A Multisensory Monitoring and Interpretation Framework Based on the Model–View–Controller Paradigm

José Carlos Castillo[1], Angel Rivas-Casado[2], Antonio Fernández-Caballero[1,3], María T. López[1,3], and Rafael Martínez-Tomás[2]

[1] Instituto de Investigación en Informática de Albacete (I3A), n&aIS Group, Campus Universitario s/n, 02071-Albacete, Spain
{JoseCarlos.Castillo,Antonio.Fdez,Maria.LBonal}@uclm.es
[2] Departamento de Inteligencia Artificial, E.T.S.I. Informática, Universidad Nacional de Educación a Distancia, 28040-Madrid, Spain
{arivas,rmtomas}@dia.uned.es
[3] Departamento de Sistemas Informáticos, Universidad de Castilla-La Mancha, Campus Universitario s/n, 02071-Albacete, Spain

Abstract. This paper proposes a monitoring and interpretation framework inspired in the Model–View–Controller (MVC) paradigm. Indeed, the paper proposes the extension of the traditional MVC paradigm to make it more flexible in incorporating the functionalities of a monitoring and interpretation system. The proposed model is defined as a hybrid distributed system where remote nodes perform lower level processing as well as data acquisition, while a central node is in charge of collecting the information and of its fusion. Firstly, the framework levels as well as their functionalities are described. Then, a fundamental part of the proposed framework, namely the common model, is introduced.

1 Introduction

Monitoring and interpretation systems have to understand and predict actions of the objects present in the scenario. To date there are many approaches developed for public transport monitoring, such as airports [14], ports [12], railway and subway stations [7,13], and traffic control [1]. Other important monitored public places are banks, shops, home or parking lots [15]. There are also systems indicated for human activity monitoring [11,4] or simply as an answer to industrial needs [5].

Moreover, the requirements of surveillance systems have led to systems able to monitor large areas. They go beyond simple object detection, including tracking and activity detection, and involving several sensors. Most works are centered in the combination of a set of cameras for this purpose. In [16] a system for classification and tracking of soccer players is proposed. The system consists of eight cameras (each one with its own processor) and a central processor for data collection and fusion. Another approach [6] describes the "EasyLiving" system.

The system performs multicamera tracking for behavior recognition in homes. Moreover, [18] introduces a client-server architecture for detection, recognition and tracking of people and vehicles. Finally, in [9] a distributed surveillance system is described.

The monitoring and interpretation framework proposed in this paper arises from the *Model–View–Controller* (MVC) paradigm [10]. This paradigm explicitly separates user inputs, world models and visual feedback, and proposes three kinds of blocks to handle them. In first place, the *model* is in charge of managing the application data as well as performing object initialization, providing information about the application state and primitives to update the state. On the other hand, the *view* provides a world representation adapted to the user's needs. And, finally, the *controller* collects all system inputs, invoking the model primitives to update its objects.

2 Extension of MVC for Monitoring and Interpretation Tasks

The MVC paradigm has been widely used for web programming where the model accesses the data (Database, XML), the view defines the user interface (HTML + CSS), and the controller reacts to the events by modifying the view and the model (PHP, .NET, JAVA). Despite the paradigm seems to be appropriate for the mentioned applications, its usage in monitoring and interpretation tasks entails a few improvements to the original paradigm. Therefore, the current work proposes the extension of the traditional MVC paradigm to make it more flexible in incorporating the functionalities of a monitoring and interpretation system. The proposed extensions also allow that already developed algorithms can be incorporated to the framework without great design changes. For that reason, business logic is separated from the model, generating a newer execution block managed by the controller. The new block is named *algorithm* and its operation corresponds to the algorithms of each framework level. Thus, the framework holds a series of extended MVC modules, each one devoted to a different level of the framework. Moreover, the model functionality in the proposed extension is the management of application data through a series of primitives (see Fig. 1).

