware mobile devices.

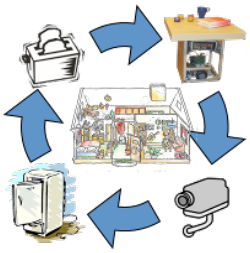
Related Work

Researchers from the AASS Mobile Robotics Lab at ¨Orebro University in Sweden coined the term Physically Embedded Intelligent Systems-ecology (PEIS) to describe an approach that combines aspects of ambient intelligent systems and autonomous robotics to develop assistive service robots. The PEIS-Ecology approach distributes the sensors and system components throughout the environment. Perception and manipulation of objects is augmented by information stored in IC tags and the robots' positioning within the environment is supported by external cameras. A variety of networked subsystems and devices communicate to cooperatively to perform common household tasks.

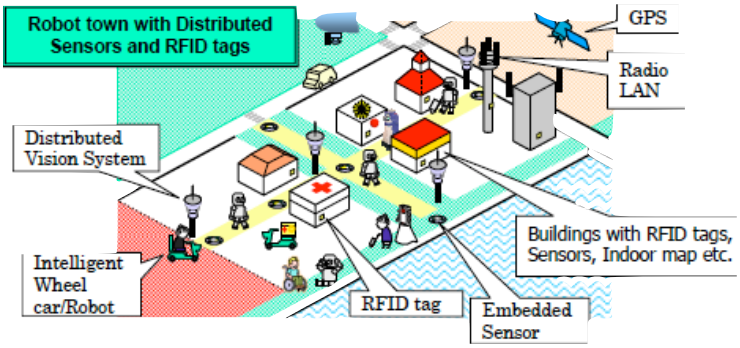


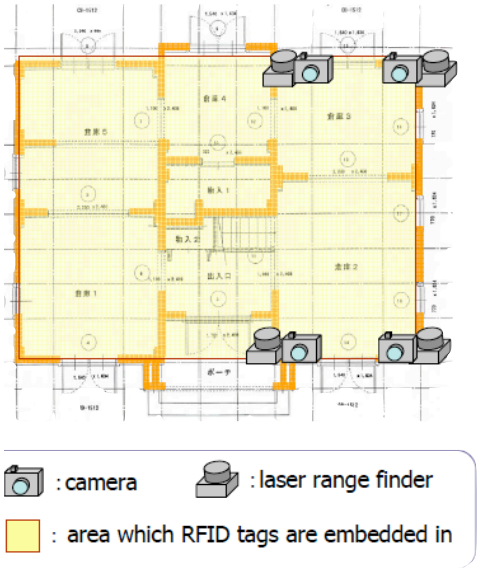
PEIS differs from the traditional view of the robot and environment. In the traditional view the robot and its environment are seen as separate entities. The robot is left to observe a non-deterministic

environment and must operate on the basis of its perception alone. PEIS unifies the robot and the environment through a system of sensors, smart appliances, RFID tagged objects, and other robots. In PEIS the term robot is used to refer to any device with computational and communication resources that is able to interact with the environment through sensors and/or actuators. Robots in the system can range from humanoid robots to a smart fridge or microwave. Using this symbiotic approach the individual robots are relieved of some task burden and are able to accomplish tasks in a coordinated and cooperative fashion with other entities in the environment. Non-robots in the environment are tagged with information via RFID tags so the robot does need to guess about what an object is, what its properties are or how it is used.



Researchers at Kyushu University in in Fukuoka City, Japan have developed the Robot Town Platform, an information-based structured environment designed to support a robot by providing it will real-time information about its location and the environment. The platform is implemented in a city block outfitted with cameras, laser range finders, and RFID tags that are distributed throughout the environment.





Robot Town also includes a 5 room home that utilizes ceiling mounted cameras and laser range finders to supply location information. GPS requires a clear line of sight so will not function properly indoors.

KnowRob, is a knowledge processing system for autonomous personal robots. The system combines encyclopedic knowledge, models of the environment, and action-based reasoning models, which are used to learn behaviors. Access to the information is via a uniform query language. The system is designed to ground the robots, by linking their perception and actuation systems with with abstract knowledge representations.

A household robot requires both knowledge and perception to perform tasks. The K-CoPMAN (Knowledge enabled cognitive Perception for Manipulation) system extends the KnowRob knowledge system by abstracting perception into a logical representation. The K-CoPMAN system acquires and stores perceptual data during a robot's operation and associates it with symbolic names that can be used for perceptually grounded processing. Raw sub-symbolic perceptual data is used to extend static symbolic data maps that are stored as data structures in the system. The system is able to use abstract (symbolic) knowledge about scenes to simplify perceptive tasks. K-CoPMAN updates world state information, such as the presence and locations of objects, so the robot is environment-aware. For example the robot will know to "look" to find a particular object in the place it belongs using a combination of static knowledge, state, and perceptual information.

Germany's answer to the Roomba was developed through a partnership of Infineon, a subsidiary of Siemens with a large R&D arm, and textile manufacturer Vorwerk. The robot navigates a "smart carpet" that contains a sublayer consisting of a grid of passive RFID tags. The robot is equipped with an RFID reader and supplies all power necessary to read the tags, and is also able to write to the tags if desired. Equipped with an internal map of the area that includes obstructions, the robot tracks its location using the tags.

Idea Filter

If a 2-year-old child cannot solve your task then your project idea is probably too complicated.

For this project scope is limited to a robot with the operating domain of the first floor of a single family home, primarily in the kitchen and dining room areas. The robot should be location aware, able to follow simple commands, and answer queries.

