VISION-BASED LOCATION SENSING AND SELF-UPDATING INFORMATION MODELS FOR SIMULATION-BASED BUILDING CONTROL STRATEGIE

A. Mahdavi, O. Icoglu, S. Camara
Department of Building Physics and Building Ecology, Vienna University of Technology, Vienna, Austria

ABSTRACT

Dynamic facility state models could effectively support simulation-based building systems control. In a simulation-based control strategy, the permutation space of control options (alternative states of control devices) at a future time step is proactively explored using computational simulation. The simulated implications of alternative control options are then compared based on users' preferences. From this comparison, the preferable control actions can be deduced. To achieve this functionality, however, the presence of a dynamic and self-updating building model (with context, room, systems, and occupancy data) is necessary. In this paper, we described the conception and implementation of a prototype vision-based object location sensing and occupancy detection system to provide the control unit of a sentient building with a steady flow of up-to-date building state infomration.

KEYWORDS

Simulation-based building systems control, location sensing, building information models

INTRODUCTION

In a "sentient building", a comprehensive sensor-supported dynamically self-updating facility state model effectively supports, amongst other operations, simulation-based building systems control. Sentient buildings possess a dynamic representation of their context, components, systems, spaces, processes, and occupancy (Mahdavi 2004). They can autonomously update this representation based on information from a comprehensive sensory infrastructure. Thus, a sentient building possesses at all times up-to-date and high-resolution information regarding external conditions, room enclosure surfaces, furniture, building components (such as doors and windows), environmental systems, occupants' presence and movements, and other static or dynamically changing building entities. Most importantly, sentient buildings can use this dynamic representation to support operations in facility management and indoor-environmental systems control (for heating, cooling, ventilation, lighting, etc.).

In a simulation-based control strategy, the permutation space of control options (alternative states of control devices) at a future time step is proactively explored using computational simulation. This simulation model is regularly fed with data from the self-updating building and context representation. The implications of alternative control options are then obtained via simulation and compared based on users' preference settings. From this comparison, the preferable course of control action can be deduced (Mahdavi 2001, Mahdavi et al. 2005).

To achieve this functionality, however, the presence of a continuously updated context, room, and occupancy model is sine qua non. In previous publications, we have described and documented approaches for dynamically capturing contextual conditions (weather, solar radiation, sky luminance, etc.) as a component of a building's self-representation (Mahdavi 2004, Mahdavi et al. 2005). In the present contribution, we present the conception, implementation, and testing of a prototype vision-based object location sensing and occupancy detection system. It is designated to provide a sentient buildings model with a steady flow of up-to-date data on the state of rooms (surfaces, system components and interior objects, occupancy, etc.). Such information, together with data on outdoor conditions and system states constitute the information core of sentient building's self-updating representation. As the underlying sensing technology, a vision-based system was selected that functions based on the well-known "barcode reader" principle, using a combination of visual markers (tags) and cameras. It utilizes image-processing methods to obtain in real-time the identity and location (both position and orientation) of the tagged objects. Additionally, cameras are used for motion-based occupancy detection. This provides an extra benefit for the vision-based methods, since it makes the implementation of an additional technology for occupancy sensing unnecessary.

The initial results obtained from our implementation suggest that vision-based sensing, when enhanced computationally and integrated with appropriate hardware, can offer a promising technology for application domains such as simulation-based building systems control.

In the following two sections of the present contribution, we first describe the overall system design prototypical implementation of VIOLAS, a prototype vision-based object location sensing and occupancy detection system that is based on an array

of (in part pan-tilt) network cameras (İçoğlu and Mahdavi 2005). We then report on ongoing modifications and enhancements to the system, which involve the use of non-moving fisheye cameras.

VIOLAS

Technology review

Prior to the implementation of a solution, available technologies were reviewed from the building automation perspective. Our technology review did not reveal a perfect sensing system for self-updating building models. However, vision-based methods appeared to come closest in meeting our requirements, as they are software-supported and open for modifications and improvements. Hence, we selected a vision-based system that functions based on the well-known "barcode reader" principle, using a combination of visual markers (tags) and video cameras. It utilizes image-processing methods to obtain in real-time the identification and location (both position and orientation) of the tagged objects. Additionally, video cameras are used for occupancy detection.

