Markov Random Field model is used to enhance the accuracy of the separation. We validated our method on outdoor and indoor video sequences captured by the surveillance system at the university campus, and we also tested it through well-known benchmark video shots.

We have developed several methods for registering cameras from arbitrary motions or from biometrics (Tamás Szirányi, László Havasi and Zoltán Szlávik). We demonstrate here an application of our new robust walk detection algorithm based on our symmetry approach, which can be used to extract biometric characteristics from video image-sequences. To obtain a useful descriptor of a walking person, we temporally track the symmetries of a person's legs. In a further processing stage, these patterns are filtered, then resampled and transformed to a subspace with a much smaller dimension of an 'eigenwalk space'. Our method is suitable for use in indoor or outdoor surveillance scenes. Image registration methods are presented which are applicable to multi-camera systems viewing human subjects in motion. Determining the leading leg of the walking subject is important and the presented method can identify this from two successive walk-steps (one walk cycle). Using this approach, we can detect sufficient numbers of corresponding points for the estimation of correspondence between two camera views. This is the case both in overlapping and in a special case of non-overlapping camera configurations.

Figure 2: Different parts of the day in sequences at the entrance of Pázmány Péter Catholic University with segmentation results. Above left: in the morning ('am'), right: at noon, below left: in the afternoon ('pm'), right: wet weather.

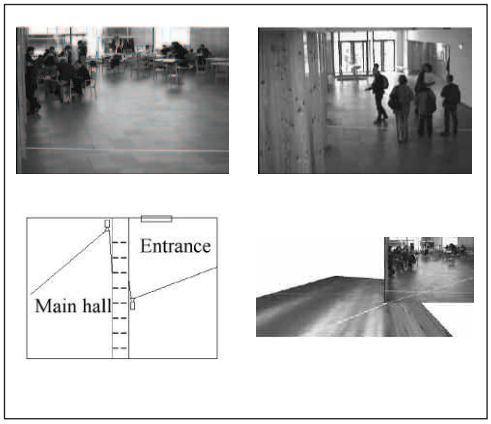

Figure 3: Top: images of the "main hall" and "entrance" (Pázmány Péter Catholic University) cameras with control lines on the ground (marked with two long paper tapes) for verification.
Bottom: on the left a schematic map of the experiment shows the placement of the cameras and their field of views. Right: result of alignment of non-overlapping views with the highlighted control lines.

Our project is partly supported by the Hungarian National Research and Development Program. With this activity, we contribute to the Network of Excellence project run by ERCIM: MUSCLE (Multimedia Understanding through Semantics, Computation and Learning).

Please contact:
Tamás Szirányi, SZTAKI, Hungary
Tel: +36 1 279 6106
E-mail: sziranyi@sztaki.hu
http://www.sztaki.hu/~sziranyi

Pervasive Computing

Applying a Model-Driven Method to the Development of a Pervasive Meeting Room

by Javier Muñoz, Estefania Serral, Carlos Cetina and Vicente Pelechano

Scientists at Universidad Politécnica de Valencia are implementing a pervasive system for managing a meeting room. The pervasive system has been developed using a model-driven method, and the final application integrates several technologies like EIB and Web Services. Three different user interfaces are provided for interacting with the system.

Researchers in pervasive systems have developed many software systems which try to achieve the Weiser vision. These systems have been implemented in a completely ad hoc fashion, or using implementation frameworks. Developing a pervasive system following these approaches is a hard and error-prone task. In order to improve the productivity and reduce the number of errors, we have developed a method that applies the Model-Driven Architecture (MDA) and the Software Factories approaches to the development of pervasive systems. The method is based on the specification of the system using PervML, a UML-like language designed for precisely describing the functionality of pervasive systems. The PervML specification is then automatically translated into Java code. The generated code extends an OSGi-based framework in order to build the final pervasive application.