Approach

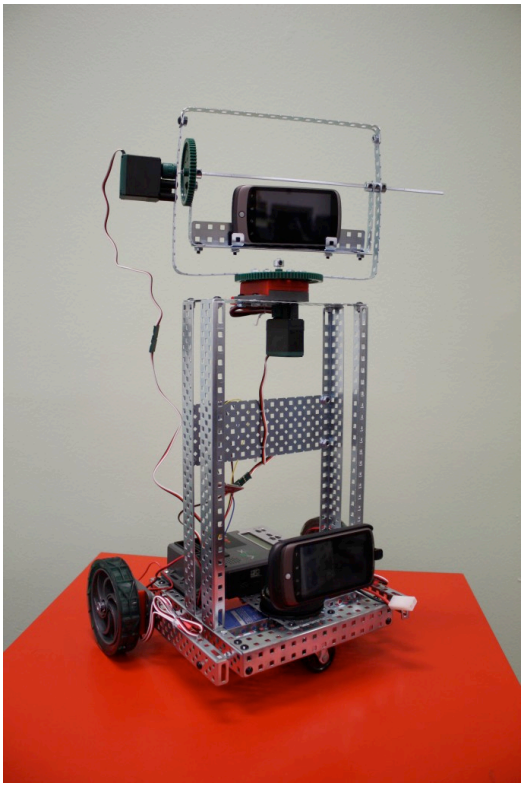
This project explored embedding location and affordance information throughout the environment to enable a robot to find and interact appropriately with objects. Shared system data and a

domain specific knowledge store is continually available through a shared repository housed via cloud storage.

Data tags were embedded throughout the environment to encode information about the physical layout and properties of static elements in the environment, such as rooms, furniture, cabinets and counters. Objects of interest (defined in this project as objects that we would like the robot to interact with) were also be tagged barcodes. The application for the system proposed here is an in-home service robot. Cues in the operating environment will supply information about the physical location, objects and available object interactions. The project will include a framework on which the robot can build a subjective representation based on experience. The perceptual repository is intended to be used as part of a training regimen where the tagged items in the environment serve to ground the physical properties of objects with commonsense and semantic knowledge about the world. Additional learning algorithms could be used to generalize the data to similar contexts. (The learning algorithms for generalization are be outside the scope of this project.)

Why Android?

The Android platform has many features out of the box that make it attractive for use as a robotics platform. As a distance education student without access to a lab, a robot, or big money, these factors played into my selection of Android for this project.



The Android is widely available, and I happen to own one. It is relatively low cost for the amount of computing power and functionality that is packaged with the device. It is small, lightweight and by definition extremely portable, so could very easily be used for a mobile robot system.

Cellbots are cell phones used as robotic control platforms. There are several robot hardware platforms that Android can interface to including: Lego Mindstorms, iRobot Create and Roomba, Truckbot, TRRSTAN, Audruino, and Vex Pro. Software libraries in Python, App Inventor blocks and a Java application in Android market are available to control Android Cellbots. App Inventor can be used to program Lego Mindstroms. The Python libraries support

communication with Arduino robots, VEX Pro, and the iRobot Create & Roomba. While today most Cellbots are being built by hobbyists, these frameworks could be used to develop a larger scale embodied robot. There are currently a number of interfaces available to enable the Android platform to control servos and motors and to interface to external sensors such as ultrasonic sensors for object avoidance, and light & color sensors for enhanced vision.

With an open architecture and many open source resources, the Android is extensible so can be used to develop custom applications or link existing ones. This capability will surely continue to evolve given that Android's market share is exploding and, since it is developed by Google, there are built-in synergies with many online services and applications.

Android phones come equipped with an impressive variety of sensors. The sensors available on most Android phones includes: a camera, microphone, accelerometer, GPS, and compass. With built-in support for multiple communication channels, the Android can communicate through wi-fi, phone networks, SMS, Bluetooth, and usb.

In addition to its hardware capability, the Android OS includes software support for a number very useful functions. Of interest to this application are: text-to-speech, speech recognition, barcode scanning, and persistent cloud-based storage of data via the TinyWebDB component.

RFID readers for Android are just around the corner. The latest release of the Android OS, Gingerbread, has built-in support for Near Field Communications (NFC). In systems composed of hardware and software using the Android NFC API, apps will be able to respond to NFC tags embedded in stickers, posters, and other devices.

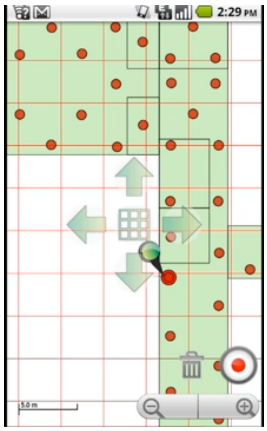
Navigation

A current challenge for many systems is indoor navigation. GPS does not function without a clear line of sight to the satellites used for positioning. Furthermore it has limited accuracy (3 meters). The laser scanning technology that has been used in other robotic systems is prohibitively expensive.

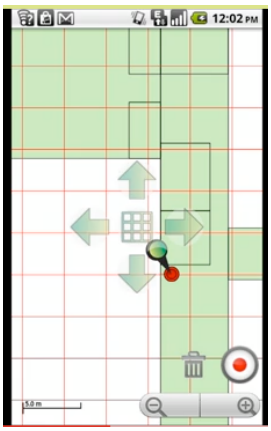
Indoor positioning technology that utilizes environmental RF (radio frequency) information including WLAN, GSM, electric compass orientation, and Bluetooth, was a seemingly attractive option for indoor positioning using using an Android phone. Using this technique a radio map is developed for a set of marker points (a training phase). During operation the position of the device (in our case the robot) is determined by matching the current RF sample with the stored map. Several indoor navigation platforms were investigated for this project. Platforms considered included RedPin, SkyHook, and Gecko. RedPin and SkyHook were eliminated due to insufficient accuracy.

Gecko, a system being developed as part of the LocLizard platform (http://loclizard.com/), by Qubulus of Sweden, was selected because it promised 1 meter accuracy and was available to developers free of charge. Unfortunately, after signing up to participate, I discovered that the API used to incorporate the fingerprint information into custom applications had not been released.

Gecko uses Google Earth to



Gecko map during training phase showing location fingerprints



Gecko map showing

Upon discovering that LocLizard would not be a viable option for indoor navigation, alternative options were investigated. Augmented reality (AR) platforms are location and context aware. Both features totally on point with the goals and objectives of this project.

