A SIMULATION AND VISUALIZATION SYSTEM FOR SENSOR AND ACTUATOR DATA GENERATION

Harald Hareter[1], Gerhard Pratl[2], Dietmar Bruckner[3]

[1] harald.hareter@arcs.ac.at, *Information Technologies, ARC Seibersdorf Research GmbH*
[2] pratl@ict.tuwien.ac.at, *Institute of Computer Technology, Vienna University of Technology*
[3] bruckner@ict.tuwien.ac.at, *Institute of Computer Technology, Vienna University of Technology*

Abstract: The evolution of automation systems has reached a point where the increasing complexity requires new mechanisms to keep control of design and operation. In future all control systems will have to cope with massive data quantities that are collected from the environment. Such systems have to react based on the recognition of whole scenarios. In this paper we present a simulation tool, which is able to generate big amounts of sensor data in an animated, virtual environment. This data is used by two different systems to verify their accuracy.

Keywords: Sensors, Simulators, Modelling, Redundancy, Data processing

1. INTRODUCTION

The number of sensors within control systems has increased dramatically in the last years. Nowadays office and industrial buildings are equipped with up to 70.000 nodes (Kastner, *et al.*, 2004). We expect this development to continue, assuming that prices for networked sensors will decrease and the request for more data will increase. In almost the same way the different types of sensors will also increase. In future control systems need to cope with massive data quantities collected from their environment.

Unfortunately, present automation models do not seem to be ready to handle such masses of information. In order to process the expected quantities of data we see the need for new data processing models that have to be integrated into a control and automation system.

Such a system shall not operate on single values, but has to react based on the recognition of whole scenarios. At ARC Seibersdorf research and the Vienna University of Technology two teams are working on two different approaches that should be able to cope with this task. The first one, called Artificial Recognition System (ARS), is based on a biological archetype, which abstracts sensor information to symbols (Pratl, 2005). The second approach, named Building Assistance system for Safety and Energy efficiency (BASE), processes sensor data with statistical methods.

As shown in (Russ, 2003) such systems require massive amounts of data; however, it is not feasible in today's situation to build a real world installation with some thousands of sensors that provides the required quantities of data for proving the concept. Sensor systems containing these amounts of sensors are very difficult and costly to realize; thus simulation is a reasonable alternative to get suitable inputs for studies of the system behaviour. Additionally, simulation may not only predict the behaviour of a model, but might also provide additional information about how to improve an installation later on (Bratley, *et al.*, 1987).

In this paper we focus on the description of a tool, which is able to simulate sensor outputs from millions of sensors and actuators triggered by different kinds of scenarios and to find the optimal amount and placement of different kinds of sensors and actuators. The simulation used here is basically divided in models of local interaction for e.g. motion detection and models of global interaction for e.g. temperature monitoring.

The rest of the paper is organized as follows: in section 2 we describe the software setup that builds the basis for the simulator, section 3 describes the procedures and algorithms of the simulation. Section 4 describes two applications for which the simulator is currently used, in section 5 we demonstrate first results and section 6 rounds up the paper by giving the conclusions and providing a short outlook of further developments.

2. SIMULATION ENVIRONMENT

2.1 Requirements

The goal of this simulation tool is to generate the output of different types of sensors that are triggered by predefined scenarios.

Therefore the tool requires knowledge about the environment, which in the first step should be an office building or parts of it. Furthermore type, position, direction and other physical properties of the installed sensors have to be given. Finally, the scenarios have to be defined. This means that the position and physical properties of objects within the building, posi-

tions, speed and attributes of persons moving around in the building and the change of physical properties like temperature, humidity or lightning have to be specified.
To arrange these settings we developed a scenario generator which is used for sensor placement and activity settings as well as manipulating physical properties in the building. The simulation tool then calculates the sensor outputs and stores these data in a database. This sensor data is used by different applications (see section 4) which, for example, condense data to symbols, which are then also stored in the database. The last part in this system is the visualization tool which presents sensor data and symbols to the user.
Furthermore the operator should be able to interact with the simulator. This requires real time calculation of sensor data and therefore algorithms with reduced calculation effort, an efficiently designed database-schema and a user interface, which is able to provide only data that are currently relevant.

