International Journal of Systems Science (2014). 45(4), 741-755.
DOI: http://dx.doi.org/10.1080/00207721.2013.795632

A practical approach for active camera coordination based on a fusion-driven multi-agent system

Alvaro Luis Bustamante*, José M. Molina and Miguel A. Patricio

Applied Artificial Intelligence Group, Universid Carlos III de Madrid, Avda. Universidad Carlos III, 22. 28270 Colmenarejo. Madrid, Spain. Madrid 28270, Spain

*Corresponding author. Email: aluis@inf.uc3m.es

Abstract: In this paper, we propose a multi-agent system architecture to manage spatially distributed active (or pan-tilt-zoom) cameras. Traditional video surveillance algorithms are of no use for active cameras, and we have to look at different approaches. Such multi-sensor surveillance systems have to be designed to solve two related problems: data fusion and coordinated sensor-task management. Generally, architectures proposed for the coordinated operation of multiple cameras are based on the centralisation of management decisions at the fusion centre. However, the existence of intelligent sensors capable of decision making brings with it the possibility of conceiving alternative decentralised architectures. This problem is approached by means of a MAS, integrating data fusion as an integral part of the architecture for distributed coordination purposes. This paper presents the MAS architecture and system agents.

Keywords: active cameras; PTZ; MAS; multi-agent systems

1. Introduction

Nowadays, surveillance systems are evolving towards complex information systems that are able to provide a great deal of data about the environment gathered through spatially distributed sensor networks. The advances of the underlying technologies, like digital communication, video transmission and, specially, wireless sensors and actuator networks (WSANs) (Akyildiz and Kasimoglu 2004), have facilitated the deployment of new surveillance sensors for use in environmental monitoring, healthcare systems, homeland security, public safety and, generally, critical environments.

The video camera is perhaps the most important sensor in surveillance scenarios, as it has attracted much of the attention of the scientific community. Many visual sensor networks are used to provide rich features like object detection and tracking, object and colour classification, activity recognition, alert definition and detection, database event indexing and so on (Hampapur 2008). All these technologies tend to be integrated in a common system to provide richer and improved video surveillance experiences (Han and Bhanu 2007).

A real visual sensor network relies not only on fixed colour cameras, but can also be composed of a wide variety of heterogeneous sensors, like active, thermal and infrared cameras, or also time-of-flight cameras (Bevilacqua, Di Stefano, and Azzari 2006).

This paper focuses primarily on the problem of controlling multiple active (or pan-tilt-zoom, PTZ) cameras. As sensors, active cameras do not work like other camera types, which always monitor the same region, and operators have to handle the movements of each video camera in the environment manually according their perceptions or desires. Operators using an active camera to monitor a scene will be able to track situations of interest more flexibly.

In an environment where there are a lot of active cameras, some with overlapped fields of view (FOV), it makes sense to have cameras controlled somehow autonomously or even give different cameras the opportunity to collaborate autonomously to satisfy or improve some established goal. In this way, operators controlling such complex environments will be less encumbered, and the surveillance task will be improved as their attention will only be required when necessary. Cameras sharing a FOV can benefit from active cooperative sensing, which has many advantages over non-cooperative sensing, as reported in the literature (Collins et al. 2011).

Two main problems have to be solved in order to design this kind of visual surveillance systems, as described in Manyika and Durrant-Whyte (1994). The first issue is data fusion, which is related to the optimal combination of data from different sources (Liggins, Hall, and Llinas 2008). The second question is multi-sensor management, which assumes that the data-fusion problem has been solved, and should optimise the global management of the whole system by applying individual operations on each sensor (Molina López, García Herrero, Jiménez Rodríguez, and Casar Corredera 2003). There are two main approaches for solving this problem: the centralised and decentralised options. A centralised architecture is usually based on a data-fusion centre, or a network node, combining all

system information to plan and execute actions on each sensor. Thus, a simple prototype centralised architecture is affordable to build, but it has many drawbacks related to scalability, fault tolerance and deployment when the sensors are highly distributed (Molina López et al. 2003). The existence of intelligent sensors that are capable of making distributed decisions brings with it the possibility of conceiving alternative decentralised architectures.

So, the main concern of this paper is to describe an architecture for a distributed solution to the two main problems described above: sensor management, and data fusion for active cameras. Assuming that sensors may be highly autonomous (as the cameras are usually distributed across large surveillance environments), the local management system rather than a centralised node can make the decisions about the task that the sensor is to perform (Wesson, Hayes-Roth, Burge, Stasz, and Sunshine 1981). Thus, a multi-agent system (MAS) would be a suitable approach for this problem.

MAS research focuses on systems in which many intelligent agents interact with each other to achieve different goals. The research community has put agent technology into widespread use in intelligent, multisensory and distributed surveillance, as there are many works on the use of agents in activity detection, image segmentation, tracking, robotic application modelling, facial identification, multisensory data fusion and so on (Gascueña and Fernández-Caballero 2011). In this paper, however, we propose a high-level definition of a MAS architecture mainly for supporting autonomous active camera control. In order to achieve cooperative sensing, we have introduced a data-fusion level as an integral part of the MAS architecture. In this way, different agents will be able to collaborate based on information-fusion events. The architecture is designed in such a way that fusion nodes can be distributed depending on the cameras and their FOV, and there is no single node controlling the camera action. As it would be far beyond the scope of this paper to fully define such an architecture and its underlying algorithms, we have tried to provide a practical overview of the different parts of this system and how they will be developed and tested.

