INTELLIGENT HOTEL ROOM ASSISTANT

Balázs Tusor
Integrated Intelligent Systems Japanese-Hungarian Laboratory, Budapest, Hungary

Annamária R. Várkonyi-Kóczy
Institute of Mechatronics and Vehicle Engineering, Óbuda University, Budapest, Hungary

Keywords: Intelligent space, Virtual room application, Machine learning, Smart environments.

Abstract: Recently, the usage of smart environment systems has become popular, in order to make everyday living more comfortable and to improve the quality of human life. Intelligent Space (or iSpace) based systems are good examples: they strive to be comfortable and easy to use, even without demanding technical knowledge from their users. However, their aim is not limited to this: in fact, their ultimate goal is to achieve an intelligent environment for higher quality, natural and easy to follow lifestyle. The goal of this paper is to present a research that focuses on developing an Intelligent Space application that is able to comprehend, interpret and execute the detected and pre-processed commands given by human users. The presented solution is also able to learn commands that are given periodically under specific conditions and execute them if the conditions occur.

1 INTRODUCTION

Today, computers play an increasing role in our everyday life. The applications of intelligent systems that aim to improve the living conditions and quality of everyday life are also gaining more and more importance. In ideal case, these systems are realized in such a way that the usage of the systems becomes as easy as possible, while the presence of the system does not bother its user at all. This leads to the basic idea of “ubiquitous computing”, proposed by (Weiser, 1991).

One example for ubiquitous computing applications is the Intelligent Space (iSpace) (Lee et al., 2000), which has been developed at the University of Tokyo. ISpace is a room or area that has built-in intelligence: it can monitor the events and actions taking place in the room or area, it can comprehend human interactions, and it is able to react to them. To mention some examples, it can share information with the user, help in orientation, or anticipate crisis situations. The user can also give commands to the Intelligent Space to use certain services. Therefore, the system should be easy to use for the people in it, without the need for them to learn how the system is to be used.

The most characteristic feature of the iSpace is that the intelligence is distributed in the whole space, not in the individual agents. The main advantages of this component based architecture are that the iSpace can easily be constructed or modified by the installation or replacement of so-called Distributed Intelligent Networked Devices (DINDs) responsible for monitoring a part of the space, processing the sensed data, making local decisions, and communicating with other DINDs or agents if necessary. The agents in the space do not have to possess any complex logic.

Any room or area can be converted to an Intelligent Space by installing DINDs into it. Although, when building iSpace into an existing area we have to keep in view that the system should be human centered, should not be disturbing for the people who are using it and the installation should not alter the area overly.

There are several iSpace applications currently developed or planned. These include the positioning and tracking of humans, the localization of mobile robots, the control of robots, finding paths for them by using itineraries taken by people, etc. (see e.g. (Lee and Hashimoto, 2000), (Appenzeller et al., 1997), and (Lee et al., 2004)).

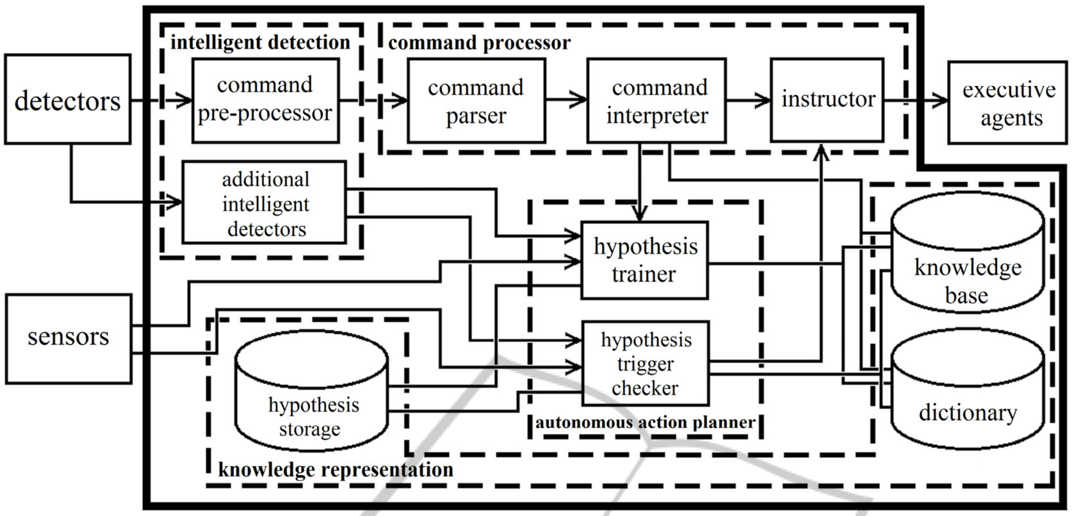


Figure 1: The architecture of the system.