Upon choosing vision-based methods as the one currently most responsive to our requirements, different camera technologies were examined: Network cameras (netcams) posses lower image quality when compared to CCTV (close circuit television) and digital photo cameras. However, in buildings that are equipped with a data transmission network, the required infrastructure for netcams is already in place. A high-end network camera can send images to ten or more computers simultaneously using the HTTP transmission protocol. Moreover, netcams enable the control of third-party devices like pan-tilt (P/T) units, thus effectively increasing the monitoring ranges. Therefore, network cameras were selected as the sensors for the initial implementation. However, ongoing work includes also a test of digital cameras equipped with fisheye lenses as potential alternative sensors in the location-sensing system.

Implementation

VIOLAS can be installed in a room equipped with one camera, and it can be applied to an entire building with one hundred rooms, equipped with one hundred cameras. The latter scenario involves intensive processing loads of data acquired from multiple sensors. To accommodate such loads, multiple computing resources must be utilized that can function in parallel and that can be reconfigured in a scalable fashion. For this reason, VIOLAS software is implemented as a distributed architecture, where the subcomponents of the system communicate through the Internet platform. Communication and data sharing follow the distributed component object model (DCOM) protocol enabling software components to communicate directly over a network in a reliable, secure, and efficient manner (DCOM 2004). This distributed structure provides scalability, incremental growth, and enhanced performance derived from parallel operation. Thus, VIOLAS software is divided into server and client tiers (Fig. 1).

Application server lies on the server tier. It is responsible for two important activities, resource management and data integration. Concerning its resource management activity, application server controls the distributed components, including the sensors and client modules, lying on the client tier. Network cameras are the sensors in this structure. They convey video images as distributed network devices. Client modules are the *image processing unit*s (IPU) implemented on different computers scattered across a facility. They process the input images captured from netcams. Application server is responsible for connecting the IPUs with the available netcams in the system. Concerning the data integration activity, application server combines the results obtained from multiple IPUs, and subsequently transfers them to the self-updating model and the applications served by this model. *Database server* is the second subcomponent that lies on the server side. This component handles the data access demands of other components and connects them with the VIOLAS database. *User interface server* is the last component of the server tier. It maintains the communication between the operator and VIOLAS. The activities undertaken by each VIOLAS component are described in the following sections in more detail.

Image Processing Units

Image processing units (IPUs) are the programs that lie on the client side, and run parallel on different computers scattered around the facility. IPUs are the consumers of sensors (cameras). They acquire images from the cameras and process them to extract the context information (Fig. 2).

Hardware interface

Hardware interface facilitates the communication with the cameras. The required communication parameters are retrieved from a camera table located in the VIOLAS database. In case of pan-tilt units, it also permits controlling the motion of these devices. Thus, the pan-tilt unit is successively guided through a certain path so as to scan the entire space.