AR platforms synchronize camera data, and smartphone sensor data with online content utilizing real-time device state data. Currently there are 3 major players in this domain, Layar, Juniao, and Wikitude. All of them are location-aware and designed to provide information to the phone according to the device's specific orientation and movement through the environment. They all utilize GPS for outdoor navigation, but use different techniques indoors. Layar and Juniao both utilize marker based systems where a code is read by the device to support navigation. Layar utilizes QR Code barcodes and Juniao uses a proprietary barcode format. Layar uses Skyhook under the hood, so did not meet the accuracy requirements for the project. Juniao's system appeared to be quite robust, however the development environments supported were php, and C# (.net). I have no experience in php so did not want to attempt it. Because I use a Macintosh, C# was not a viable option.

Proof of Concept

This is a proof of concept exercise because, as noted above, the technology I had hoped to use for this project, Gecko and RFID for Android, are not available yet. All proposed aspects of this system are technically feasible and should be available within 6 months to a year. The exercise of executing this project will provide a foundation for a full implementation at a later date.

QR Code Barcodes encode location information and object properties

I opted to use QR Code barcodes to track and store location, navigation data, and object property data. The barcodes are used to simulate location fingerprint data in the test system by storing coordinates that correspond to locations in the home. QR Code barcode tags simulate RFID tags and also store location data. The data contained in these markers is used to build out the location and object property database needed to construct the model of the environment during a training phase and will provide data required during the operation phase. An Android phone will be used to simulate a robot for the reasons noted above.

A robot should be able to connect the behavioral repertoire linked to an object through the affordance and property data associated with it and by using the practical knowledge provided from the knowledge store. A tagged object will be looked up in the knowledge store. If the object instance does not exist, it will be added to the knowledge store. Perceptual data will be captured and associated with object instances as they are encountered in the environment. The perceptual information is intended to be utilized by learning algorithms that could ultimately allow the robot to recognize objects based on their physical attributes alone, using it's acquired perceptual and practical knowledge. An example of a command sequence might be: Where are you? Go to kitchen. Where is blue coffee cup?

Given the project timeframe, it was determined that it was not feasible to attempt native Android OS development in Java using the Android SDK. Google App Inventor was used to program all data collection and command software on the phone. App Inventor is a drag and drop web-based interface that can be used to create applications for Android. It is quite flexible, but does not support the entire Android API. For example, it is not possible to program the phone to automatically take a photo without user intervention.

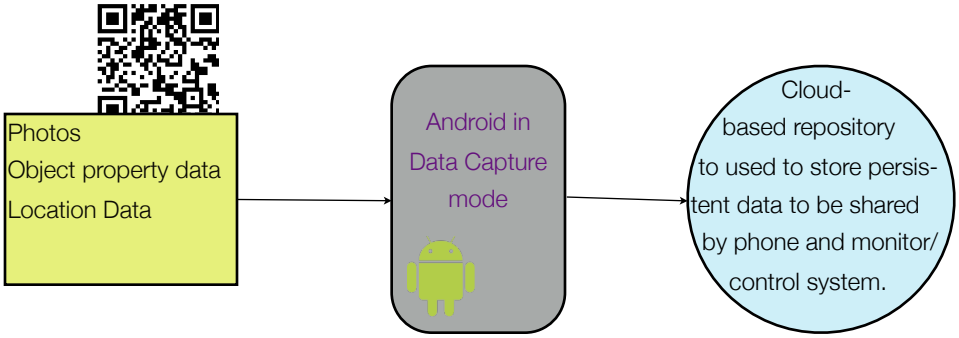
The data collected is saved to a cloud-based data repository using the App Inventor TinyWebDB component. To train the robot, information embedded in the environment via simulated RFID tags (barcodes) was associated with data acquired through the other senses (location, orientation, and images). In a 'real life' system, it is expected that RFID tags would be used. Perceptual data is stored on the phone in a perceptual store TinyDB (the App Inventor component used for local persistent storage).

During Command mode the Android responds to voice commands. In response to spoken commands it indicates whether the command was understood and then states the success or failure of the command.

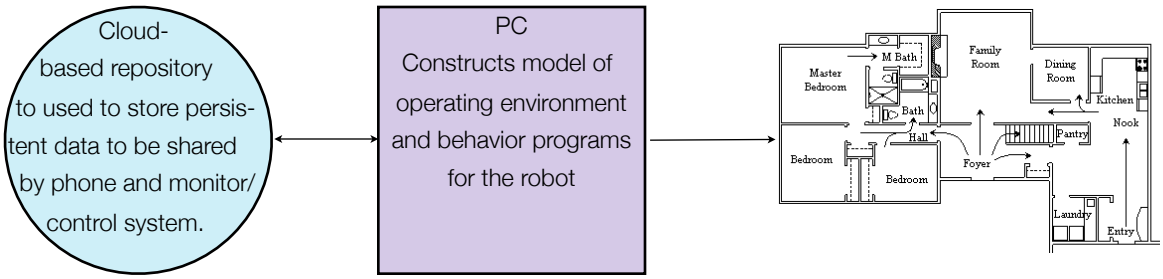
Whenever the Android changes location, it updates its current location state locally and also updates the shared data repository with the new location. A monitoring/control application tracks the Android location and displays it in a browser user interface.

Modes of Operation

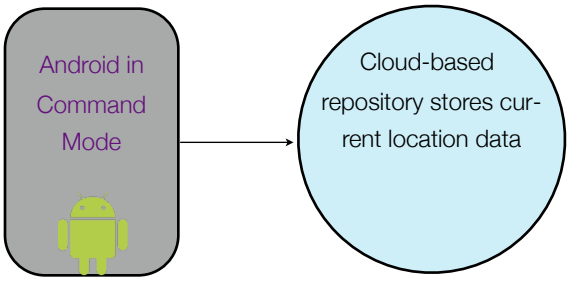
Learn Mode - During the training mode the 'robot' (Android phone) takes readings of the tags containing location, object property, and context data. The tagged data will be sent to a cloud-based data repository using the TinyWebDB component. During the training period, the Android also collects sensory and context data, including image capture, location, and orientation data.