In this paper, the authors focus on a different approach fitting into the frame of the iSpace. This application covers learning and satisfying varying daily routine actions of different people. A typical example can be an intelligent hotel room assistant that provides services to make the daily life of the people inhabiting the room more comfortable.

To achieve this goal, several problems need to be solved. First, the *knowledge* has to be represented in a flexible and easy to process way. Secondly, the commands given by users need to be *detected*. Thirdly, the detected commands need to be *processed* and *executed*. Lastly, frequently repeated commands need to be *learned* and *executed* autonomously. Thus, the system can be divided to 4 major parts; each part solves one of the problems mentioned above.

The rest of the paper is organized as follows: In section 2 the authors present an overview of the proposed system. Section 3 gives more specific descriptions about the modules of the system. In section 4 the current implementation of the system is presented. Section 5 draws the conclusions about the current state of the system and presents ideas for future work.

2 THE INTELLIGENT HOTEL ROOM ASSISTANT

Figure 1 shows the architecture of the system. The inputs for the system are provided by detectors and sensors. Former are devices which are capable of detecting audio or visual commands given by human users, latter are simple devices which can measure simple properties of the room, like its temperature.

Executive agents are electric devices that are able to satisfy the needs of the user. They have just enough built-in intelligence to be able to do their pre-defined task. The iSpace is connected to them as well.

The system consists of 4 parts. The first major part is *Knowledge Representation*, which defines the way how knowledge is described and stored in the system. The second major part is *Intelligent Detection* part, which detects the command given by human users and prepares it. It also detects additional information. This part is designed to be added or be replaced by a new module easily. The third part is the *Command Processor* part. It processes the command and orders the appropriate executing agents to carry it out. The last part is the *Autonomous Action Planner*, which is responsible for the learning and trigger-based autonomous actions of the system.

3 THE MAJOR PARTS OF THE SYSTEM

3.1 Knowledge Representation

The knowledge representation part consists of the *knowledge base*, the *hypothesis storage* and at least one *dictionary*.

The *knowledge base* represents the knowledge of the system. It is realised as a graph-based structure, where the nodes denote the known abstract objects and concepts. The directed edges between them describe their relationship.

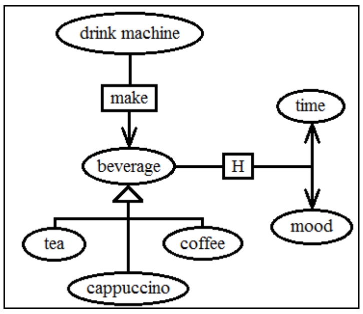


Figure 2: Structural example for the knowledge representation in the knowledge base.

The nodes are homogenous and nameless. For the sake of identification *dictionaries* are used, which assign words or expressions to nodes. In the current implementation only one (English) dictionary is used, although the concept is designed for the usage of multiple dictionaries. This way the knowledge base can be kept independent of natural languages.

The different types of edges are the following:

- **Ability Edges:** node *A* is connected to node *B* with label *s*, if *A* can do *s* to *B*. For example, *A* = „*drink machine*", *B* = „*beverages*", *s* = „*make*" means „drink machine can make beverages". In figure 2 it is denoted with its *s* label.
- **Instance Edges:** an instance edge assigns the address of an executive agent to node *A*. For example, *A* = „*drink machine*" and the assigned address is the address of the drink machine. This is a flexible way to bind an executive agent to the node that describes it.
- **Meta Edges:** node *A* is connected to node *B*, if *A* has a numerical value in *B* quality. For example, *A* = „temperature", *B* = „value" and the value is „20" (which means „the value of the temperature is 20"). Another example, to describe that the minimum value of the temperature is 0: *B* = „*minimum value*" and the value is „0". With the application of meta edges environmental variables can be appointed, which can store the values of the sensors connected to the iSpace.
- **Heuristics Edges:** node *A* is connected to node *B*, if *B* heuristics can be bound to concept *A*. *B* is usually an environmental variable. For example, actions done to „curtain" are dependent of „time". It is denoted by an arrow with label H.

  Inheritance Edges: node *A* is connected to node *B*, if *A* is a *B*. For example, "coffee is a beverage". *A* inherits the properties of *B:* for example, the heuristics and ability edges connected to *B*. It is denoted by a triangle headed arrow.