The rest of this paper is organised as follows. Section 2 discusses some related work dealing with multi-camera sensor management, algorithms used for controlling active cameras and some basic knowledge that must be taken into account to understand the architecture. Section 3 describes the general MAS architecture. The different parts of the architecture, like sensor management, data fusion and user interface with their underlying agents are detailed in Sections 4, 5 and 6. Section 7 describes how agent knowledge can be represented using an ontology. Finally, Section 8 provides some experiment results and discusses the possibility of using virtual environments to test this MAS.

2. Related work

Many interdisciplinary approaches are required to rise to the challenges of managing multiple active cameras in order to perform automatic control and coordination, where a set of active cameras track an object or a face, or zoom in to acquire high-resolution images of a region of interest. They range from utilising vision processing, through communication and networking, to artificial intelligence. In this section, we give a brief overview of the different approaches applied in the research community to solve the different parts of the problem.

The most basic issue is related to object detection and tracking with active cameras. As opposed to passive cameras, which always provide the same FOV, where a simple temporal difference algorithm is enough to detect the objects in the scene, a PTZ camera can easily change its FOV. So, applying traditional algorithms for active cameras will fail, providing motion detection for all pixels. Therefore, the same methods as for passive cameras cannot be used with PTZ cameras if tracking is to be robust against camera movements.

Some researchers have dealt with this issue. A proposal by Shibata, Yasuda, and Ito (2008) reported a novel technique based on optical flow distortion used to detect moving objects in the scene, even when the camera is changing its FOV. A similar approach presented by Lim, Elgammal, and Davis (2003) detects objects using an adaptive background subtraction algorithm. The mean shift procedure (Meer 2003) was adopted in Everts, Sebe, and Jones (2007) for successfully tracking objects from PTZ cameras. Other researchers give examples of face rather than object tracking with active cameras (Comaniciu and Ramesh 2000; Petrov, Boumbarov, and Muratovski 2008).

Once object tracking in the environment has been enabled, the camera pixels, or the object bounding box, need to be mapped to real-world locations. This is normally done by calibration. A vast amount of literature is available on this subject, and there are even automatic calibration algorithms (Heikkila 2000; Zhang 2000; Jones, Renno, and Remagnino 2002). However, traditional calibration techniques deal with stationary, passive cameras. In this context, active cameras have to be calibrated for any pan, tilt, zoom settings. Some very recent approaches that propose this type of calibration are described in Galego, Bernardino, and Gaspar (2012), Possegger et al. (2012) and Puwein, Ziegler, Ballan, and Pollefeys (2012). So the calibration process for active cameras and passive cameras is slightly different. Nonetheless, it is affordable if suitable specialised methods are used.

Now, assuming that we have solved the problem of tracking objects and mapping their positions to real-world coordinates, we have need of an upper control layer to handle the camera coordination when necessary. There are some interesting dual-camera frameworks for this purpose, in which a master camera takes wide panoramic images and

the slave zooms into the targets to get more accurate images, as described in Marchesotti, Messina, Marcenaro, and Regazzoni (2003) and Collins et al. (2011). A more complex approach than the master-slave method is dynamic target assignment according to camera availability and accuracy (Singh, Atrey, and Kankanhalli 2007). Moreover, Krahnstoever, Yu, Lim, Patwardhan, and Tu (2008) addresses the generic problem of collaboratively controlling a limited number of PTZ cameras in order to optimally capture an observed number of subjects. Previous research by our group (GIAA), reported in Garcia, Carbo, and Molina (2005), Castanedo, Patricio, García, and Molina (2006) and Patricio, Carbó, Pérez, García, and Molina (2007), deals with the problem of still/active camera management with MAS. However, it does not contemplate data fusion as an integral part of the MAS architecture and management. Investigation related to MAS and data fusion was presented in Castanedo, García, Patricio, and Molina (2010), where a MAS system was used along with a data-fusion system to improve multiple camera tracking. It does not, however, contemplate the use of active cameras, and fusion is only used to improve tracking, not as a source of coordination.

A prototype for cooperative object tracking between two PTZ cameras was built in Everts et al. (2007). In this case, the authors described the need to share common knowledge about target location and appearance; therefore, the cameras had to be able to communicate somehow. An alternative for a proactive PTZ camera control was described in Qureshi and Terzopoulos (2011). They employ a centralised planning strategy to select optimal camera assignment and handoff with respect to predefined observational goals. However, they encountered a scalability problem when dealing with numerous active cameras spread across an extensive region, so they are planning to tackle the scalability problem by investigating distributed multi-agent planning strategies in the future.

So, whereas the approaches for the object tracking and active camera calibration problem appear to be sound, we found that attempts at managing the upper layer controlling the active camera are likely to fail on a common point: camera coordination in a single node. The use of MAS in surveillance environments as a method to achieve better scalability and robustness is not new to the research community. There are some good approaches dealing with a heterogeneous net of sensors like the research reported by Pavón, Gómez-Sanz, Fernández-Caballero, and Valencia-Jiménez (2007). In this research, a whole surveillance system is designed as a MAS, identifying agents, how they are grouped and the roles they can play in the system. This is sound groundwork that highlights the design issues that have to be addressed when applying MAS to this kind of problems. It is also a good example of how to design MAS using the INGENIAS methodology.

3. Multi-agent system architecture

Consider an active camera network of N_a cameras, some of which have overlapped FOV, described by the tuple $\{\theta, \alpha_{min}, \alpha_{max}, \beta_{min}, \beta_{max}\}$, where we assume that the three-dimensional position θ of each active camera is known *a priori*, and where $[\alpha_{min}, \alpha_{max}]$ and $[\beta_{min}, \beta_{max}]$ represent pan and tilt limits respectively for each active camera. Suppose, moreover, that each active camera knows how the gaze direction parameters (α, β) map to the three-dimensional world locations. Thus, given the three-dimensional location of a target, the active camera is able to direct its gaze towards the target. Some of the algorithms that fit these assumptions were discussed in Section 2, so this paper will focus on the high-level control layer. So, basically, consider that we have a set of cameras N_a with the required configuration and algorithms to perform single actions like search, fix, zoom and track targets.