The previously described modules are integrated into the traditional architecture through the incorporation of a new module. This module is the base where the remaining levels of the architecture are placed. In this special MVC module, the new model block composes the *common model.* This common model can be considered as the backbone of the architecture, since it holds the parameters needed or generated by the different levels of the architecture. Thereby, the dependencies among modules are removed and the data access is simplified by providing a unique module with well defined inputs and outputs. Later, in section 4, the common model features are defined, as well as the parameters at each architecture level. The view has been modified too in order to contain every interface of the different modules that compose the architecture levels. Thereby,

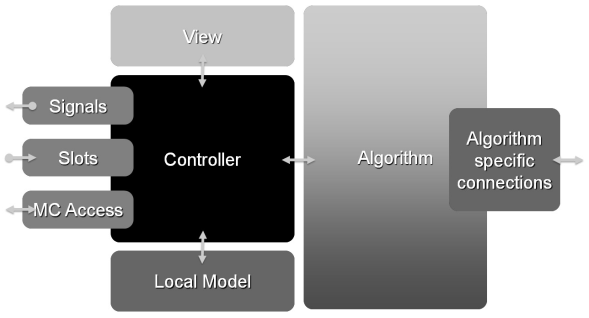

Fig. 1. Extension to the traditional MVC

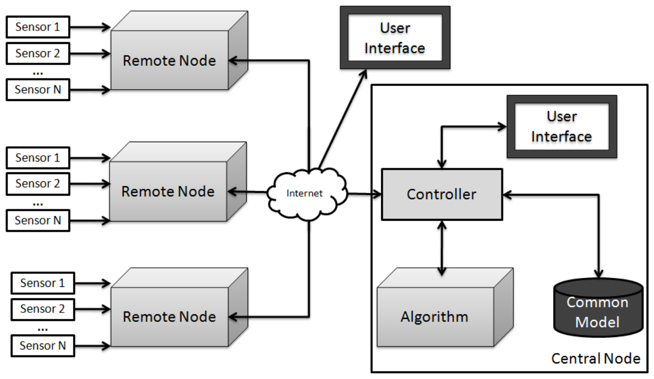

Fig. 2. Execution model. Hybrid system.

this module's view is defined as a multiple document interface (MDI) form where each view of the remaining modules (levels) appear as forms within the main view. In this module, the controller's function is to manage the execution of the remaining modules, monitoring their correct operation.

3 Definition of the Framework

Prior to detailing each system level, it is necessary to describe the execution model. It is defined as a hybrid distributed system where remote nodes perform lower level processing as well as data acquisition, while a central node is in charge of collecting the information and of its fusion. In Fig. 2 a schematic representation of the framework modules is shown. Remote modules have the MVC extended structure described in the previous section, but perform just a subset of all architecture layers described in the next section. Common model, a global control and view are also held in the central node, with MVC extended structure too.

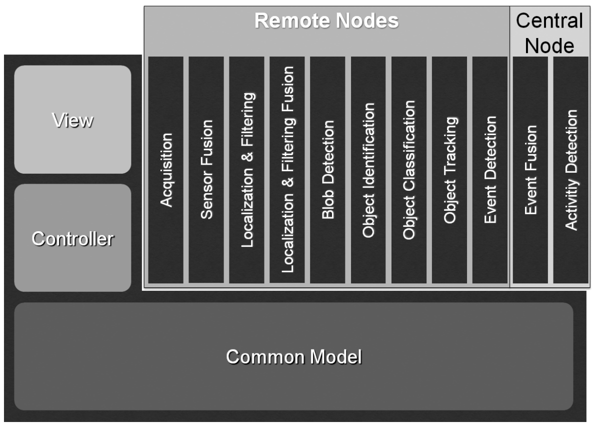

Fig. 3. Framework layers

3.1 Layers of the Framework

The current section describes the framework levels (see Fig. 3). In first place, the common model stands out. As aforesaid, this block is accessible from every level. The levels are composed by MVC extended modules (see Fig. 1). Next, the functionality of each level is described, although, in order to keep the framework as generic as possible, no algorithm is associated. Of course, the proposed levels are just a guideline to create the framework, but it is possible to include new levels according to the application requirements.

Acquisition: This level directly interacts with the digital analog devices, measuring from the physical world and adapting these measures to be usable by the system. The measures are data from the sensors as well as data from other information sources (disk, database, and so on). The acquisition level also performs information preprocessing.

Sensor Fusion: This level is in charge of merging the sensor data to improve the information quality (more complete and accurate). Fusion algorithms may also operate with different spectrum images and are capable of introducing knowledge on the domain.

Localization and Filtering: The third framework level is dedicated to isolate the objects of interest contained in the input images. This level may hold a wide range of methods, from the simplest one (a binarization applied to infrared (IR) images [2]) to other more complex approaches yielding better results [8]. On the other hand, this level also filters the spots corresponding to the objects of interest with the aim of eliminating possible noise.

Localization and Filtering Fusion: This level fuses images obtained in localization and filtering stage as there might be several localization and filtering approaches running in the framework (e.g. one devoted to color images and another to IR images). Thus, this level seeks for the most benefic features from the input images.