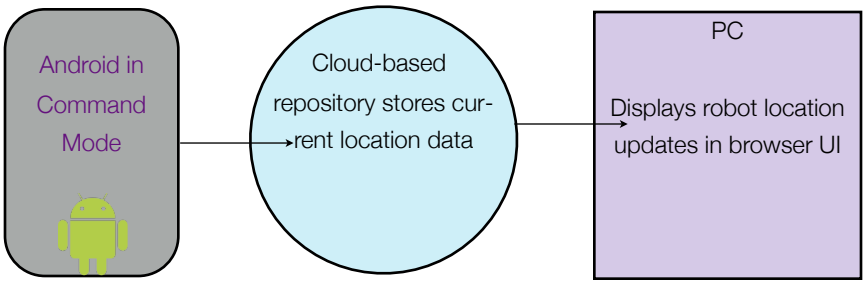
Construction Mode - The construction mode will be used to build out a model of the robot's environment using data collected during training mode. Data points collected will be stored with a map of the robot's operating environment.



Command Mode - The system responds to voice commands and utilizes information encoded in the environment through the data tags and also data stored in the shared repository. Because this is a data-driven system, with a persistent data store, we are able to simulate RFID tags in the sense that the Android can query for an object's location . Communication between any control systems and the robot will be through the common data repository.



Monitor Mode - A monitor and control program continually checks the current location of the Android and displays it's location on a map of the floor plan.



Software libraries

This project utilized a combination of Google App Inventor blocks and Google App Engine libraries. A TinyWebDB component was deployed to Google App Engine to serve as a shared persistent datastore between the Android and the rest of the system. To limit the number of development

environments and libraries, the monitoring user interface was developed using Google App Engine instead of Web2Py as had been originally planned.

Components used: Barcode scanner, text-to-speech, speech recognition, camera, orientation sensors, TinyDB (for local storage), TinyWebDB for shared storage.

To utilize text-to-speech a speech synthesis application, SpeechSynthesisEN was installed. For barcode support, ZXing Team Barcode scanner was installed.

Equipment

Equipment the equipment used on this project will include an Android HTC Desire phone with version 2.2 of the Android OS (necessary to enable program storage on the SD card), a networked Macbook Pro computer, and printer to produce barcodes.

Evaluation Methodology

Test Plan

The test protocol will consist of a training mode, construction mode, a command mode, and a monitor mode as described above.

The the training phase will encode location data for a residential home using location data based on the measured floor-plan of the house. Static features such as cabinets, furniture, counters, drawers, and appliances will tagged for reference as well. A number of common household objects will be tagged with properties relevant to the robot's capabilities. The data capture application will acquire barcode data and also store sensory data that is associated with the barcoded objects. The sensory data will be stored to a local database on the Android and tagged with the object ID. The text data captured though the barcode will be stored in the common repository.

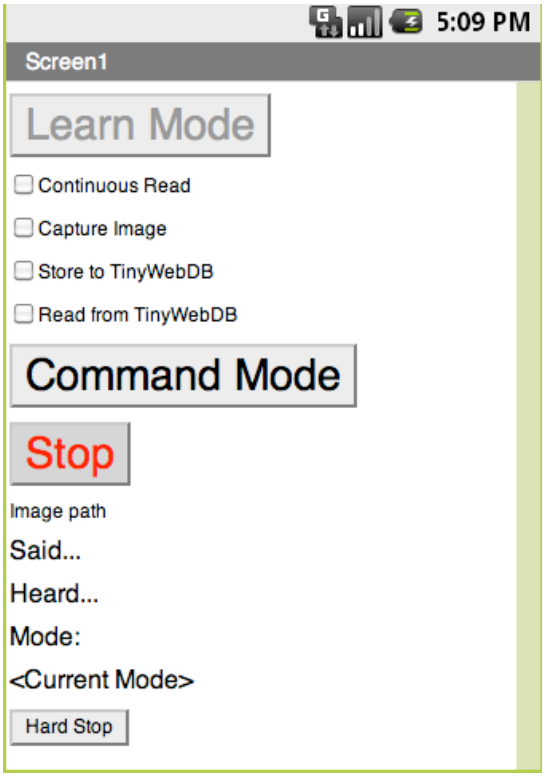
The construction mode will be used to build the model of the robot's operating environment by associating the data collected through data tags with a known floor-plan and knowledge base containing robot behaviors. It will also create on inventory of all of the objects that the robot is expected to interact with.

The operating mode will be used to demonstrate the system capabilities through a command/ response protocol with between the control system and the 'robot' (android phone). The robot will be expected to respond to a series of commands by sending back appropriate responses to the control system via the common repository. This project will be considered a success if the robot is able to successfully navigate to the commanded location, find the object commanded (based on

proximity), send back a picture of the object and describe correct behaviors based on object properties.

Design

User Interface



A single Android application was created to be used for Learn Mode and Command mode. Options were added so it could be used for storing information to the web database, scan continuously, and capture image data. The Android user interface is shown below.

•When Learn Mode is selected the Android is placed in the data collection mode.

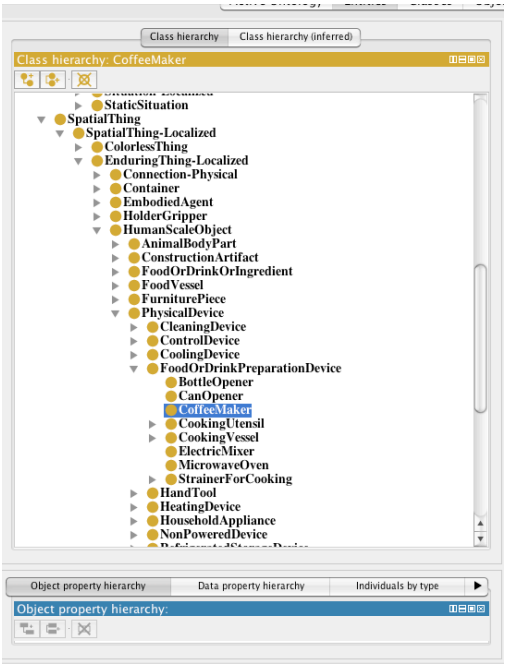
•When Command Mode is selected the Android accepts voice commands.