2.2 Learning about the environment

The basis for the simulation environment is a plant layout of a building or a factory hall, which is in most cases provided by the architect of the building. Because of the diversity of architectural Computer Aided Design (CAD) programs and the multitude of file formats in this special field it was impossible to use the plan layout directly. It has to be converted into a suitable file format. We have decided to use the .X-file format defined by Microsoft (Microsoft, 2004) because it is widespread, easy to parse and there are lots of tools to convert other CAD-files into the .X-file format.
The .X-file consists of templates, which contain 3D mesh data, position data, material data and so on for each object (e.g. walls, windows, doors and other content). Furthermore it is possible to define templates for additional data.
Unfortunately the generated .X-files contain only information of objects like walls, windows, doors and so forth which are the building's constituents. It does not contain any information about the building's structure like floors or rooms. We assume that the .X-file contains only information about objects belonging to the building itself and does not include any furnishing objects.
The first step is to extract structural information about the building from the given object information. This is done by dividing the parts to different floors using compare and sort algorithms. Thereafter the parts are combined to rooms by finding neighboring edges. Finally objects, which belong to walls like windows and doors are integrated into the structure.
The information about rooms will be stored in different data structures. The first one consists of two vectors, which contain the location of two opposing corners of a cuboid which is parallel to the axis of the building and encloses a room. The second structure contains information about the centre, the dimension and the direction of a cuboid enclosing a room. The latter is a linked list, which includes all objects belonging to that room. This diversity of data storage is used to save calculation time during different sections in the simulation.
Unfortunately the algorithm is not able to find all necessary structural information about the building itself. Therefore, in a first approach, the scenario generator (described in section 3.1) has to be used to verify, update and correct the information extracted from the plant layout. This information is stored in another .X-file which contains all additional data that is used for the simulation of the building. Additionally, the whole building data is stored in a database to provide the environment information for the applications using the simulation output.

2.3 An object oriented approach

Every item in the building has different kinds of properties like position, size, material and so forth. Therefore an object oriented approach was chosen for internal representation.
We distinguish between different kinds of objects. The global object in the simulation is the building itself which features global properties like time, environment temperature, outdoor lightning and so on. This main object also contains objects like floors, rooms, walls, etc. Additionally, furnishing and people are also represented by objects but do not belong to the building object.
Each object has different kinds of properties to represent its physical conditions. Universal properties like position belong to every object within the simulation. The availability of other properties like temperature or weight depends on the physical model and the required simulator output. If, for example, the physical model in the scenario does not include the calculation of temperature there is no need for managing these properties. That is to say that depending on the physical conditions of the scenario the properties of objects were extended on runtime.

2.4 Physical model and sensor types

In the first step the simulation is able to handle the following types of sensors: motion detectors, light barriers, door contacts, door opener buttons, pressure sensors, temperature sensors and luminance sensors.
This sensor types are divided into sensors which are mainly affected by movement of objects, like motion detection or light barriers, and sensors which are affected by the change of physical properties like temperature or luminance sensors.
This is done because of the high amount of sensors. It is not feasible to calculate the output of each sensor in each simulation step. Another matter for this separation is that calculation of sensor output based on movement can be done by comparing the position of the moving object and the sphere the sensor covers. It does not need any physical model for calculation.
Let's first focus on sensors affected by movement. To reduce computation effort the environment is divided into areas of interest. This means that only regions of the building which contain movable objects are involved in the calculation process. Each area belongs