Blob Detection: The blob detection level filters isolated spots misdetected in the previous levels. Besides, the blob detection level is in charge of extracting information associated to the spots to allow a more efficient analysis of the objects. This information is application-dependent.

Object Identification: This level operates with objects instead of blobs. This enhances the information abstraction, mapping object coordinates into the real world instead of simply operating with image coordinates.

Object Classification: This level is specially important to perform a good activity analysis because it provides knowledge about "what" the object is. Also, object classification may provide information about the objects' orientation.

Object Tracking: This level is in charge of mapping the image objects' coordinates into the real map. Thus, it calculates the trajectories followed by the moving objects within the scenario, independently of the particular sensor that detected them. It also makes predictions about future positions of the objects on the basis of the previously detected trajectories. This level uses the information from the common model referring to the map, the sensors situation and its coverage range.

Event Detection: The event detection level generates semantic information related to the behavior of the objects in the scenario. These events are considered instantaneous, that is, they are time independent. Some examples are events such as running, walking or falling, which can be detected with just one or at most a few input images. This is the last layer of the framework held within remote nodes (see Fig. 2). The next layers are implemented in the central node together with the common model and the central controller and view.

Event Fusion: In a multisensory monitoring and interpretation system, where several sensors monitor a common scenario, the events generated from different sources usually do not match. This is why the event fusion level is necessary to unify the information arriving from the different sensory data generated in the previous level.

Activity Detection: This final level of the architecture is in charge of the analysis and detection of activities already associated to temporal features. After event fusion, the current level has a better knowledge of what is happening in the scenario according to the detected events. Hence, the activities detected at this level can be translated into actions along the scenario, providing a higher abstraction level.

4 Definition of the Common Model

After describing the framework levels, it is time to simplify the information exchange among them. For this reason, a new and fundamental layer is considered. This layer, known as the common model, gathers all the information from the different levels while providing primitives to access the information. The common model introduced is a variation of the traditional model of the MVC, where the algorithm of each module processes the information – always under the controller's management. Thereby, the common model is only in charge of holding the common information to be accesses by every execution module. For this purpose, the primitives that allow managing the data are provided. To properly define the common model, we will start with the layers that compose the architecture. Since the input and output parameters are known, it is possible to estimate which of them belong to the common model.

Acquisition: This level obtains data from diverse sources (camera, sensors, database, etc.) which determine the nature of the parameters contributing to the common model. In first place, the common model counts on a list of images captured from the cameras, regardless of the subjacent camera technologies (IR, color, range, etc.), adapted to the application requirements. A time parameter (ID_TIME) is associated to each image to ensure a correct synchronization. Moreover, sensor readings are also included in the common model as XML structures (again with an associated time parameter).

Sensor Fusion: As the most common parameters in a monitoring and interpretation system are images, this level contributes to the common model with a list of new fused images, independent from the acquisition image list. The content of this list depends on the fusion algorithms implemented . Again, time information is associated to fused images. Fusion of non-visual sensory data is open due to their high dependence from the sensor technology and the application.

Localization and Filtering: In literature, it is common to mix two terms when talking about segmentation [3,17]: Firstly, there is localization and filtering defined as the process whereby, from an input image, a set of spots containing the objects of interest are isolated. And, secondly, there is the blobs detection process. Clearly, this level receives images as input coming either from acquisition level or from sensor fusion level. Output parameters are a list of images containing isolated regions. These regions are highlighted from the background (white foreground on black background). Again, it is important to provide time information to the images. Thus, a time field (ID_TIME) is again attached to the images.

Localization and Filtering Fusion: This level fuses the information coming from the previous one. This is because several algorithms can be incorporated to the framework at localization and filtering level and its combination should provide a more accurate localization. This level incorporates a new fused image list, containing the results of its operation, to the common model.

Blob Detection: This level operates with blobs instead of sensor measures. The detection process uses information from the localized and filtered images to obtain the blobs contained. As some kinds of sensors provide distance information (e.g. range sensors), blob coordinates are defined with six components (x_{image},y_{image},z_{image},$width_{image}$,$height_{image}$,$depth_{image}$), even though depth components (z_{image},$depth_{image}$) might be void. During the blob detection process, other parameters defining the spots, such as the contour, brightness or color information may also be extracted. On the other hand, heuristic methods for detection can be applied to discard wrong spots. Again, time information and a unique identifier are associated to each blob.

Object Identification: At this level, object-level information is obtained from the blobs information. This level takes into account the history of the objects in the scene, that is, object features are updated along time by comparing blob information from current and previous iterations. Thanks to the knowledge about the history of an object, it is possible to calculate parameters derived from its motion (e.g. direction and speed). Also, information regarding corners or invariant points can be added to the object's definition, depending on the application needs.