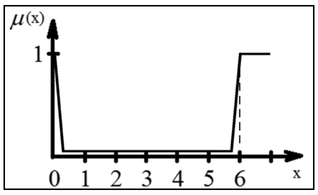


Figure 3: The fuzzy membership function for *Saturday*.

- **Fuzzy Edges:** similar to meta edges, but with fuzzy sets instead of numerical values. For example, *A* = „*Saturday*", *B* = „*day*" and the assigned value is given by the membership function that can be seen on figure 3, for input value x. (x = 0..6, assuming the days of the week start with Sunday (with index 0)).

Figure 2 shows a structural example for the realization of the knowledge base and figure 4 shows an example for the application of meta edges. The meta edges connecting nodes *minimum value* and *maximum value* to node *day* store the possible minimum and maximum value of the day environmental variable, the edge between node *value* and node *day* stores the current value of the parameter, similarly; while the edge between nodes *granularity* and *day* stores a value that remarks how sensitive the variable is to change.

These 4 nodes are connected to specific nodes like *mood* and *time*, similarly with the appropriate values, thus making them environmental variables.

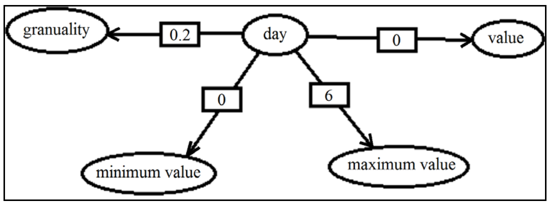


Figure 4: The application of meta edges.

The system creates and manages hypotheses based on the interpreted commands given by the user. The hypotheses are stored in the *hypothesis storage*. A hypothesis consists of (pointers to) the action and subject nodes. It also can have optional numerical values, similarly to the command the hypothesis is based on. A hypothesis also has at least one trigger, which consists of a justification value that denotes how reassured the system is in the trigger of the hypothesis; and at least one condition. A condition is derived from an environmental variable: it consists of a condition node (the node of the environmental variable), value (the value stored in the meta edge connecting the environmental variable to node „*value*”), *sensitivity* (which can be derived similarly to value using node „*granularity”* and the appropriate meta edge), and *affirmativeness*, which is a Boolean value. If its value is false then the condition is used as if it would be negated. (Thus the trigger will be triggered only if the condition is not fulfilled.)

3.2 Intelligent Detection

The Intelligent Detection part consists of *Command Pre-Processor (CPP)* modules and *Additional Intelligent Detector (AID)* modules.

The task of the CPP modules is to detect the command given by human users and transform it into a whole sentence (for example, “open the window”). The methods used to achieve this depend on the type of the available detectors. For example, for voice commands microphones are required, and to be able to give command to the iSpace via hand gestures (so users suffering from hearing impairment or muteness can communicate with the system as well via sign language) one or more cameras are necessary. For the latter case, the hand gesture detection and recognition system proposed by (Várkonyi-Kóczy et al., 2010)can be used.

The task of the AID modules is to provide additional information to the system, like estimating the emotional state of the human users.

3.3 Command Processor

The modules of the Command Processor analyze the given command and order the appropriate executive agents to carry it out. These modules are the Command Parser Module, the Command Interpreter Module and the Instructor Module.

3.3.1 Command Parser Module (CPM)

The task of the CPM is to determine the type of the command and parse it with regards to its type. Its input is the pre-processed command in the form of a whole sentence and its output is the parsed command.

Two different types of commands are distinguished: instructions and prohibitions.

Instructions are simple commands, which are given by the user to achieve change in the environment. The first part of instructions describes the action that is needed to be executed, while the second part describes the subject of the action. After that there can be optional numerical values.

Prohibitions are commands that are given by the user to bound one or more commands that were already learned, thus achieving change in the hypotheses. Their structure is similar to instructions, with the difference that they start with DO NOT and have additional (at least one) text parameters, which represent fuzzy sets. For example, in case of „DO NOT make coffee on Saturday” prohibition command „Saturday” means the fuzzy membership function that can be seen in figure 3 (the input value can be retrieved from the meta edge connecting nodes *value* and *day*). Since the *granularity* of Saturday is 0.2, early Sunday morning and late Friday night count as “almost Saturday”.

Since the input is a whole sentence in predefined format (since English language is quite strictly defined), the parsing algorithm is simple. For example, in case of instructions the first word is always the action and (after the removal of occurrent articles, like “the”) the second one is the subject of the command.

3.3.2 Command Interpreter Module (CIM)

The task of the CIM is to determine which executive agent can execute the command. Its input is the parsed command and its output is the address/reference of an executive agent.