to a single room, the base is rectangular and parallel to the x-y-Axes and covers the whole height of the room. The base's size is given by the operator (this means that the size is constant for each area in a room) or could be dynamically calculated depending on the size and the shape of the room and the sensors integrated into the room.
Another method of reducing the amount of sensors involved in the calculation is to give the moving objects an area of influence. This means, each object is surrounded by an active area and sensors with active spheres within this area are included into the calculation process. This method turned out to be unsuitable for simulation a big amount of moving objects because of the high effort for updating of areas.
The second kind of sensors like temperature and brightness sensors require a physical model for calculation. These models have to be as simple as possible for fast calculation, but on the other side must be accurate enough to meet the demands of the application. Therefore the models are separated from the main program and were included into discrete modules. These modules were linked on demand and an interface handles the communication between the main process and the modules. This technique allows the use of different models for different requirements to handle the trade off between computation speed and precision. Furthermore this is an easy method to extend the number of the physical models without recompiling the simulator.
The formalism to calculate thermal conduction in a wall using fouriers law, for example, is stored in a module. For more precisee calculation another module based on finite element calculation can be used by linking the temperature model with this module.
A second separation is done by using the time basis of the sensors. Some types of sensors have to be updated more often than others. The refresh times for sensors, which are triggered by movement have to be fast enough to recognize all movement. The change of the room temperature caused by a heating element in the room is slow-going and therefore the recalculation rate can be low. In the simulator the time base for each sensor type could be defined by the operator or calculated by the program. The lower bound for recalculation of fast sensors depends on the amount of sensors and objects in the building. This time is approximated by a calculation step on start-up by repeating a calculation of an active area including an above-average number of sensors for each movable object. Additionally, the time for recalculating the active areas has to be added.
For slow changing properties like temperature the time basis for each room is derived from the given gradient. It is also possible to define local areas within rooms with other recalculation times. These areas were associated to objects like a stove or other heating elements.
During each calculation cycle the output of affected sensors is stored together with a timestamp in the database. In the same way the environment properties as well as the properties of other objects are stored in the database.

2.5 The user interface

One of the main questions in this project was how to present such an amount of sensors and sensor data to the user. We have decided to use different kinds of views of the building and the sensor network.
The basic view is a 3D-view of the whole building in which the operator is able to rotate the building and zoom into different floors and rooms down to single sensors or sensor groups. Starting from this view the operator is able to select parts of the building and handle them as one complete structure (e.g. by selecting a single floor the rest of the building will be hidden). It is also possible to change to 2D-view (for example a top view) or to interactively move through the building by using a camera view. The presentation of the building can be done in solid mode or in wire frame mode. The first one gives a nearly photorealistic picture of the environment including textures and shade effects. This kind of visualization is mainly designed for demo purpose. The second kind gives the object as wire frame only. For an easier differentiation of the miscellaneous objects each object group is represented by a different color.
In addition a tree view of all objects is included. The root is the environment object, which is divided into room and floor objects. The bottom layer consists of sensor groups and sensors. The object's properties can be displayed and if necessary modified.
In 2D or 3D view the operator is able to check the positions of sensors or sensor groups and can hide sensor types that are currently out of interest. If objects in the simulation interact with sensors and the sensor output changes there will be a short highlight at the sensor position to show the change. Other physical properties like temperature pattern are shown by false color representation. Additionally, all sensor data can be listed in table format to get an overview of all changes.
Another feature of the visualization is to display symbols which are derived from sensor data by different applications. The graphical representations of each symbol are integrated into the visualization by overlaying them.
Simulation and visualization do not necessarily have to run on the same computer. On start-up the operator can select the operation mode: simulation only, visualization only, or both. This separation increases the calculation power and decreases calculation time.

3. MODELLING AND SIMULATION TOOLS

Before the simulation process can be started the building has to be equipped with sensors and actuators and the scenarios, which shall be simulated, have to be defined. The operator is able to change properties of the environment at runtime and shall also be able to easily monitor and check the output.
Therefore the whole simulation system consists of a scenario generation interface for sensor placing and scenario definition, a simulation interface and a visualization interface. In the next section a closer look on each of these interfaces is provided.