Object Classification: The classification level uses contour information for clipping the images (acquisition or fusion level) as inputs to the classifier. This way, the classification algorithm provides information about what the object (its class) is. Classification methods can also obtain the object's orientation, which is useful for the activities detection. This level utilizes as input the acquired images as well as the information generated from the previous level (contours, invariant points, and so on).

Object Tracking: This level is in charge of calculating the trajectories followed by the moving objects in the scene. Previously it is necessary to calculate the objects' positions in the real world's map. On the other hand, a monitoring and interpretation system must keep tracking the objects, independently of which sensor is detecting it. For that reason, the calculated trajectories must be independent from the sensors (if possible).

Event Detection: This level translates the information coming from the lower levels into semantic primitives to model the relationships involving the objects present in the scenario. Thus, the inputs are the objects tracked by the previous levels and the sensor data whilst the output is closely linked to the event detection algorithm (HMM, SVM, Bayesian networks, etc.). Anyway, it is possible to define a common event representation format by using the flexibility of XML abstractions (see Table 1). XML provides an open structure, able to wrap all proposal outputs, homogenizing them for use by upper layers. This representation also allows managing the probability associated to the events. Again, a time parameter is associated to the events to simplify the event fusion process in the next level.

Table 1. Event structure

```
<events>
    <event name=Run>
        <object>1</object>
        <speed>5</speed>
        <probability>0.7</probability>
        <timeStamp>1256953732</timeStamp>
    </event>
    <event name=Walk>
        <object>2</object>
        <speed>1</speed>
        <probability>0.6</probability>
        <timeStamp>1256953735</timeStamp>
    </event>
</events>
```

Table 2. Main features of the common model

Level	Parameters	Details
Acquisition	- Acquired image list - Data from sensors ⇒ XML - IdTime	Data collection Information sources management Synchronization
Sensor Fusion	- Fused image list	Create new images from the acquired ones
Localization & Filtering	- Localization & Filtering image list	Isolate spots containing the objets of interest
Localization & Filtering Fusion	- Fused image list	Merge localized & filtered images
Blob Detection	Blob list containing: - IdTime - Contours - Box (X,Y,Z,W,H,D) - Color and brightness	From pixel-level information to blob-level Obtain parameters to characterize the spots (coordinates, color info, contours, etc.)
Object Identification	Object list containing: - IdTime - Box (X,Y,Z,W,H,D) - Real position - Label - Contours - Characteristic Points - Direction - Speed	Blob-level information to object level Information remain among iteration, being updated Calculation of parameters defining object motion Mapping of the image coordinates of the objects into the real world
Object Classification	- Class - Orientation	Update object list Provide information concerning "what" the objects are and their orientation
Object Tracking	- Trajectory	Calculation of object trajectories Match objects changing the sensors' detection field
Event Detection	- Event list	Instantaneous event detection (semantic primitives) Events are wrapped into XML structures
Event Fusion	- Fused event list	Performed by central node Update event list, discarding mismatching events and unifying repeated ones Synchronization is a key aspect (when to fuse) Feedback to local nodes according to their received events
Activity Detection	- Activity list	Global view of the scenario Spatial and temporal information
Scenario Modeling	Scenario map - Light conditions - Temperature - Sensors and their ranges - Detection distance	Related sensors information Global and local modeling

Event Fusion: This event fusion level allows the consideration of new information sources (the remote nodes), even though, at this level, sources provide events. Event fusion provides a general view of the scenario by merging all events, and eliminating wrong events. This again brings up the necessity for a good synchronization of the nodes. The operation of this level generates an event structure (as shown in Table 1) used as input by the next level.

Activity Detection: The higher level of the framework is devoted to the detection of activities from the events detected in the previous levels. As the events are generated by remote (distributed) nodes, their fusion provides a general view of the activities carried out in the scenario. This level provides the common model with high level activities detected that may involve several sensors in the scenario and several objects.

Scenario Modeling: Even though scenario modeling does not appear as a level within the framework definition, it is a key aspect that enables the relations among sensor information for situating the objects in the scenario. Scenario modeling can be seen as a two stage modeling. On the one hand, there is a global modeling that includes aspects such as map definition, map calibration, environmental condition setting (ambient light or temperature), and sensor placement and its range. On the other hand, there exists a modeling that is sensor-specific like the camera field of view. Some parameters are dynamically updated according to the sensed values. Table 2 summarizes the main features of the architecture levels, as well as the parameters provided to the common model.