With the above assumptions, we propose a distributed camera control architecture based on a MAS driven mainly by fusion events. This architecture will allow each agent controlling an active camera to collaborate with others, whether or not they share part of its FOV. For this purpose, the architecture has been divided into three different logical tiers. The first tier, called *Sensor Management*, is related to the agents that will control the physical sensor itself. This low-level tier, mainly composed of agents, will support the other two tiers: *Data Fusion* and *User Interface*. The *Data Fusion* tier is related to distributed fusion that will enable agent coordination. The *User Interface* will enable a potential operator to monitor and control the state of the environment.

As the architecture overview illustrated in Figure 1 shows, the *Sensor Management* tier includes the *Sensor Agent* (S_a), *Control Agent* (C_a) and the *Manager Agent* (M_a), which together provide the active camera control and management. There is only one M_a in the environment that will control agent deployment, whereas there are normally a pair of S_a and C_a for each camera. The *Local Fusion Agent* (LF_a) and the *Global Tracking Agent* (GT_a) come into play in the *Data Fusion* layer. The *Local Fusion Agent* (LF_a) provides local coordination between C_a sharing FOVs, and the *Global Tracking Agent* (GT_a) provides the user interface with a set of non-redundant tracks of the different targets in the environment without loss of information about the underlying tracks. Also, it can provide a coordination mechanism between C_a, as it can notify interesting tracking events for some agents. Finally, the *User Interface* subsystem will be composed of a variety of agents depending on user requirements. Some agents that may be useful *a priori* are, for example, the *Event Agent* (E_a), which can recollect events that occur in the environment, like a pedestrian entering a restricted zone (saving its location, pictures, etc.), a *Recorder Agent* (R_a), which receives and records the video streams from the different S_a, and an *Interface Agent* (I_a),

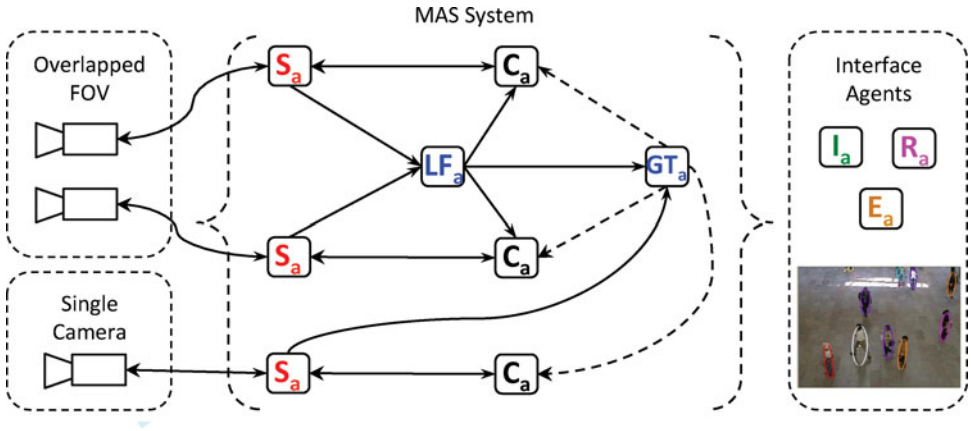

Figure 1. MAS architecture solving sensor management and sensor data fusion.

which will provide a user interface for users to establish new requirements, watch the different video streams, view the system track list, track features, and so on.

As discussed in Everts et al. (2007), the cameras, or in this case the different agents, must share some knowledge for coordination purposes. In this way, some kind of intermediate framework is required that is able to straightforwardly deploy new agents, enabling agent communication and reasoning. Perhaps the best known framework is JADE (Bellifemine, Poggi, and Rimassa 2001), which implements an important standard enabling heterogeneous interactions among agents. The communication standard was developed by the Foundation for Intelligent Physical Agents (FIPA) and is known as the Agent Communication Language (FIPA-ACL). Frameworks like JADE simplify the implementation of MAS using middleware that complies with FIPA specifications and a set of graphical tools that support the debugging and deployment phases. In this platform, agents can be distributed across machines (which do not even need to share the same OS), and the configuration can be controlled via a remote GUI. It would therefore be handy to design our MAS based on a framework like this, as the development, deployment and testing phases will go much faster. We opted for the JADEX framework (Braubach, Lamersdorf, and Pokahr 2003), as it extends JADE, adding the possibility of running agents based on the belief-desire-intention (BDI) model. The BDI model is now possibly the best-known and best-suited model of agent practical reasoning (Georgeff, Pell, Pollack, Tambe, and Wooldridge 1999). Reasoning agents are potentially proactive, tending to initiate changes rather than only reacting to events.

Bearing these ideas in mind, we describe the different tiers of the multi-agent architecture in more detail in the following sections. We also discuss some low-level details for this real-time problem or even deal with the communication specification for some agents.

4. Sensor management

The *Sensor Management* layer is the part of the architecture that allows the MAS to both communicate and control cameras. It acts somewhat like a low-level interface between the upper-level control procedures and the sensors themselves. It is mainly composed of three different agents, the *Sensor Agent*, *Control Agent* and *Manager Agent*. The purpose of each agent is discussed in the following sections.

4.1. Sensor Agent

This agent is related to the sensor itself and basically provides a sort of interface, rendered as capabilities, enabling other agents and systems to perform actions on the camera or get data from the sensor. This agent mainly wraps a software component called *Sensor Controller*, which is middleware between the camera and the agent. This controller physically interacts with the camera, and, as described in Bustamante and Molina (2011), currently provides three high-level interfaces, independently of the active camera model, which are decomposed into video, control and tracking interfaces.