•Continuous read will cause the barcode reader to read without a separate button press in Learn Mode.

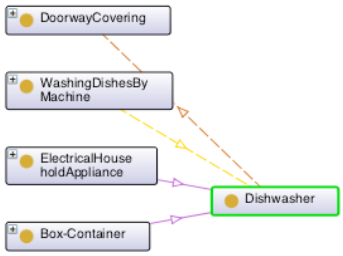
•Capture image will take a photo after the barcode is scanned. The image path, orientation data, and object ID are stored in a separate database on the phone. in Learn Mode.

- When *Store to TinyWebDb* is selected, the Android will store all scanned data and updated location data to the cloud storage.
- When *Read from TinyWebDB* is selected, the Android will initiate the read commands to fetch all data from the cloud storage.

KnowRob Ontology



A local copy of the KnowRob ontology was downloaded and installed along with, Protégé, a free open source ontology editor and knowledge acquisition system developed by Stanford. The KnowRob ontology is available at http://ias.cs.tum.edu/kb/knowrob.owl. I had hoped to interface to KnowRob for this project and to create and link instances of the tagged objects in the KnowRob ontology to objects in my datastore. I found this was a much more involved process than had been anticipated. Nonetheless, it was very instructive to explore the classes to see how the ontology was defined. Given sufficient time (and using Java) this would be very possible. Even though WebProtégé was developed in App Engine, there is not an App Engine interface available, nor is one planned. As my project was already underway and running with App Engine, it was not realistic to attempt the KnowRob interface for this project.



Graph of a the Dishwasher class in KnowRob.

Data Structures

The barcodes are encoded with a tag identifier in position 1 to create a unique ID to identify the object. This was followed by a sequence of field/value pairs. Fields were designed to be separated by a pair of semicolons :: and fields and values were separated by the pipe character |. The table below shows the objects represented in the system and a representative object instance.

Object	Sample Record	Count
sofa	type|sofa::room|family room::x|312::y|43::material|fabric::color|yellow::affords|sitting:: moveable|semi	2
chair	chair3::type|chair::room::family room::x|465::y|115::material|fabric::color|multi ::affords|sitting:: moveable|semi	4

Object	Sample Record	Count
table	table1::type|coffee table::room::living room::x|362::y|457::material|oak::color|brown::affords|object support::moveable|semi	3
sink	sink1::type|sink::room::powder room::x|660::y|433::material|porcelain::color|white::affords|container::moveable|no	3
dishwasher	dishwasher1::type|dishwasher::room::kitchen::x|895::y|74::ht|36::material|metal::color|black::affords|container::moveable|no	1
counter	counter1::type|counter::room::kitchen::x|895::y|74::ht|36::material|corian::color|white::affords|object support::moveable|no	6
cupboard	cupboard1::type|cupboard::room::kitchen::x|895::y|74::ht|24::material|wood::color|red::affords|container::moveable|no::affords|knob grasp	12
drawer	drawer1::type|drawer::room::kitchen::x|895::y|74::ht|31::material|wood::color|red::affords|container::moveable|no::affords|knob grasp	6
cup	cup1::type|coffee cup::belongs in::cupboard8::room::kitchen::ht|54::material|porcelain::color|red & white::affords|container::moveable|yes::affords|handle grasp::capacity|10 oz	5
plate	plate1::type|dinner plate::belongs in::cupboard7::room::kitchen::ht|54::material|china::color|blue::affords|object support::moveable|yes::affords|object grasp::capacity|10 dia	5
navigation point	nav1::type|navpoint::room::family room::x|300::y|156	17
navigation route	route1::type|navroute::from|family room::to|kitchen::nav1|nav2|nav3|nav4|nav5|nav6|nav7|nav8|nav9|nav10|nav11|nav12|nav13|nav15|nav16	6

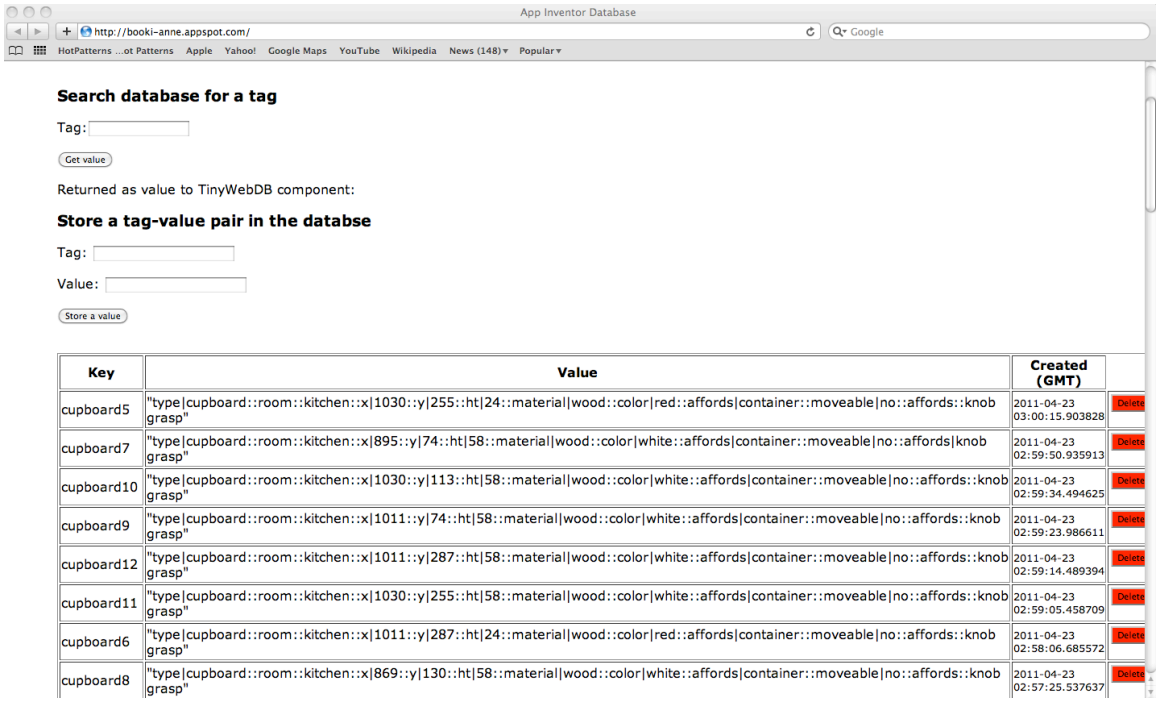
It was discovered that some of the fields were coded incorrectly. Fortunately the format was consistent so it was possible to be accommodated the deviation in the parsing routines. It would have been possible to correct the data on the web server, however that would have required that all of the object bar codes be reprogrammed, printed, and cut.