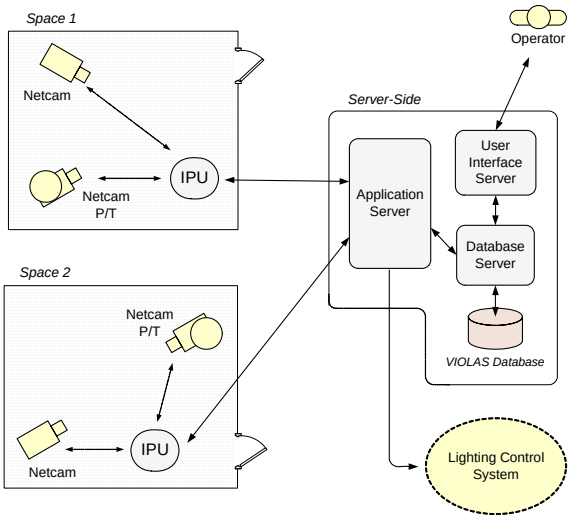

Figure 1 Distributed structure of VIOLAS

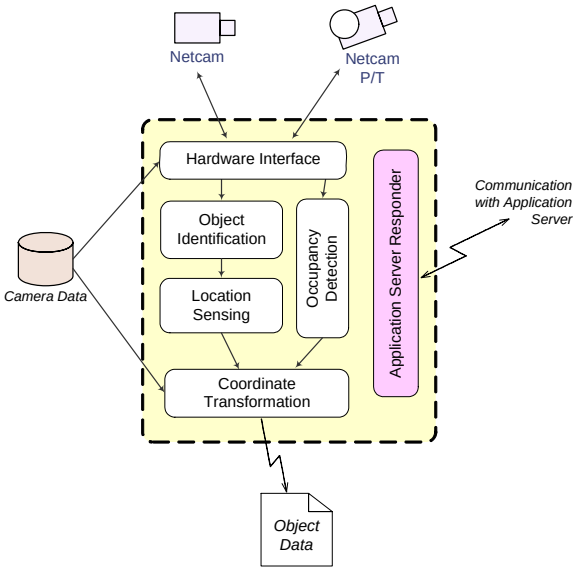

Figure 2 Structure of the Image Processing Units

Object identification and location sensing

Amongst vision-based methods, the algorithm proposed in TRIP (Lòpez et al. 2002) provides in our view a suitable basis for object identification and location sensing in buildings. For location sensing, TRIP uses "pose from circle" algorithm (Forsyth et al. 1991, Trucco & Verri 1998), estimating the pose of a circle in space from a single intensity image. The algorithm can be concisely described as follows: A circle on the target plane generates an ellipse on the image plane of the camera. From the known parameters of the ellipse, it can be back-projected to the original circle, enabling the extraction of the orientation and the position of the target plane with respect to the camera origin. If tags with *reference circles* are attached to the objects, their location can be estimated with respect to the viewing cameras.

In TRIP system, object identification is performed with a method similar to the barcode principle: special marks are placed around the reference circle used for location sensing. We also utilized the same method, but applied a different coding structure. This coding structure enables the definition of 2031 distinct tagged objects. The tag size is 12 by 12 cm. It can be printed using regular black-and-white printers. Thus, tags are low-cost, low-maintenance, and require no power input. As previously mentioned, netcams provide lower image quality (low resolution, less sharp images) than CCTV or digital photo cameras. In order to compensate these drawbacks, image enhancement methods are applied prior to the implementation of location sensing and object identification. The details of the enhancement methods and the location sensing and object identification algorithms utilized in VIOLAS are given in İçoğlu & Mahdavi (2005).

Occupancy detection

In addition to the object location data, occupancy information is needed for a comprehensive model construction. Even though there are off-the-shelf products for this purpose, we explored the potential of our vision-based solution for occupancy detection, thus benefiting from an already established infrastructure. Occupancy sensors typically operate based on detection of motion. We thus implemented a method to detect motion from sequential camera images. The temporal changes in the intensity (brightness) values of the pixels are evaluated to sense the motion within the camera's field-of-view. However, the intensity values may change even though there is no motion. Illuminance variations in the scene may cause false occupancy reports. Additionally, the aperture of the camera is managed automatically by the camera mechanism, leading to intensity changes in image pixels. To deal with these problems, a model was adapted, in which the image intensity is considered to be generated by incident light, which is reflected by the surfaces of the objects in the observed scene. For diffuse surfaces, the relation between observed intensity, illumination, and reflectance is multiplicative, and the images can be decomposed into their illuminance and reflectance components utilizing homomorphic filtering (Toth et al. 2000).

In many realistic cases, a reflectance map contains the object information, whereby the effects of illumination are typically suppressed. Therefore, it is more convenient to utilize reflectance maps rather than the intensity images for evaluating the pixel changes in image sequences. Thus, the raw camera images are decomposed into their illuminance and reflectance components, and subsequently, the reflectance components of the successive images are compared to detect occupancy.