The *Video Interface* enables the real-time transmission of video sequences across the network using the JPEG-2000 codec. This is an important part of the architecture for meeting the real-time requirements of the video surveillance scenario. For example, considering the transmission of video sequences directly using the JADEX framework via the FIPA-ACL communication language would lead to an immediate overhead, and excessive message processing for the framework, which will ultimately result in a big delivery delay in the video visualisation. So, this is not an affordable way of streaming video sequences. As defined in Bustamante, Molina López, and Patricio (2011), this *Sensor Controller* achieves the transmission with a state-of-the-art real-time transport protocol (RTP), which encodes

Table 1. Tracking events raised by the S_a.

Event name	Description
Track creation	Sent when a new track has been detected in the camera FOV, and cannot be correlated with any previously existing track.
Track deletion	Sent when an existing track is no longer in the camera FOV, i.e. a track has not been updated for a set time period.
Track update	Sent when an existing track has been updated, i.e. its position or any other feature has changed. Track update will be always sent after track creation, and before track deletion.

and compresses video frames using the JPEG-2000 codec. This protocol also allows multicasting, so many network destinations can receive the same video source in real time without increasing the delivery delay.

The *Control Interface* is used to control the different parameters offered by the camera. The typical parameters for active cameras would be pan, tilt and zoom. It allows external entities, like other agents or operators, to change the camera FOV to satisfy their own goals.

Finally, the *Tracking Interface* is probably the most important part of the *Sensor Controller*, as it is concerned with offering tracking capabilities for the upper control layers. Tracking features refer to the capability of offering tracking information, like number of tracks, shape, size, position, identifier, colour density and any other track features that may be available depending on sensor type, implemented tracking algorithms and the enacted feature extraction process. In this way, this interface may execute several tracking modes working at the same time depending on the features that other agents want to extract, like blob, colour, thermal tracking or even face recognition. The tracking information generated by S_a and other agents is communicated via tracking events (transmitted under the JADEX messaging framework). The tracking events have been defined in Table 1, and basically consist of track creation, update and deletion, which represent the different track statuses. The metadata generated by the tracking algorithm, i.e. track identifier, track location, track shape, colour density and so on, associated with the event itself is also attached. This will be useful for later fusion processes.

Figure 2 illustrates this idea. In this case, the *Sensor Agent* does not interact directly with the physical camera; instead the *Sensor Controller* provides a common interface for different cameras, and the *Sensor Agent* does not have to be adapted to every network camera. Notice that the *Sensor Controller* does all the camera processing, specially the tracking process. In this way, an extra video transmission process is not necessary prior to image processing. The tracking result and the video sequences can be delivered separately depending on agent requirements. On the one hand, a given agent, like *Recorder Agent*, may only need the video stream for recording purposes, whereas another, i.e. *Control Agent*, may require only the tracking information for camera management purposes. On the other hand, an *Event Agent* may require both streams: the tracking information to detect events (like intrusion detection in some restricted zone), and the video stream to take the event pictures. Moreover, with just one image processor residing in the *Sensor Controller*, the tracking information can be delivered to multiple agents at the same time. This approach reduces the workload of remote agents, as the video sequences do not have to be transmitted to all remote agents, and the agents do not then have to process the images for every incoming frame. If an agent were to require any special image processing to execute a special algorithm, it can always retrieve the video stream and perform its own analysis.

The *Sensor Agent* in conjunction with the *Sensor Controller* should either be run on the computer to which the camera is attached, or could, ideally, be embedded in a smart camera with onboard computing and networking features. This will provide for an easier integration of the different parts: video acquisition, transmission, processing, running the agent and so on.

4.2. *Control Agent*

The *Control Agent* (C_a) is responsible for controlling camera movements according to the established goals and the incoming environment events. The C_a will expect two different types of incoming events (see Figure 1). First, the tracking events from the S_a, which, at their simplest, may

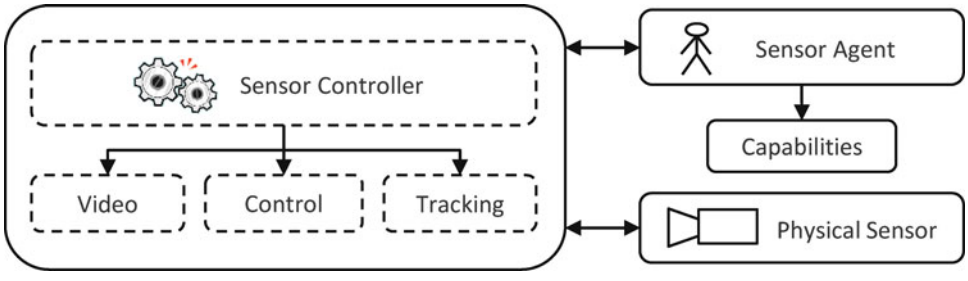

Figure 2. *Sensor Controller* middleware used by the *Sensor Agent*.

provide the camera with some kind of autonomous control, like automatically changing camera gaze according to a predefined or random path in search of new tracks, following an operator-specified track, switching between different monitoring areas, maximising the number of tracks in the camera FOV, automatically starting to follow tracks with some predefined features and so on. All these actions can be achieved by listening to the tracking events provided by S_a, and each event may trigger different agent reactions. For example, a track creation event alerts the C_a to the presence of new tracks, which may provoke an agent reaction, like starting more detailed monitoring; a track update event will allow the agent to continue tracking at the latest known track location; or, finally, a track deletion event can cause the C_a to start monitoring different zones in search of new tracks.