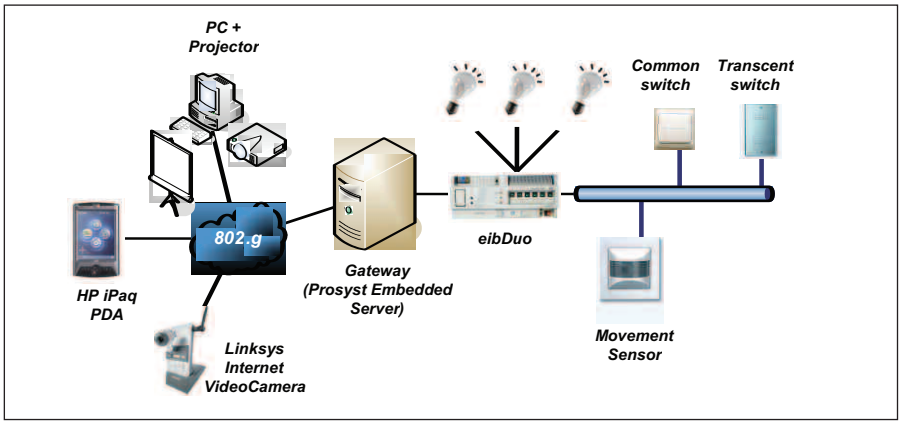

Figure 1: Network structure of the meetings room system.

The application studied here is that of the pervasive meeting room. Such a meeting room would contain features such as lighting management, multimedia reproduction, presence detection and video recording. For instance, when someone is near the projection screen, the intensity of the lightings may be automatically decreased in order to provide better visibility. Security systems may also record what occurs in the room.

In order to develop the system following our model-driven method, certain steps are followed:

- The system analyst specifies the system requirements using the service conceptual primitive. The system analyst uses three kinds of PervML models in order to describe (i) the kind of services available on the system, (ii) the number of services which are available in every location and (iii) how they interact when some condition holds.
- The system architect selects the kind and number of devices or software systems that are suitable for providing the services specified by the analyst. The system architect uses three other PervML models for describing (i) the kind of devices or software systems that are used for providing the system services, (ii) the specific elements that are going to implement each service and (iii) the actions that the device or software systems must carry out in providing each service operation.
- An OSGi developer implements the drivers for managing the devices or software systems. These drivers provide access from the OSGi-based framework to the devices or external software systems. They must be developed by hand, since they deal with technology-dependent issues.
- The transformation engine is applied to the PervML specification. Many Java files and other resources (Manifest files etc) are automatically generated as a result of this action.
- Finally, the generated files are compiled, packaged into bundles (JAR files) and deployed in the OSGi server with the implementation framework and the drivers.

The software that manages the pervasive system has been deployed in a Pentium IV barebone, which runs the Prosyst Embedded Server 5.2 as the OSGi implementation. In order to support the control devices (lights, switches and presence detector), an EIB network has been deployed.

The user interface is a key element in a pervasive system. Users interact with the system using several kinds of devices, so multiple user interfaces must be provided. Currently, we provide three different user interfaces:

- A Web interface for desktop browsers, which can be used by meeting attendees via their laptops or, for instance, by a supervisor in a central control unit.
- A native PDA application, which could be used from the PDAs of the company employees. The client application can be installed in their mobile devices since they interact frequently with the pervasive system, and the user experience is richer than using a Web application.
- A Web interface for PDA browsers, which can be used from the PDAs or other mobile devices of any room user. It is a Web interface with a resolution of 320x240 pixels.

Following the proposed method, the specification of the system functionality is independent of the devices selected for implementing the system. Moreover, the manufacturer-dependent details are isolated in the drivers' layer. With this approach, we can change all the implementation technologies just by replacing the drivers, but from the user's point of view the functionality provided by the pervasive system remains the same.

Link:
http://oomethod.dsic.upv.es

Please contact:
Vicente Pelechano,
Universidad Politécnica de Valencia, Spain
E-mail: pele@dsic.upv.es

Figure 2: The three user interfaces for managing the pervasive system.