Coordinate transformation

The extracted location data is the position and orientation information with respect to the coordinates of the camera from which the processed image is acquired. In this state, the location data is not usable for model generation. By using 3D transformations, "coordinate transformation" converts the position and orientation data with respect to camera coordinates to the position and orientation data with respect to the real-world coordinates (İçoğlu & Mahdavi 2005). The coordinate transformation is also applied in occupancy detection, if the camera is mounted on a pan-tilt unit. The position of the camera is known with the provided *pan* and *tilt* angles. Together with the camera's field-of-view range information retrieved from the camera table, this information allows for the determination of the region within which the occupancy is detected. The calculation does not provide a precise location data, but it allows for the spatial derivation of the occupancy region. Once the context information is obtained via IPU, the results are conveyed to the application server (Fig. 1).

Application Server Responder

Located within the IPU, "application server responder" is an auxiliary module that provides the communication to the application server. The tasks undertaken by this module are as follows: *i)* Register the IPU in the system when it is initiated; *ii)* Respond to the application server's "control" request sent for checking the status of the IPUs. The application server obtains the status of the IPUs (active or not) by sending this request message; *iii)* Fetch the application server's "new resource sharing" request. The cameras assigned for the IPU are informed through this message. Thus, the IPU recognizes the cameras to be connected for image acquisition. Note that, at the startup of the IPU, no cameras are assigned. "Application server responder" first registers the IPU in the system. After recognizing the new IPU client, application server rearranges the assignments between the cameras and the IPUs. Consequently, the application server informs all IPUs, whose camera assignments are changed by sending a "new resource sharing" request. This is how the IPU lines up its cameras at the initial state. Subsequently, the IPU connects to the designated cameras and applies the aforementioned sensing methods to the images acquired.

Application Server

As the key component in VIOLAS, the application server controls the overall system. First, it manages data integration by combining the results coming from parallel running IPUs. Second, it dynamically performs the assignment of resources to consumers. In other words, the application server detects changes in the status of the cameras and IPUs in the system, and accordingly rearranges the assignments of cameras to the existing IPU clients. While performing these assignments, application server also balances the workload of IPUs.

The structure of the application server is illustrated in Figure 3. It is comprised of five distinct modules. The first three modules undertake the resource management activities, whereas the last two modules implement data transfer and integration.

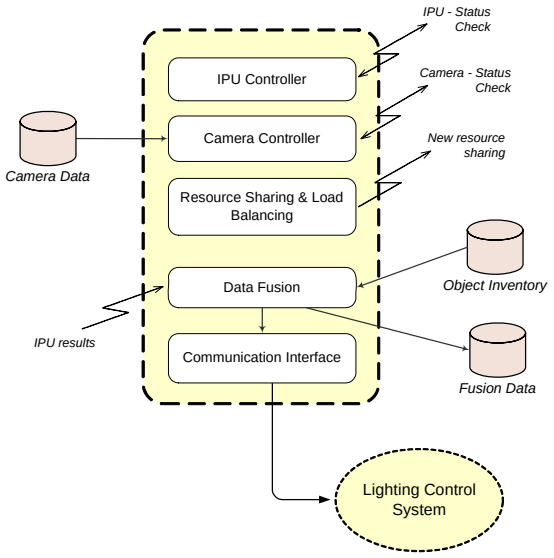

Figure 3 Structure of the application server

Resource management

It is not feasible to assign an operator to continuously and manually manage the large amount of distributed components (cameras and IPUs). To support resource management in VIOLAS, the three following modules are implemented.

The first modul, "IPU controller", continuously checks the state of IPU clients. It sends a "control" request to each IPU (see also "application server responder" in Fig. 2). The IPU's status is assumed as "active", if it responds within a certain time.

The second module, "camera controller", checks the status of the cameras in a similar way. It reads the available cameras and their communication details (Fig. 3) from the camera table, and then sends a "control" request to each of them. The netcams have a built-in FTP server. If the camera's FTP server responds within a certain time, the camera's status is updated as "active".

The third module, "resource sharing and load balancing", is executed at the initial run of the system and whenever the status of cameras or IPUs change. This module assigns the active cameras (resources) to the active IPUs (consumers). This assignment is performed in a manner so as to continuously balance the workloads of IPUs. Thus, the application server

computes the number of cameras that will be assigned to each IPU in the next run. In addition to the number, the "resource sharing and load balancing" module also determines which cameras will be assigned to a specific IPU by taking the network domain of the cameras into consideration. The cameras that lie on the same network area or on a network area close to the IPU are assigned to reduce the network load and increase the image transfer speed. Consequently, multiple IPUs can share a single camera, when the number of processing units exceeds the number of available sensors. This arrangement prevents jobless IPUs. In order to eliminate inconsistencies, the application server selects one of the IPUs as the "master". The master IPU is authorized to control the pan-tilt unit if one is available for the shared camera.