The benefits of relying on the S_a is that this agent does not have to perform any image processing to achieve its purposes. Instead, the tracking information is received directly and can be used to control the active camera. Figure 3 illustrates an example of this mode of operation, where S_a outputs different information flows, like the video streams to be displayed in the interface and the tracking information that is used to both control the camera and complement the visual information in the interface. The C_a merely receives and processes the tracking information and then controls the S_a.

There can be no question that the C_a is very useful working in autonomous mode. However, as stated in the Introduction, the use of active cooperative sensing would improve or will, at least, expand system capabilities. Hence, apart from running in a stand-alone way, this agent is also assumed to be part of the process of collaboration among different cameras. So, the fusion events generated by LF_a are the second source of environment events, which is useful for establishing collaboration among different C_a. This mechanism will be explained in more detail in the *Data Fusion* tier. Basically, though, any $C_a i$ will be able to collaborate with any other $C_a j$ to perform a common task, like following the same track, taking different regions or details from the track, looking for a lost track, trying to cover different tracks and so on.

4.3. Manager Agent

The *Manager Agent* (M_a) is responsible for simplifying MAS initialisation to expedite MAS architecture startup every time the surveillance system is initialised. This agent relies on a *Sensor Repository* that contains information about all the deployed sensors, like sensor type, sensor ID, real-world sensor location, calibration parameters, etc. It also contains the relationship among the different cameras, such as cameras sharing some FOV. This information should be initialised manually while setting up the architecture, and it can be reused in other system executions.

The initial functioning of the *Manager Agent* can be divided into two steps. The first is related to loading the sensor repository for the dynamic creation/initialisation of S_a for every camera present in the environment. Each S_a will then be associated with a C_a that will handle the camera controls, as discussed in Section 4.2 and presented in the system architecture (see Figure 1). The second relies on the creation of static coalitions through LF_a for S_a that share some FOV. In this way, a LF_a is created for each set of S_a with some overlapped FOV, for tracks from the respective cameras. These LF_a are also related to the associated C_a, so the C_a can benefit from the fusion events sent by the LF_a. The last agent in the architecture, that is, the GT_a, is created after all the S_a, C_a and LF_a have been deployed.

5. Data fusion

Data fusion is normally related to the process of integrating multiple data and knowledge representing the same real-world objects into a consistent, accurate and useful representation. Data fusion is mainly used in the geospatial (GIS) domain, where there is a need to combine diverse data sets into a unified (fused) data set that includes all the data points and the time steps from the input data sets. The fused data set is different from a simple combined superset in that the points in the fused data set contain attributes and metadata that the original data set might not have included for these points. Therefore, integrating data from multiple sensors of different types provides a better result because the strengths of one type may compensate for the weaknesses of another type (Hall and Llinas 2001).

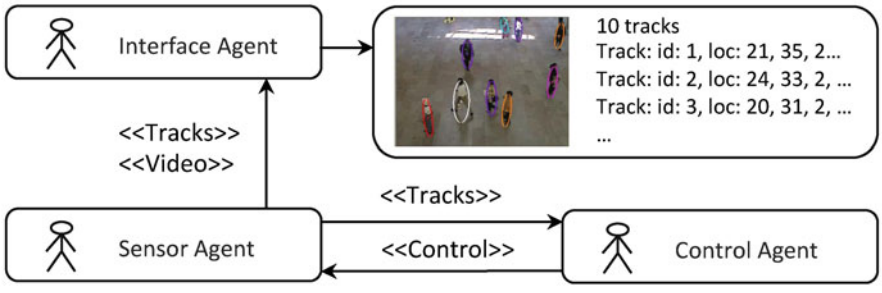

Figure 3. Using the *Control Agent* to control an active camera.

The *Data Fusion* tier of this MAS architecture is not so related to trajectory improvement as in research reported by Castanedo et al. (2010). Instead, this architecture uses data fusion mainly for coordination purposes. In an environment where there are a lot of cameras (some with overlapped FOV) and different targets need to be tracked, the agents have to know what the nearby cameras are watching, i.e. whether two cameras are looking for the same track. Agents can use this knowledge, which is *a priori* unknown, to proactively collaborate to achieve a common goal with respect to the track, like taking pictures of different regions of interest. Here, data fusion is a way of somehow providing dynamic coordination of the different C_a, communicating the information on shared observed tracks among cameras. This is done by raising fusion action events between the C_a sharing some FOV. Another reason for using data fusion is to provide centralised knowledge of the environment for the operator. As the MAS is running in a distributed manner, there has to be a way of displaying to the operator a real-time summary of the environment, like the number of tracks being monitored, their positions, their shapes, etc., omitting, if possible, redundant data. The interface part will be discussed in the *User Interface* section.

The *Data Fusion* tier is composed of two different agents, the *Local Fusion Agent* (LF_a) and the *Global Tracking Agent* (GT_a), which are explained in the following sections.

5.1. Local Fusion Agent

The LF_a is deployed at system startup and communicates with the S_a and the C_a, as shown in Figure 4. There is one instance of LF_a for each set of cameras sharing some FOV (as the same real-world objects viewed by different cameras should be fused). This agent may be physically distributed in the network, i.e. by proximity to its S_a, as there is no restriction on all LF_a instances having to run on the same computer. The sensor-related agents (S_a) provide the result of the local tracking process, which is a set of tracking events, as the source input for LF_a. Then, the LF_a processes the incoming track information to discover possibly

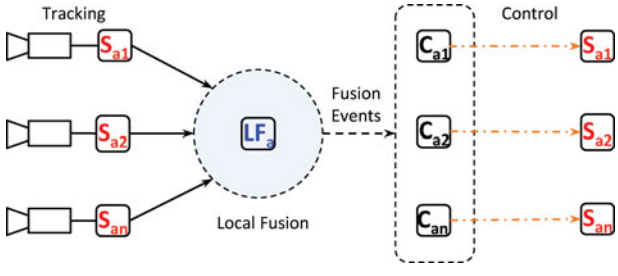

Figure 4. The LF_a agent will receive the tracking events from the cameras sharing some FOV (from S_a agents) in order to provide fusion events to the C_a agents. C_a agents may use this information to start collaboration.