3.1. The scenario generator interface

As described in section 2.2 the main object in the simulation process is an office building. The structure of the building is provided by a plant layout. Since the algorithms used so far are not able to recognize the building structure completely the scenario generator is used to remodel the structure.
After loading the .X-file the scenario generator offers a 3D-view of the building and a list of all identified objects like rooms and floors. The operator is able to select objects from the list, verify properties and change it. If, for example, a room object contains a wall object, which does not belong to the room the operator can delete the wall object from the room's object list.
The next step is to place equipment into the building. Objects are loaded from an equipment library, which at this stage includes objects like desks, chairs, cupboards and so on. The library objects can be built with CAD or render applications, which are able to export or convert the object information into .X-Files. This library will increase with every scenario which needed new equipment.
After the building is furnished the scenarios can be defined. This includes the definition of placement and moving paths of objects, the initial value and change of physical properties of the environment and objects as well as the interaction between objects. If there are few objects and properties these settings can be done with the scenario generator. A person, for example, could be placed into a room using drag and drop and the movement path could be established by mouse movement.
To provide a slightly easier way for placing a big amount of objects a set of instructions (similar to a programming language) was defined. With this instruction set a placement file can be created with a text editor including all necessary information.
Movement of objects can thus be defined by assigning the starting point, the starting time, the direction and the speed of the object. Another possibility is to define the starting point, start and end time of the movement and the direction. A further variety is to define start and endpoint, the starting time and the speed of the object. In this stage of implementation this works well if the moving object does not leave the starting rooms. To solve this problem, other path finding algorithms have to be implemented.
The change of environment properties can be done in the same way.

3.2. Sensor placing

The applications described in chapter 4 try to perceive information about what is going on in a building by processing data supplied by sensors. Therefore the sensors have to be placed in such a way that they are able to gather a maximum of information out of the environment. In present installations sensors are placed by experts with reasonable know-how and sometimes it is a cost-intensive trial and error process to get an optimal placement. With this simulation tool the optimal placement of sensors can be easily determined. Furthermore, the minimal number of sensors for detecting environment behaviors can be acquired.
As mentioned above this simulation tool is able to handle hundreds of thousands of sensors and therefore a placement by hand is impossible. The vision is that in near future a great number of sensors is integrated into the furnishing, carpets or wallpapers (Vorwerk, 2005). Therefore one possibility to easily place many sensors is to define sensors placed on a grid and assign these sensors to an object. This could be done by selecting an object (or parts of it) in the scenario generator interface, add a sensor property and set the intervals in x, y and z direction. The sensors are than attached to the surface of the object. In this way the floor of a room for example can easily be equipped with pressure sensors.
The placement of motion detectors, door contacts or light barriers needs other methods because they have to be directed to an area of interest. In most cases these kinds of sensors are not so numerous and therefore a placement by hand is feasible. If, for example, every door within the building should be equipped with door contacts an easier workflow has to be designed. There are few thoughts how this could be done, but in the current stage of implementation this features are not built-in. The placement algorithms for sensors on surfaces have also to be overworked because the arrangement of the sensors is not optimal. Primarily on uneven surfaces and on edges the placement could be improved. Therefore the use of other placing techniques like grid based algorithms (Lin, Chiu 2005; Dhillon, Chakrabarty 2003) has to be researched.
Another feature, which will be implemented in near future is to change the granularity of the sensor fields in areas of special interest. This could be done by a simple tessellate algorithm. This algorithm is useful to find the optimal amount of sensors to recognize different effects.
Furthermore the implementation of a virtual sensor is planed. This tool will allow the operator to try out the effect of sensors on different positions without changing the configuration stored in the database

3.3. The simulation and visualisation interface

After the scenarios were created they are loaded into the simulation environment. Here all objects are listed in a tree view and the sequence of actions take place during simulation time are displayed on a time line. Here the operator is able to make changes on start up properties of objects or reschedule the order of actions.
If initialization is finished the operator can chose to start the visualisation and simulation on one computer or source out the visualisation. The IP address of the visualisation computer and the computer hosting the database can also be modified.
During simulation the operator is able to switch between the different types of visualisation.

3.4. Process communication

As mentioned above the simulation and the visualization process do not have to run on the same computer. The data processed by the simulation is stored in a database and the visualization or another application (such as ARS or BASE) gets the data from the database.
Every time the simulation writes new data into the database the other applications have to be informed to fetch these new data. The database is not able to inform the applications and polling the database is inefficient. Therefore we use a synchronization module, which can also run on a separate computer; it handles all the communication between the applications. This means that the simulator sends the data to the synchronization module, which again hands the data to the database and informs the other applications. The applications send a request to the synchronization module and receive the data in return.