Data integration

Data integration is vital for handling multiple results coming from concurrent IPUs. Towards this end, "data fusion" module is implemented. This module is activated regularly, whereby the context data acquired from all IPUs are combined. There are two phases of the data fusion, namely "tag-level fusion" and "object-level fusion". The same tag can be detected with more than one camera. Likewise, one camera can be assigned to multiple IPUs. This leads to repeated tag records coming from multiple cameras (or IPUs) in the system. In the tag-level fusion phase, data fusion combines these records. The second phase of the data fusion is implemented at the object-level. The system enables the attachment of multiple tags on a larger object to reduce the occlusion possibility and to increase the line-of-sight between tags and cameras. This requires the second level fusion in order to prevent redundant object records when both tags are identified.

In addition to location, occupancy data is considered in data fusion. IPU assigns time stamps to detected occupancies, as it is done for tag identifications. Based on the detection times, occupancy durations are determined. The ones that exceed a predefined time interval (elapsed since the last occupancy detection report) are removed. Consequently, the output of the data fusion (the final and consistent context data) is stored in a fusion table in the VIOLAS database. The results are also transformed into XML-like data packets for convenient data communication. Thus, they can be transferred to operational applications (such as the office lighting control system in our project) through the "communication interface" module.

User Interface Server

User interface server provides the communication between VIOLAS and the operator. It enables an operator to add and modify system data and to see system results on screen. The location values retrieved from the fusion table are combined with the values retrieved from the object inventory, and 2D graphical representations of the objects within the room are generated. Towards this end, the system represents every object as a minimum bounding-box covering the original object shape.

The communication is achieved by a method called common gateway interface (CGI). CGI programs are special applications that can run upon the initiation of a web server. This method allows for the remote execution of programs on the server platform, eliminating any hardware or software requirement on the client side. The operator can run the program through his/her web browser, from any computer on the network.

Database Server

Database server is an important module in VIOLAS, as its underlying structure enables the access to distributed COM (component object model) objects over a network. COM is a software architecture that standardizes the programming interfaces, implementation models, and the data structures. Based on this architecture, distributed COM objects (DCOM 2004) are developed that can be employed by other applications remotely in a network environment. This allows the software components to access data or to implement a function remotely from multiple distributed computers. In this context, the software components of VIOLAS (image processing units, application server, and the user interface server) act as client applications that request service from the database server. Database server manages the communication between these programs and the VIOLAS database. The programs can also communicate with each other utilizing the database as a broker. Note that simple text messages between the applications server and the distributed components (for camera and IPU status-checking or new-resource-sharing) are carried out with TCP socket connection rather than DCOM.

A DEMONSTRATIVE TEST

Test-bed configuration

To evaluate the performance of VIOLAS, a demonstrative test was performed in an office space of the Department of Building Physics and Building Ecology, Vienna Technical University. The building is equipped with a local area network, thus offering the required infrastructure. A netcam attached on a P/T unit is connected to the network. Within this network, our department possesses a domain via a mainframe computer. The server applications (database server, application server, and user interface server) are installed in the mainframe together with the VIOLAS database. This computer

also runs a web server. The user interface server is placed under a specific folder where the web server can access and execute CGI applications. Prior to the initialization of VIOLAS, required parameters are defined via user interface server. However, the database server is activated first so that the user interface can access the database.

Towards the implementation of the test, a typical office environment (test-bed) is used that involves 25 objects relevant for the demonstrative operational application (lighting control system). First, these objects are defined. For each object, a tag is generated. Consequently, the tags are printed and attached on the corresponding objects. Figure 4 shows the 2D sketch of the entire test-bed together with the tagged objects and sensor devices. As also depicted in the figure, motion was generated at three points to test the occupancy sensing capability.