Table 2. Fusion events raised by the LF_a.

Event name	Description
Fusion creation	Sent when at least one pair of source tracks from different source cameras have been fused for the first time as their characteristics (location, colour, etc.) are similar.
Fusion deletion	Sent when an existing fused track has been broken. This may happen when a source track within the fused track has been removed by the S_a, and the fused track has no meaning.
Fusion update	Sent when a fused track has been updated. This may happen when an underlying source track has been updated, or when a source track has been added or removed from the fused track. Like the track update event, this event is only sent after fusion creation, and before fusion deletion.

redundant tracks generated by overlapped FOV for fusion to provide a single representation of each track, which is the LF_a output. The proposed LF_a will be able to perform data fusion using the tracking information provided by each S_a, like track size, real-world track location (output by calibration processes), etc. A good approach for this purpose is described in Snidaro, Foresti, Niu, and Varshney (2004). Remember that this agent does not have to perform any image processing because all the necessary information is received as tracking events processed locally in S_a.

Like the S_a, which provides tracking events, the LF_a will provide fusion events with the same statuses (creation, deletion and update). Table 2 lists the fusion events along with a short description. These output events are used as input for the C_a, as shown in Figure 4. This allows collaboration or coordination among the different C_a working with a shared FOV to perform different actions. The use of fusion events in C_a agents is similar to the autonomous reaction to tracking events. In this case, however, a fusion event implies some shared knowledge about tracks, which can be used to initiate dynamic coalitions among agents for the purpose of coordination.

The purpose of a dynamic coalition is to enable the respective C_a (at least two) to collaborate on a common target goal. Suppose that we have a scenario with a C_a1 sharing some FOV. Thus, this C_a1 is connected to one local LF_a with other C_ax agents. At some point, it receives a fusion creation event from LF_a related to another C_a2, which means that both agents are monitoring the same track. Depending on the goals of C_a1, it may suggest the creation of a new dynamic coalition with C_a2 for the purpose of coordination to satisfy a collaborative goal. Suppose the goal of creating this coalition is to gather as much information as possible about the fused target. If C_a2 accepts the creation of this coalition, then a new temporary agent

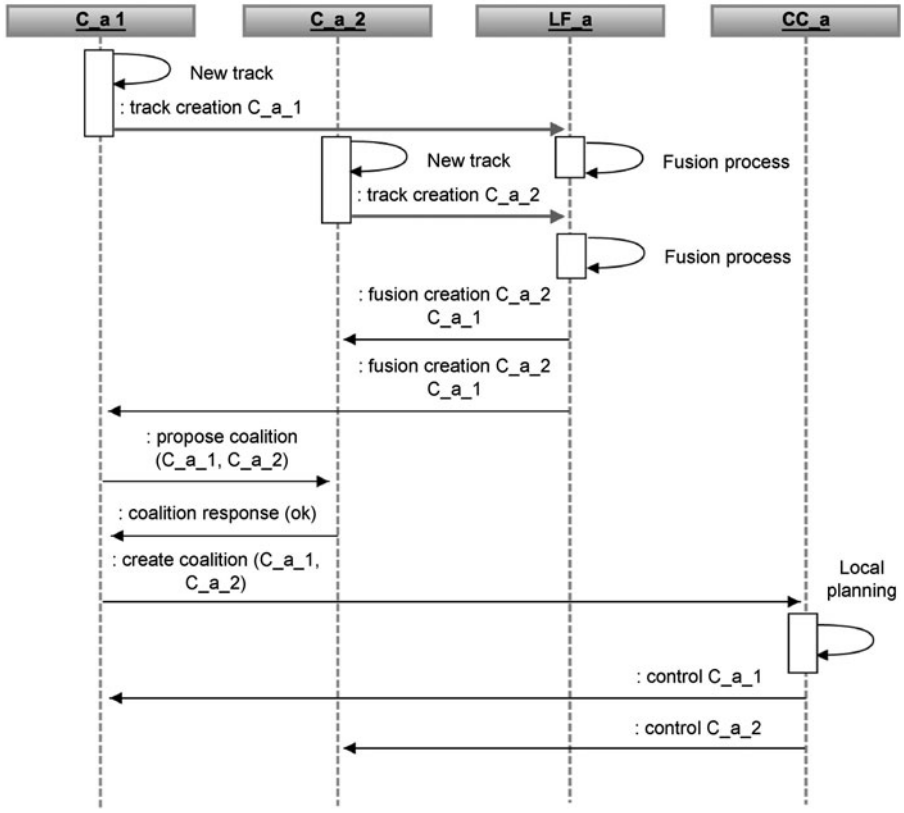

Figure 5. Example of dynamic coalition creation based on a fusion event. C_a1 and C_a2 share a common track, so they decide to create a dynamic coalition with a CC_a agent that will schedule the camera controls.

called *Coalition Agent* (CC_a) can be started. CC_a will be able to coordinate C_a1 and C_a2. This CC_a agent will start to receive the tracking and fusion information used by both C_a1 and C_a2 from the LF_a in order to decide or plan the tasks that C_a1 and C_a2 should perform. Figure 5 illustrates the complete interaction for this kind of dynamic coalition creation. Thanks to the use of CC_a to plan camera movements, different *Coalition Agents* will be able to specialise in different features, i.e. one CC_a may specialise in coordinating agents to gather the maximum information about the track, and another may be interested in gathering different track features from different cameras (face, pose, etc.).