The algorithm of the command interpretation is applied only if the command is an instruction, since prohibitions are not needed to be executed. First the algorithm needs to find the node corresponding to the subject of the command, with the usage of the dictionary. Then if there is such node, the next step is to find out what can do the action of the command to the previously found node. This is done by searching among the action edges. If there is one and it has an instance (there is an instance edge to it), then the algorithm has found the executive agent that can execute the command. If there is none (or there is, but it does not have any instances) then the algorithm looks at the ancestors of the subject node (via inheritance edges), recursively searching for an ancestor with at least one instance. If there is at least one, then the algorithm returns the address/reference of the executive agent bound to that ancestor. Otherwise, the command cannot be executed.

3.3.3 Instructor Module

The task of the Instructor Module is to instruct the executive agents chosen by the interpreter to execute the task to provide the desired service for the user.

3.4 Autonomous Action Planner (AAP)

The AAP is responsible for the learning of the system via hypotheses and to decide whether or not to take actions according to what the system has learned. It consists of the *Hypothesis Trainer* and the *Hypothesis Trigger Checker* modules.

3.4.1 Hypothesis Trainer Module (HTM)

The task of the HTM is to determine which executive agent can execute the command. Its input is the parsed command and has no outputs.

The algorithm of the hypothesis training is the following:

If the command is an instruction, then search for a hypothesis that has the same action, subject and numerical parameters. Since there can be only one hypothesis like that, it is sufficient to get that one. If there is none, then make a new hypothesis using the parameters of the command and add a new trigger and new conditions with the (environmental variable) nodes connected to the subject node with heuristic edges. There will be as many conditions as many heuristic edges are connected to the subject node. The value and granularity of each condition is derived from the current *value* and *granularity* of the environmental variable. If there already is such a hypothesis, then check its triggers. If there are any triggers with conditions triggered by the current values of the environmental variables, increase the justification of that trigger and end the algorithm. If there is no trigger like that, then make a new trigger using heuristics just like it is explained above.

If the command is a prohibition then search for a corresponding hypothesis with triggers. If there is one (if it doesn't have any triggers, add a new trigger to it), then add a new condition to all its triggers using the negated fuzzy membership function of the prohibition. An example will be shown for this in section 4.

3.4.2 Hypothesis Trigger Checker Module (HTCM)

The task of the HTCM is to frequently check the conditions of the hypotheses in the hypothesis storage. If a hypothesis is triggered, the HTCM sends the command of the hypothesis to the CIM, which instructs the appropriate executive agents to carry out the command.

4 THE IMPLEMENTED SYSTEM

In the current implementation (shown in figure 5)
the system is realised in a virtual room where a virtual man is living his everyday life. He gives commands to the Intelligent Hotel Room Assistant integrated to the room to have his desires satisfied. The system is equipped with detectors that can detect the given command. It also has an *emotion estimator* as an additional intelligent detector (which is still under development; in the current implementation it is assumed that it correctly estimates the mood of the human user).

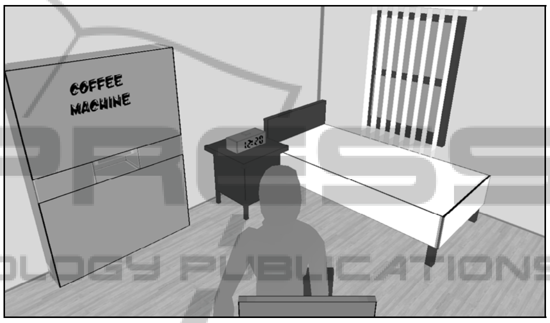
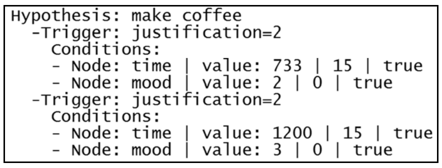


Figure 5: The current implementation of the virtual room.

He gives certain commands in certain times (for example, „open the curtains" after he woke up at 7:00, „make coffee" at 7:33 and 12:00, „turn on the heating" if the temperature is lower than 15 °C / 59 °F, etc.), the system makes and manages hypotheses based on these commands. The justification threshold of the triggers of hypotheses is set to 2, thus the system is only able to give out the command of the triggered hypotheses if the command was detected at least 2 times under the same circumstances.

Figure 6 shows an example for a learned hypothesis based on the instruction „make coffee". The command was given at 7:33 and 12:00 (the parameters after „value" are: the value and granularity of the environmental variable and the affirmativeness of the condition). (The mood of the user was *frustrated* (2) and *neutral* (3)).