VIOLAS in operation

Following the configuration, application server is initiated. Application server immediately starts checking the status of the cameras and the IPUs in the system, and finds the netcam connected to the network. However, the camera is not assigned to an IPU, since no IPU was activated before. Therefore, an IPU is initiated from a computer in the department. After its initiation, the IPU client registers itself to the system and responds to the application server's control request. Detecting an active IPU client, application server executes a resource management computation and assigns the netcam to the IPU. To test the system's behavior with multiple clients, a second IPU is initiated from another computer. This IPU also registers itself successfully in the system, and responds to the application server's control request. After the detection of a second active IPU, application server enacts a reassignment, linking the camera to the IPUs, whereby the second IPU is given the "slave" role. Thus, the first IPU controls the pan-tilt unit attached to the network camera. Consequently, both IPUs successfully run in parallel and scan the test-bed. They extract the context information in the test-bed and concurrently transfer the results to the VIOLAS database.

The output of the IPUs is fetched by the application server. The incoming results are fused by the internal data fusion module, and subsequently recorded in the database. In this test, the system achieves a 100% identification performance, extracting all tag codes and recognizing all objects. To evaluate the accuracy of location results, "position error" is defined as the distance between the ground-truth position (actual position information) and the sensed position of the tag. "Orientation error" is defined as the angle between the tag's true surface normal and the sensed surface normal. Generally, the test implies for the system an average position error of 0.18 meters and an orientation error of 4.2 degrees on aggregate. The position error percentage has a mean value of 7.3% (cp. İçoğlu & Mahdavi 2005).

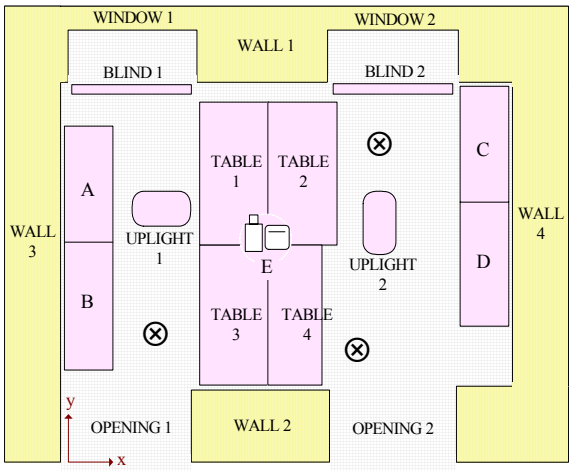

Figure 4 2D sketch of the test-bed ("A" to "D" refer to Cabinets; "E" refers to Camera and P/T unit. The marks, ⊗, refer to the occupancy events)

Occupancy in the test-bed was detected consistently and without false reports. The system is robust against illumination changes due to opening or closing of the aperture in the camera mechanism. Likewise, slow changes in the intensity of the light sources do not cause false reports. As mentioned before, the system does not provide an exact occupancy location. However, the general region within which the motion takes place can be specified. A graphical representation of the test-bed, as generated and displayed by the user interface, is illustrated in Figure 5. The object location results can be seen together with the sensed occupancies.

MODEL VISUALIZATION AND REUSABILITY

As seen in Figure 5, VIOLAS can present the results in a 2D display. However, this display is not very convenient for the visualization of the resulting model. A more proper visualization can include the 3D display of the model, and enable the model's examination by an operator with rotation and walk-through capabilities. Towards this end, VRML (virtual reality modeling language) function is added to the user interface server. A sample VRML model of the test-bed constructed by the user interface server is shown in Figure 6. Moreover, VRML enables the reusability of the models generated by VIOLAS. Other 3D model editing applications can import the VRML files. This allows the user to modify and store the VIOLAS outputs in different platforms offered by commercial CAD vendors.