Barcodes

70 QR Code barcodes were generated using the utility at http://www.barcodesinc.com/generator/qr. A set of codes were used to 'train' the Android and also to create the permanent datastore. The codes were distributed to the items listed in the table above. While in learn mode the Android records the tag data to its local database and optionally to the web database. When the image capture option is selected. Barcodes were sized according to their placement. Navigation points were created at a larger size, 300x300 pixels because the images would be viewed from a greater distance than would be the moveable objects like plates and coffee cups. Barcodes for those items were created at 150x 150 pixels. Barcodes for furniture, cabinets, counters, etc. were created at 200x200 pixels.

Data Service

App Inventor makes use of a TinyWebDb component to share data with other applications. Google App Launcher was used to build and deploy a Google App Engine web service in Python. The service includes a user interface that can be used maintain the records. It is accessed at URL http://booki-anne.appspot.com/



Operation:

Learn Mode…

The Android stores semantic data records to the object database and sensory information to perceptual storage. Semantic data can be stored to the Android only or to both the Android and the cloud. The perceptual data only resides on the Android at this time due to limitations of App Inventor. This data could also live in the cloud if Java had be used to build the project. App Inventor and the TinyWebDB do not currently support storing or accessing image data.

Command Mode…

When the Command Mode button in pressed, the system activates the SpeechRecognition component, which waits for a spoken command. When the AfterGettingText event is triggered the command text is parsed for a supported command. If the text string does not contain a valid command, the Android uses the TextToSpeech component to say “I don’t understand.”

If a supported command is found it is parsed into command and command detail components. The Android responds using current location data and the information it has accumulated through the learning mode. If appropriate tags will be read to determine the current location, and the update is sent to the shared datastore.

Results

Data Collection

In Learning Mode, data is stored to the persistent Google data store as well as to local memory on the Android to improve performance. During this operation, the Android also stores location, context, and sensory data. This was successfully completed using the barcode scanner.

Command Mode - voice recognition and speech

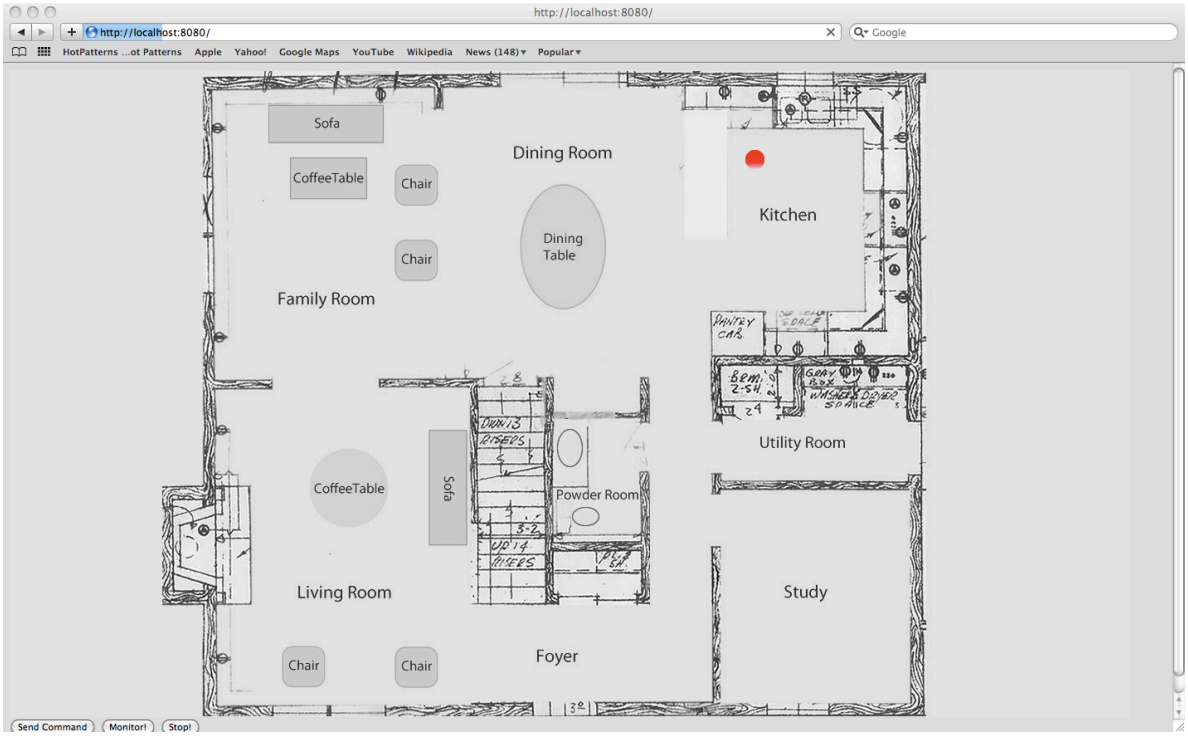
The Android was able to recognize and respond to a set of spoken commands with mixed success. The supported commands were recognized consistently by the Android, but some of the object types were problematic.