5 Conclusions

This paper has introduced a monitoring and interpretation framework inspired in the MVC paradigm. The paper has proposed the extension of the traditional MVC paradigm for the inclusion of the functionalities of a monitoring and interpretation system. The proposed model is defined as a hybrid distributed system where remote nodes perform lower level processing as well as data acquisition. A a central node is in charge of collecting the information and of its fusion. The remote nodes hold the following layers of the framework: acquisition, sensor fusion, localization and filtering, localization and filtering fusion, blob detection, object identification, object classification, object tracking, event detection, event fusion. The central nodes incorporates activity detection and scenario modeling.

Acknowledgements

This work was partially supported by Spanish Ministerio de Ciencia e Innovación TIN2010-20845-C03-01 and TIN2010-20845-C03-02 grants, and by Junta de Comunidades de Castilla-La Mancha PII2I09-0069-0994 and PEII09-0054-9581 grants.

References

1. Celik, T., Kusetogullari, H.: Solar-powered automated road surveillance system for speed violation detection. IEEE Transactions on Industrial Electronics 57(9), 3216–3227 (2010)
2. Fernández-Caballero, A., Castillo, J.C., Serrano-Cuerda, J., Maldonado-Bascón, S.: Real-time human segmentation in infrared videos. Expert Systems with Applications 38(3), 2577–2584 (2011)

3. Fernández-Caballero, A., Castillo, J.C., Martínez-Cantos, J., Martínez-Tomás, R.: Optical flow or image subtraction in human detection from infrared camera on mobile robot. Robotics and Autonomous Systems 58(12), 1273–1280 (2010)
4. Gascueña, J.M., Fernández-Caballero, A.: Agent-oriented modeling and development of a person-following mobile robot. Expert Systems with Applications 38(4), 4280–4290 (2011)
5. Geradts, Z., Bijhold, J.: Forensic video investigation. In: Multimedia Video Based Surveillance Systems, ch. 1, pp. 3–12 (2000)
6. Krumm, J., Harris, S., Meyers, B., Brumitt, B., Hale, M., Shafer, S.: Multi-camera multi-person tracking for easyliving. In: Proceedings of the Third IEEE International Workshop on Visual Surveillance, pp. 3–10 (2000)
7. Lo, B.P.L., Sun, J., Velastin, S.A.: Fusing visual and audio information in a distributed intelligent surveillance system for public transport systems. Acta Automatica Sinica 29(3), 393–407 (2003)
8. Moreno-Garcia, J., Rodriguez-Benitez, L., Fernández-Caballero, A., López, M.T.: Video sequence motion tracking by fuzzification techniques. Applied Soft Computing 10(1), 318–331 (2010)
9. Nguyen, N.T., Venkatesh, S., West, G., Bui, H.H.: Multiple camera coordination in a surveillance system. ACTA Automatica Sinica 29(3), 408–422 (2003)
10. Reenskaug, T.: Thing-model-view-editor - an example from a planning system (1979), `http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html`
11. Rivas-Casado, A., Martínez-Tomás, R., Fernández-Caballero, A.: Multiagent system for knowledge-based event recognition and composition. Expert Systems: The Journal of Knowledge Engineering (2011) (in press)
12. Rodriguez, M.D., Shah, M.: Visual surveillance in maritime port facilities. In: Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 6978, pp. 697811–697818 (2008)
13. Ronetti, N., Dambra, C.: Railway station surveillance: the Italian case. In: Multimedia Video Based Surveillance Systems, ch. 1, pp. 13–20 (2000)
14. White, I.H., Crisp, M.J., Penty, R.V.: A photonics based intelligent airport surveillance and tracking system. In: Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 11–16 (2010)
15. Wong, W.K., Lim, H.L., Loo, C.K., Lim, W.S.: Home alone faint detection surveillance system using thermal camera. In: Second International Conference on Computer Research and Development, pp. 747–751 (2010)
16. Xu, M., Orwell, J., Lowey, L., Thirde, D.: Architecture and algorithms for tracking football players with multiple cameras. IEE Proceedings - Vision, Image and Signal Processing 152(2), 232–241 (2005)
17. Yin, Z., Collins, R.: Moving object localization in thermal imagery by forward-backward motion history images. In: Augmented Vision Perception in Infrared, pp. 271–290 (2009)
18. Yuan, X., Sun, Z., Varol, Y., Bebis, G.: A distributed visual surveillance system. In: Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance, pp. 199–204 (2003)