4. APPLICATIONS

Two examples shall be given to demonstrate the necessity of sensor input generation in large quantities. The first one is the Artificial Recognition System (ARS), a cooperation project between ARCS Seibersdorf and Vienna University of Technology. ARS makes use of diverse, redundant sensory inputs in a bionic way to create a perception of the surrounding world (Roesener, *et al.*, 2004).
In order to create a system that is able to perceive its environment in a way that is similar to the processes that occur in the human mind, a broad base of sensory input is required. Perception at first depends on vast redundancy. Therefore we need to take all available sensor data and extract the information that is important for the functionality of the system. This data input is condensed by means of *symbolization*. A symbol is a piece of information, for which the system has a concept and is therefore able to process.

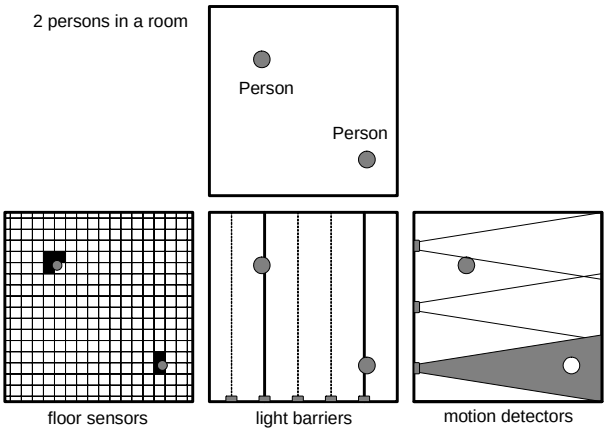

Figure 1: Redundant information is used to determine the position of persons

A simplified example is shown in Figure 1: the top square shows a room with two persons. Pressure sensors in the floor provide information about the position of the two persons. This information is also available from the light barriers and the motion detectors. By using all this redundant information, the ARS system can deduce the current position of each person, even if some sensors cannot contribute to the position, because they are obscured by other objects or they are faulty.
This person surveillance is the basic application that enables other applications built on top of it. These applications are, for example, a geriatric care system that supervises elderly people and recognizes possibly dangerous situations.
Although we expect the costs for sensors to decrease significantly in the next years, we are today unable to provide the amount of sensors that are envisioned in the ARS project. We estimate that it will be necessary to have some ten thousand sensors of different domains in each building. This is the point where we expect classic building automation to be unable to exploit the full potential of the available installation. Therefore we look at such a setup today, however without having a real installation available.
The solution is to use the simulator to apply the bionic model of the ARS project on the simulated environment and the available sensor data. This way is a convenient way to generate lots of different setups with various amounts and types of sensors.

The second example where a large quantity of sensory input is used is the in-house project Building Assistance System for Safety and Energy Efficiency (BASE). BASE makes use of numerous sensory input devices to generate a statistical model of what happens “normally” in a building. The knowledge of normal sensor behaviour is used to predict system behaviour in the near future and to recognise sensor malfunction.
In order to model the building’s behaviour, it is necessary to have huge numbers of sensors. For building up models that are capable of representing useful situations in, for example, an office building, pressure sensors in the ground, light barriers, motion sensors, door contacts and sensors for temperature, light, humidity as well as measures of air-quality and particles in the air are necessary. All factors that cause humans to change their behaviour have to be observed in order to have the chance to predict their behaviour.
An example scenario in a kitchen in a dwelling house with unlikely sensor behaviour could be as follows: The system detects the furniture, a person and the stove because of the pressure sensors in the ground and the behaviour in the past by comparing it with templates for tables, chairs, cupboards and so on. This means the system’s representation of the room is initialized and stable.
In the current example a person lingers in front of the stove for some time and the temperature in that area rises. From previous observations the system knows that in case of the detection of a lot of steam and temperature around the stove, the person turns down the

stove. If, for some reason like in Figure 2, the person leaves the room while cooking and a pot overflows, the system computes very unlikely probabilities for the combination of temperature, steam, stove control and presence and can therefore launch an alarm.
Another possibility is that one of these values is incorrect due to sensor faulty. In this case the system expects different values than it gets and launches an alarm.