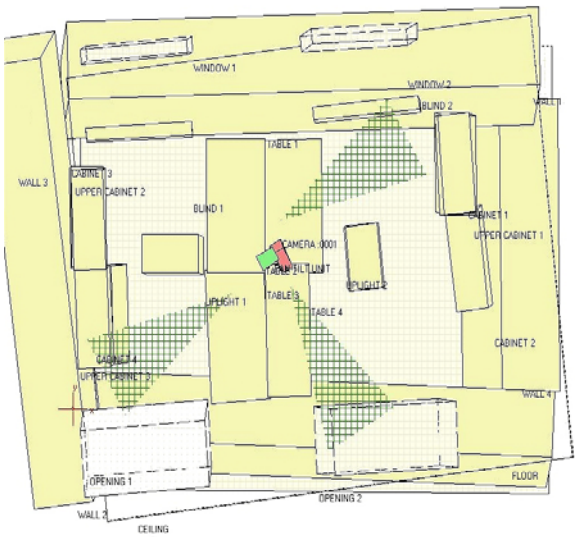

Figure 5 Graphical representation of the test-bed generated by the user interface server. The objects are drawn with the extracted locations. The slides are caused by orientation errors that have an average value of ~5°

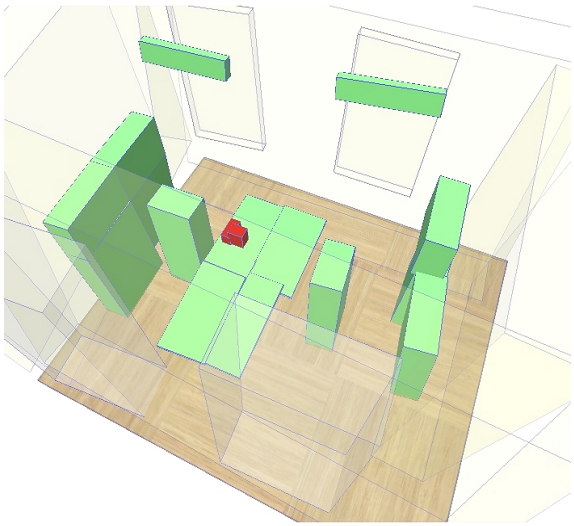

Figure 6 3D model of the test-bed (VRML)

ONGOING WORK

The initial VIOLAS implementation was based on network cameras. To achieve the required level of scene coverage, most of such cameras in a facility need to be augmented with pan-tilt units. While our experience with this technology has been promising, certain problems exist, such as the intrusiveness of this technology both in terms of the camera size and the existence of moving parts. Thus, to explore an alternative that would not involve moving parts yet would offer a wide scene coverage, we are currently testing the potential of static cameras with fisheye lenses as the primary visual sensing device. Toward this end, we have performed an initial test, whereby, other than the cameras, all other components of the previous implementation (tags, detection algorithms, test space) are unchanged. The test involved the following steps: *i)* we equipped an ordinary digital camera with a fisheye lens; *ii)* we mounted this camera in the center of the test space. Altogether 17 tags were used to mark various room surfaces and furniture elements; *iii)* four fisheye images of the test space were generated by the camera from four different vantage points close to the center of the room (see Figure 7 as an example); *iv)* These four images were dissected into nine partially overlapping segments (see, for example, Figure 8) via the application of an equi-rectangular projection technique (Iris 2006); *v)* the resulting image segments were analyzed using the image processing methodology described in the previous section.

Figure 7 Sample fisheye image of the test space

Figure 8 Examples of image segments extracted from the fisheye picture using equi-rectangular transformation (Iris 2006)

Figure 9 shows the tag detection performance in terms of the percentage of correctly identified tags as a function of the distance of the tag from the camera. Figure 10 shows the relationship between the actual and computed tag-camera distances. Even though this

initial test resulted in a rather modest tag detection performance (67 %) and distance estimation accuracy (6 ± 10%), further calibration of the camera and improvements in the application of the equi-rectangular projection are likely to improve the performance of the system in the near future.