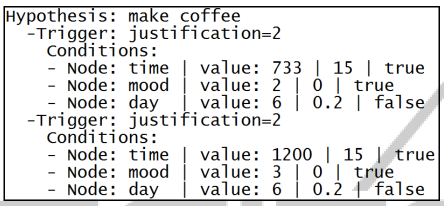


```
Hypothesis: make coffee
  -Trigger: justification=2
    Conditions:
     - Node: time | value: 733 | 15 | true
     - Node: mood | value: 2 | 0 | true
  -Trigger: justification=2
    Conditions:
     - Node: time | value: 1200 | 15 | true
     - Node: mood | value: 3 | 0 | true
```

Figure 6: Example for the instruction command.

His weekday and weekend schedules are different, though the system does not know about this on the first time, so it makes coffee at 7:33 on Saturday. In reaction the user gives a prohibition: "DO NOT make coffee on Saturday."

Thus, the hypothesis based on command „make coffee" is modified by the complement of fuzzy membership function *Saturday* as it is shown in figure 7, which means the system will not make coffee if the value of environmental variable day is 6. (If the user decides after that he wants to drink coffee after all on Saturday, the system will add a brand new trigger to the hypothesis.)



```
Hypothesis: make coffee
  -Trigger: justification=2
    Conditions:
    - Node: time | value: 733 | 15 | true
    - Node: mood | value: 2 | 0 | true
    - Node: day  | value: 6 | 0.2 | false
  -Trigger: justification=2
    Conditions:
    - Node: time | value: 1200 | 15 | true
    - Node: mood | value: 3 | 0 | true
    - Node: day  | value: 6 | 0.2 | false
```

Figure 7: Example for the prohibition command.

5 CONCLUSIONS AND FUTURE WORK

In this paper, an Intelligent Space application is presented that is able to comprehend, interpret and execute the detected and pre-processed commands given by the human user, if the commands are given in a specific form. The system is able to learn repetitive commands that are given in similar conditions with the usage of heuristics and execute them when the conditions are fulfilled without requiring the user to give those commands again. During the simulations of the current implementation the system could learn all the instruction and prohibition commands given by the virtual inhabitant of the virtual room.

The system is able to operate real-time because of the graph-based knowledge base which can be expanded and modified dynamically, thus new sensors and executive agents can be added to the system any time. The system is still under development, there are numerous ways to improve the capability and efficiency of the system.

Complete language independency: The language independency is not complete in the current implementation because of the text parameter of the ability edges. However, appointing nodes for the actions (and binding words to them by the dictionary) and converting the ability edges to three-ended-connections, complete language independency can be achieved.

More flexible command parsing: in the current implementation the command given by the user needs to be in a predefined form, thus the user is limited in giving commands. This limitation needs to be decreased with the application of more intelligent parser algorithms.

Causality: the system should realize the indirect desires of the human user. For example, if the user says that he or she is cold, then the system should realize that the correct course of action is to close the windows if they are open, turn on the heating, etc. To achieve this, the improvement of the Knowledge Base and Interpreter Module will be necessary.

Interaction between the iSpace and the human user: if the system is unsure about giving out a learned command (the justification value of the trigger of the hypothesis is just at the threshold) or there are two very similar hypothesis triggered at the same time (e.g. „make coffee!" and „make tea!"), the system should ask the human user about what actions should it take.

These improvements will be applied to the system in future work.

ACKNOWLEDGEMENTS

This work was sponsored by the Hungarian National Scientific Fund (OTKA 78576).

REFERENCES

Appenzeller, G., J.-H. Lee, H. Hashimoto, 1997. "Building topological maps by looking at people: An example of cooperation between Intelligent Spaces and robots," *Intelligent Robots and Systems*, Vol. 3, No. 7, pp. 1326 – 1333. 1997.

Lee, J-H., H. Hashimoto, 2000. "Intelligent Space," in *Proc. Int. Conf. on Intelligent Robots and Systems, IROS 2000*, Vol. 2, pp. 1358 – 1363.

Lee, J-H., K. Morioka, N. Ando, H. Hashimoto, 2004. "Cooperation of distributed intelligent sensors in intelligent environment," *IEEE/ASME Trans. on Mechatronics*, Vol. 9, No. 3, 2004

Várkonyi-Kóczy, A. R., B. Tusor, 2010, Circular Fuzzy Neural Network Based Hand Gesture and Posture Modeling, In CD-ROM Proc. of the *2010 IEEE Int. Instrumentation and Measurement Technology Conf.,* I^{2}MTC'2010, Austin, USA, May 3-6, 2010, pp. 815-820.

Weiser, M., 1991. "The Computer for the twenty-first century," *Scientific American*, pp. 94-104, 1991.