This kind of dynamic coalition, which enables camera coordination, would normally be started by C_a agents, as described above, depending on the fusion events raised by the LF_a. However, a potential operator using the system may also deploy different CC_a assigned to a LF_a. In this case, richer coalitions with different goals can be established to improve surveillance capabilities. In this way, a CC_a that minimises the number of fusions for a given set of cameras sharing a FOV would maximise the number of tracks and zones being covered. Another possible example is a CC_a that prioritises the targets being observed according to some rules and selects the important target to be monitored by each C_a.

The use of BDI agents to support these actions is a perfectly good option, since there is a belief base (information about the environment mainly gathered by tracking and fusion events), a set of desires (like maximising the tracking targets) and a set of intentions that can be used to achieve each goal (like move the camera, create a coalition, etc.). On this occasion, the use of JADEX as the MAS framework will provide the necessary reasoning engine to execute the task required at any time depending on agent goals and beliefs.

5.2. Global Tracking Agent

The *Global Tracking Agent* (GT_a) works by receiving both tracking and fusion events from a single S_a (without overlapped FOV) and the different LF_a running in the environment (illustrated in Figure 1). All this information is integrated in this single node in order to be able to handle a global view of the environment. The centralisation of this part of the architecture is not a critical issue, since this agent does not require much computing power because all it has to do is keep an up-to-date status of the different tracks present in the environment. So, the only task assigned to this component is to receive the events, update the ongoing information and notify environment changes to

other agents, e.g. interface agents. Moreover, the use of this agent may provide other features that can be used to improve tracking continuity across the whole area covered by active cameras.

6. User Interface

The *User Interface* is probably one of the most important parts of a video surveillance system, as the system should be able to be finally monitored and controlled by an operator. Thus, a user interface displaying the status of the entire environment is necessary. The traditional interfaces used in this kind of systems mainly rely on a set of video streams displayed on one or more screens. Some sources, like active cameras, are normally controlled by a joystick to change the camera FOV. Nowadays, however, surveillance systems may also include tracking features that can provide some automated information about the scenes, like the number of tracks, their positions, certain features, etc. The benefits of using data fusion as an integral part of this architecture is that the fused output is very useful for display purposes too. So, we think that, apart from displaying tracking information, the *Interface Agent* (I_a) should also integrate data-fusion information. Displaying the fused tracks will provide the operator with a superset of non-redundant tracks without loss of information on any underlying track. Figure 6 shows an example of how this interface would work using the MAS. In this case, the I_a will display the tracking information provided by S_a agents, but it is also able to group the fused tracks in the interface, allowing the operator to easily obtain all the information about each track. Moreover, as this agent receives the tracking information (including track locations), the operator could establish restricted areas that can generate interface alerts when a track enters a protected zone.

It is necessary not only to display the real-time information captured by the system, but also to record it for later analysis. In this way, new agents can be deployed with different goals. The first may be a *Recorder Agent* (R_a) dealing with the centralised storage of raw video streams from the cameras, acting as a common digital video recorder (DVR) used in surveillance systems. This agent could be running on a remote computer with high storage capabilities, receiving the video streams from the different S_a deployed in the environment and saving the video files for each camera. Another potentially interesting agent is the *Event Recording Agent* (E_a), which can listen for both track events and video streams. With this information, this agent would be able to detect and capture interesting events (a track enters a predefined zone). So, the output would possibly be some kind of log with the different events and their respective pictures.

7. Agent communication

By using a framework like JADEX, which relies on the FIPA-ACL standard language for agent communications, we are assuming that the agents will know how to communicate, that is, understand or speak the same language. However, agents not only have to speak the same language, but also require a common ontology in order to understand each other. An ontology is a part of the agent's knowledge base that describes what kind of things an agent can deal with and how they are related to each other. So, basically, it is necessary to represent knowledge as a set of concepts within the domain of video surveillance with active cameras, and the relationships among these concepts. It would be beyond the scope of this paper to define the whole ontology for this domain. Instead, we highlight the major points

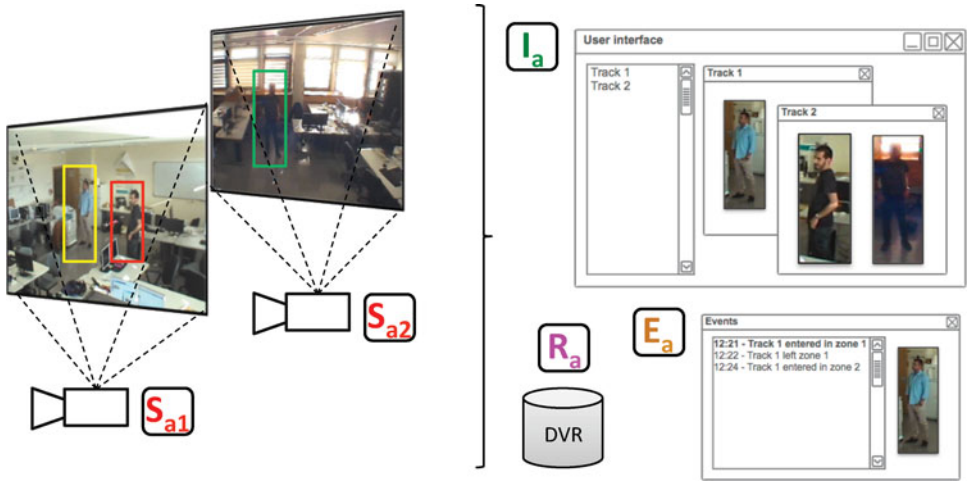

Figure 6. The benefits of using data fusion, camera coordination aside, is to present more detailed information about each track, like the information obtained from each sensor. Different *Interface Agents* to support different actions, like show events, track information and record the video streams.

of the ontology and its modelling for its later use in the JADEX framework.