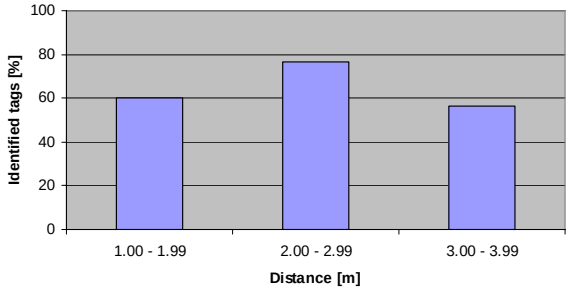

Figure 9 Percentage of correctly identified tags in a scene captured as a fisheye image

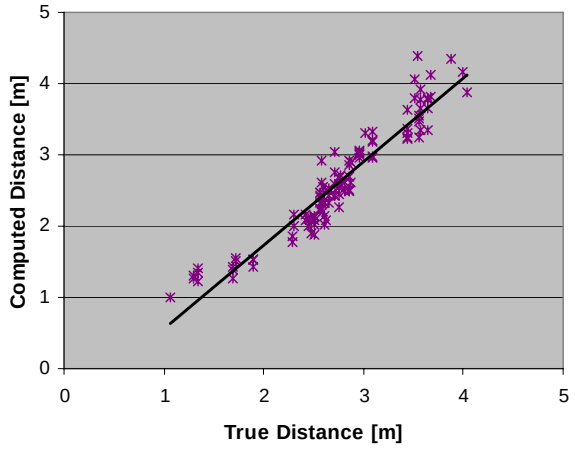

Figure 10 Actual versus computed tag-camera distances

CONCLUSION

Operational applications (such as control systems for heating, cooling, and lighting in buildings) require high-resolution self-updating spatial models with context awareness, if they are expected to implement advanced simulation-based control approaches. This necessitates identifying objects, sensing their location, and detecting occupancies. To explore how these requirements can be met, we developed VIOLAS. Thereby we selected network cameras as sensors, implemented assorted sensing algorithms, and the designed the system software architecture.

The system's sensing performance currently covers a 4 meters range and 75 degrees incidence angle, enabling the identification of all objects in the test-bed, and locating them with an accuracy of 0.18 meters average position and 4.2 degrees average orientation error. Thus, assuming a single netcam and pan-tilt unit, VIOLAS possesses an effective scanning area of approximately 50 m^2 (for both object location sensing and occupancy detection).

The initial results obtained from VIOLAS as well as recent experiments with cameras with fisheye lenses suggest that vision-based sensing, when enhanced computationally and integrated with appropriate hardware, may offer a promising technology for application domains such as facility management and indoor-environmental control applications in buildings.

ACKNOWLEDGEMENT

The research presented in this paper is supported by a grant from FWF (Austrian Science Foundation), project number L 219-N07.

REFERENCES

DCOM. 2004. Microsoft COM technologies. *http://www.microsoft.com/com/*

Forsyth D, Mundy JL, Zisserman A, Coelho C, Heller, A. & Rothwell, C. 1991. Invariant descriptors for 3-D object recognition and pose. IEEE Transactions on Pattern Analysis and Machine Intelligence. 13(10): 971-991.

İçoğlu O & Mahdavi A. 2005. A vision-based sensing system for sentient building models, 22nd CIB-W78 Conference - Information Technology in Construction. pp. 559 - 566.

Iris 2006. Iris software (Version 5.34) for equi-rectangular transformation (http://www.Astro surf.com/buil/us/iris/iris.htm).

Lòpez de Ipina D, Mendonca PS & Hopper A. 2002. Visual sensing and middleware support for sentient computing. Personal and Ubiquitous Computing. 6(3): 206-219.

Mahdavi A. 2001. Simulation-based control of building systems operation. Building and Environment. 36(6): 789-796. Vol. 1. pp. 3-18.

Mahdavi A. 2004. Self-organizing models for sentient buildings. In A.M. Malkawi & G. Augenbroe (eds), Advanced Building Simulation: 159-188. London: Spon Press.

Mahdavi A, Spasojević B. & Brunner K. 2005. Elements of a simulation-assisted daylight-responsive illumination systems control in buildings. BS 2005: 9th International IBPSA Conference, Montreal, Canada. pp. 693 - 699.

Toth D, Aach T. & Metzler V. 2000. Illumination-invariant change detection, 4th IEEE Southwest Symposium on Image Analysis and Interpretation.

Trucco E & Verri A. 1998. Introductory techniques for 3-D computer vision. New Jersey: Prentice Hall Inc.