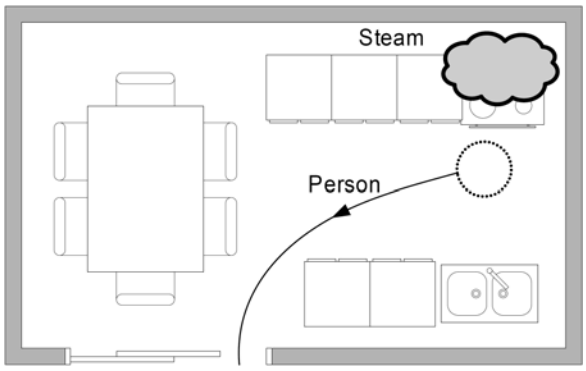

Figure 2: An alarm generating scenario: After some cooking the person leaves the kitchen. One plate is heated too much and therefore steam appeared. The system computes very low probability for the combination of temperature, steam, stove control and presence of a cook and launches an alarm.

5. FIRST RESULTS

The test object is an office building with 6 floors. The floor space on each level is approximately 3200 m² with gives a total area of 19200 m². Each floor consists of 53 to 78 rooms including halls, giving a total of 395 rooms. The mean amount of sensors per room is nearly 1200, the most of them being pressure sensors on the floor. The total amount of sensors is almost 500000.
Each room is equipped with office furniture like desks and cupboards. The scenario consists of 1500 persons moving on random paths through the building. Additionally, the temperature level in each room is different on start-up and the temperature equalization is calculated.
The test system consists of a standard office PC which includes a 3 GHz Pentium 4 Processor,1 GB of Ram and a NVIDIA Quadro PCI-E video card.
Running the visualization interface on the same computer as the simulation tool make the visualization a bit slow in solid mode. This is due to overload of the graphic hardware which has to calculate the shading for each lightning source in every visualization cycle. Switching to wire frame mode let the visualization run more smoothly.
The time basis for fast sensors was fixed to 50 ms which demands 20 recalculations per second. The mean amount of sensors recalculated on each calculation step was nearly 600 with gives a total of 12000 calculations per second. Here the transmission of the data into the database shows a possible bottle neck. The used MySQL-Server was overstrained.
Another weak point of the simulation tool is the memory usage. On runtime the simulation uses up to 520 Mb. A closer look on the memory management is therefore necessary.

6. CONCLUSION AND OUTLOOK

We have presented the concept of a simulation tool, which is able to generate the output of a large number of sensors that are triggered by simulating different scenarios. These sensor data are the basis for the evaluation of new models for future automation systems based on statistical models or biological archetypes. These systems should be able to recognize the simulated scenarios and react in a given way.
Most of the presented concepts were integrated in the simulation so far. Some of them have to be improved or replaced. Some are still concepts and will be implemented in near future. The first steps have been done, and there are some to do to be ready for future requirements.

REFERENCES

Bratley, P., B.L. Fox, L.E. Schrage (1987). *A guide to simulation*,. (Edition 2) Springer, New York

Dhillon, S.S., K. Chakrabarty (2003). Sensor placement for effective coverage and surveillance in distributed sensor networks. *Wireless Communications and Networking*, ***Vol. 3,*** pp. 1609 - 1614

Kastner, W., P. Palensky, T. Rausch, CH. Roesener (2004). A Closer Look on Today's Home and Building Networks, *7th AFRICON Conference in Africa*, ***Vol. 2***, pp. 1239 - 1244

Lin, F.Y.S., P.L. Chiu (2005). A near-optimal sensor placement algorithm to achieve complete coverage-discrimination in sensor networks. *Communications Letters*, ***Vol. 9, Issue 1***. pp. 43 - 45

Microsoft (2004). X File Reference. http://msdn.microsoft.com/archive/default.asp?url=/archive/en-us/directx9_c_Dec_2004/directx/graphics/reference/d3dxfile/xfilereference.asp

Pratl, G (2005). A Bionic Approach to Artificial Perception and Representation in Factory Automation. EFTA 2005

Roesener, Ch. Hareter, H. Burgstaller, W. Pratl, G. (2004). Environment Simulation for Scenario Perception Models, *5th IEEE International Workshop on Factory Communication Systems*, pp. 349 – 352

Russ, G. (2003). Situation-dependent Behavior in Building Automation, PhD, pp 140

Vorwerk (2005). Intelligent carpets as a navigation system for robots. http://www.vorwerk-teppich.de/sc/vorwerk/rfid_en.html