Command	Results
Where are you?	The system is able to accurately report its location based on the last navigation point scanned.
Where is [object description]?	The system is able to describe where an object is located according to the object described. It reports the room in which it is located, where it belongs (if applicable) and the stored affordances (if applicable).
Go to [room]?	The system is able to determine that you are not asking for a valid room, but does not navigate correctly.
How many [object description]?	The system is able to respond to voice command respond by matching instances according to the attributes desired. So it is able to respond correctly to “How many plates” or
Show me [object description]	The system is able to locate an image of the object your are seeking if it is found. The object is not displayed.

The Android had a difficulty with some words, but did get them eventually with one exception. Sink proved almost impossible unless it was combined with another word. ‘Kitchen Sink’ was no problem, but sink proved virtually impossible for the Android to discern. (I did not keep statistics on these frequencies as it was not the focus of this project.)

Navigation Monitoring

A second Google App Engine application was developed to ping the TinyWebDB server at 5 second intervals while the robot is in Command Mode. The user interface for the application shows the floor plan of the home with the furniture, counters, appliances, etc. When the Android scans a barcode with a navigation point, it’s location information is updated and the new coordinates are logged it to the TinyWebDB. The application view is updated to show the new robot location. Because the location information is stored to a cloud based repository, this application can be used as a remote monitoring system for the robot. Next steps would be to expand the application to accept text commands for remote control of the robot and include mapping for the active objects in the environment.



Lessons Learned: the good, the bad, and the ugly

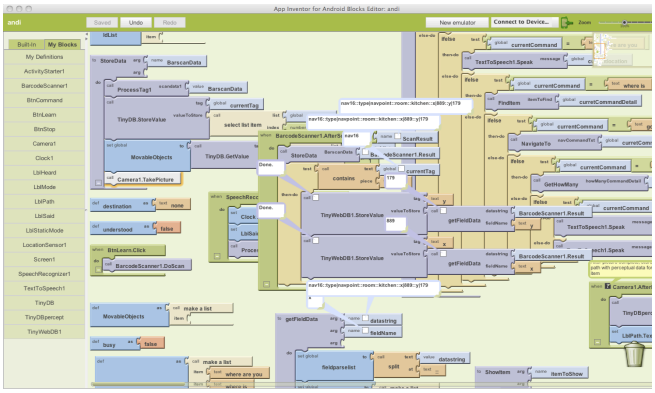
I elected to use App Inventor instead of attempting to program a native Android application in Java. I spent an inordinate amount of time wrestling with the clunkiness and limitations of App Inventor. In hindsight, I believe that the project would have been *easier* to develop in Java. While App Inventor is a very innovative idea, in that it seeks to make programming accessible for 'non-programmers', not surprisingly there are significant shortcomings in terms of the functionality provided and programing environment.

Good

On the plus side there is access to most of the Android's sensors and hardware. At first it is pretty fun to click the blocks together, but that gets old very quickly. The framework has fairly good support for text and list processing (though there I would have appreciated more control over how things were implemented, and often needed to devise work-arounds to get the desired results). The speechRecognizer and text-to-speech modules worked better than expected, though there were a few exceptions that have been noted above.

Bad

There is not fine-grained control over the programming language., which does not support good programming practices, so was very frustrating to work with. App Inventor only supports global variables, no OOP, no constants, no data structures. Another advantage to Java, may have been the ability to interface with the KnowRob OWL ontology, which has a Java-based API. The TinyDB and TinyWebDB components do not support any SQL queries, not even wildcard searches on the record ID. So all queries and searches require referencing the exact tag ID. The value returned from the query is a single text string.[1] This required a great deal of parsing and manual processing. It it not able to store binary data, only text, so it was not possible to store images remotely.

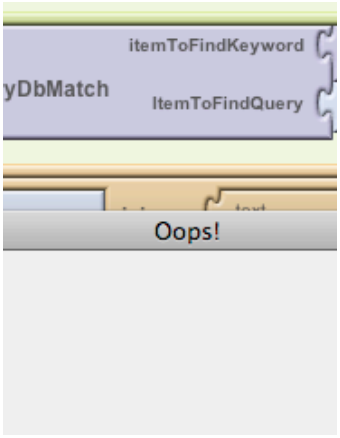


Ugly

The development environment is horrendous once the application reaches any size beyond a trivial application, and it becomes extremely time consuming to do the most basic of tasks. I think I took a few years off

[1] Near the end of the project support for CSV, comma separated values was added, but it was too late to take advantage of.

my wrists dragging and dropping the blocks. There is no way to search for *anything* in your code, you cannot copy from one program to another, so when I decided to combine my code into a single program, I basically had to start over, which was a shock. Because App Inventor is considered Beta software, there were lots of surprises, not all of them pleasant!



Limitations

As noted above there were a number of constraints imposed by the development platform and also the technology employed.

Most of people would not tolerate a home peppered with barcodes. Also the barcodes are artifacts that would not be desirable to include in images stored to the perceptual database. Barcodes could used to deriving the properties of some household items, such as groceries though, so the capability should not be totally dismissed. Given that this was undertaken as a proof-of-concept, these limitations were acknowledged as not producing an ideal solution. It is proposed that functional system should use RFID tags instead of barcodes. Currently RFID tags are still too expensive to use on every item within the home. (Though tagging select items with proper storage location could make sense with more expensive tags for a home organization system)

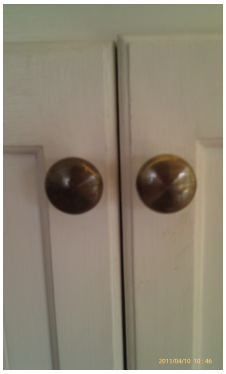
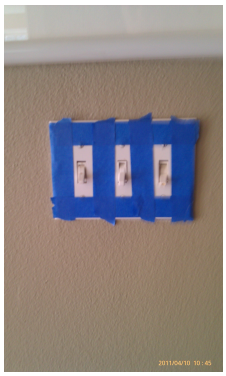
At times the phrases understood by the speech recognizer were amusing and not wholly accurate. The was unable to recognize the word sink used alone. When used with other words it was able to translate it properly. All other words were recognized by the speech recognizer, though some less reliably than others. Words that cause the most problems include: plate, and drawer. The language parsing I used for this system was crude, and didn't understand some variations in grammar or command variations.

Future Work

First and foremost, port the system to Java. After a certain point developing an application with App Inventor becomes unwieldy. It is not possible to do the most basic of tasks, such as searching for a variable name or copying code from one program to another. It does not support local variables, object oriented design, or fine grained program flow control. The fact that it does not permit the implementation of good programming practices made it a very unproductive environment to work in. Also using App Inventor meant that data services were limited to using the TinyWebDB, which has very limited functionality. The API for the Protege ontology system is in Java, another point I wish I'd known when beginning this project.

Object recognition

Intrigued by the potential of using the AR systems with the Android, I conducted a small experiment using Junaio. The Junaio site touts sophisticated object recognition technology that can be associated with custom content. Currently this content is served through a Junaio channel using their UI, but there are plans to allow third party applications to access and serve up data through Junaio. As a proof of concept as to the viability of using Junaio image recognition as part of a robotics system using the Android, pictures were taken of several objects that a household robot would be expected to interact with. The images used for testing are shown below.

The Junaio was set up to recognize the the objects when the phone was active on the Junaio channel I created. The center image was recognized most consistently and was able to be generalized to other cabinets with the same configuration. Recognition for the single knob was somewhat less reliable, but was also able to be triggered from multiple drawers. I expected the marked switch plate to me the easiest to recognize due to the painters tape outline, but despite several trials, the image was not recognized.

Integration with KnowRob

As indicated above, the KnowRob Map is an ontology that is used to provide autonomous robots with semantic knowledge that can be linked to real world objects. Integration to KnowRob or an alternate knowledge base would be a requirement for a functional system.

Tracking of additional properties and state information

The lack of an efficient programming platform meant all planned functionality was not implemented as envisioned. A more complete solution would track the state of objects such as where the object was last found, full/empty, clean/dirty, open/closed, on/off, etc.

Interface camera to web monitoring application

A suggested enhancement to the monitoring application would be to stream camera images to the monitoring application. This is not possibly using App Inventor.

Barcode Interface to the Web

The KnowRob package has a utility that takes an EAM number and that is used to query upcdatabase.com. upcdatabase.com returns html that is parsed for item information which is used to create a class in the KnowRob ontology.

RFID

To be realized as envisioned, RFID should be used to store and track objects in the environment, as it is a more functional and less obtrusive means of encoding and tracking items.

Indoor Tracking and Navigation

Indoor tracking and navigation systems that utilize radio signal fingerprints are emerging technologies that show promise, and have the advantage of not requiring additional hardware. Though they appear to be close to release, they are not yet widely available.

Conclusion

A distributed architecture for a home-based robot is a viable option with many advantages. Amongst them are the ability to use cheaper components and devices. share code and knowledge, semantic web, cooperation, persistence, easy to upgrade and swap components in and out, remote access, growing area, practical.

For those of us without access to a full blown robotics lab, the Android is good option to consider for use as a robotics control platform. Its wide availability, multitude of sensors, extensibility, low cost, and myriad communication options make Android a viable robotics platform.

Experience - about me

I am a User Experience Architect for a large financial services company and am currently enrolled in the HCI Online Masters program at Iowa State. My areas of interest are usability, design, and information architecture. Prior to my work in UX, I worked for many years as a software engineer. Though I no longer code for a living, I have attempted to keep my skills intact by taking an occasional course for continuing education. Professionally I have coded in Fortran, C, C++, and C#. My coursework has included Java and Python. I do not have any direct experience with vision systems or image processing, and have not done much web-based programming. I have worked with numerous API's over the years and am comfortable digging in to figure stuff out. I have

worked in with barcode readers, smart card systems, CAD systems, and as a database architect, though all of that was eons ago. For the record I hate housework and would be first in line for a household bot.

REFERENCES

Coradeschi, S. and Saffiotti, A. Symbiotic robotic systems: Humans, robots, and smart environments.IEEE Intelligent Systems, 21(3):82–84, 2006.

Gates, B. (2007-01). A Robot in Every Home. Scientific American.

Gellersen H., et.al (2000). Sensor-based Context-Awareness for Situated Computing. TecO, University of Karlsruhe, Karlsruhe, Germany.

Gibson, J.J. (1977). The Theory of Affordances. In Perceiving, Acting, and Knowing: Toward an Ecological Psychology. Hillsdale, NJ: Lawrence Erlbaum.

Harnad, S. (1990) The Symbol Grounding Problem. Physica D:Nonlinear Phenomena, 42:335--346.

Hasegawa T., et.al (2007). Robot Town Project: Sensory Data Management and Interaction with Robot of Intelligent Environment for Daily Life. The 4th International Conference on Ubiquitous Robots and Ambient Intelligence.

LeBlanc, K. and Saffiotti (2007), A. Issues of Perceptual Anchoring in Ubiquitous Robotic Systems AASS Mobile Robotics Lab, ¨Orebro University, ¨Orebro, Sweden.

Norman, D. A. (1988). The psychology of everyday things. New York: Basic Books.

Pangercic, D., et.al (2010). Combining Perception and Knowledge Processing for Everyday Manipulation. Perception (2010) Issue: Section III, Pages: 1065-1071.

Saffiotti, A. and Broxvall, M. (2005). PEIS Ecologies: Ambient Intelligence meets Autonomous Robotics. Proceedings of the sOc-EUSAI (Smart Objects and Ambient Intelligence) conference Grenoble, France.

Saffiotti, A. and Broxvall, M. (2008). Affordances in an Ecology of Physically Embedded Intelligent Systems. AASS Mobile Robotics Lab, ¨Orebro University, ¨Orebro, Sweden.

Tenorth, M. and Beetz, M (2009) KNOWROB: knowledge processing for autonomous personal robots. IROS'09: Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems (2009), Pages: 4261-4266.

Tenorth, M., et.al (2010). KNOWROB-MAP – Knowledge-Linked Semantic Object Maps. Proceedings of 2010 IEEERAS International Conference on Humanoid Robots, Pages: 430-435.