The Protegé Ontology Editor (Horridge, Knublauch, Rector, Stevens, and Wroe 2004) is used to define the agent ontology. Protegé is a tool for easily defining ontologies with a graphical user interface. The OntologyBeanGenerator, a plugin for Protegé that generates Java files representing an ontology that is usable with the JADEX environment, is used to get interoperability between the designed ontology specification and the JADEX framework. Thus, FIPA/JADEX-compliant ontologies can be generated from the Protegé ontology specification files. In this case, the ontology has to be filled by subclassing some predefined classes from an abstract ontology, like *AID*, in order to define the different agents in the domain: *Concept* for defining any concept in the ontology (i.e. entity types such as Camera, Track, Event, ...) and *Agent Action* for specifying the different actions to be achieved by the agents. This is how the different concepts, agents and actions for this domain have been defined. Figure 7 shows an example of a subset of the designed ontology.

8. Validating the architecture

This kind of surveillance systems is hard to evaluate since there are many interrelated components, like camera acquisition, image processing, agent communication and reasoning, fusion techniques and so on. Thus, we would have to conduct an exhaustive analysis of every independent part in order to test the whole system. Therefore, this paper evaluates an operational prototype with some capabilities in order to check whether the proposed agent-based distributed architecture is suitable for this type of real-time sensor management and processing.

For this purpose, we have a real environment with three active cameras (C_1, C_2, C_3) placed at fixed positions. Each camera is connected to an independent server that can access its video feed and PTZ control interface. The *Sensor Controller* middleware described in Section 4.1 runs on each server, enabling each *Sensor Agent* to access the physical camera. Once this setup is ready, we can start deploying the agents using the Jadex framework running on each machine. Each server will run an instance of both *Sensor Agent* and *Control Agent*. The remaining *Local Fusion Agent* and *Global Fusion Agent* will run on a fourth computer. As we can see, this system is running in a fully distributed environment, with agents running on four different machines. Agents are also communicated through the Jadex platform, as described in Section 7. All computers are connected to the same local area network.

The purpose of the prototype is to track different targets in the proposed environment. In this way, we can test the whole architecture as all the components should work together to perform the task. PTZ camera tracking

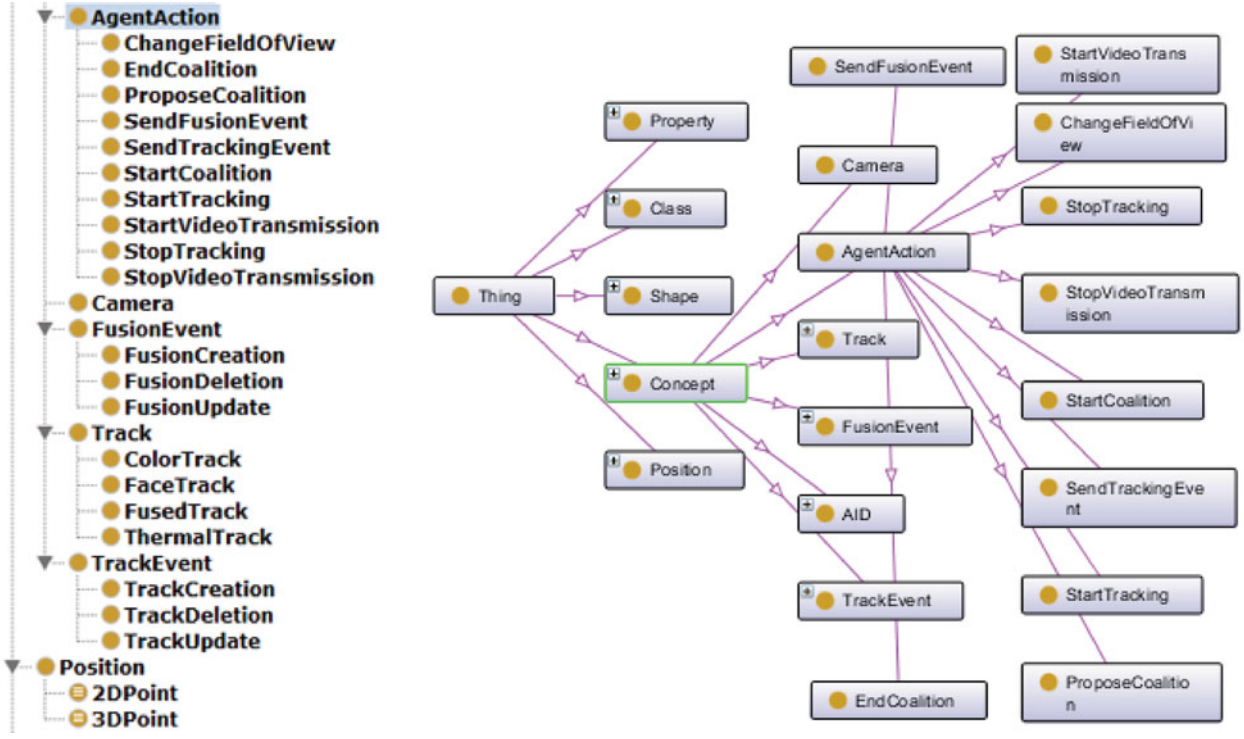

Figure 7. An ontology has to be used to define agent knowledge for proper agent understanding. In this case, the Protegé tool is used along with the OntologyBeanGenerator for easily exporting the domain to the JADEX framework.

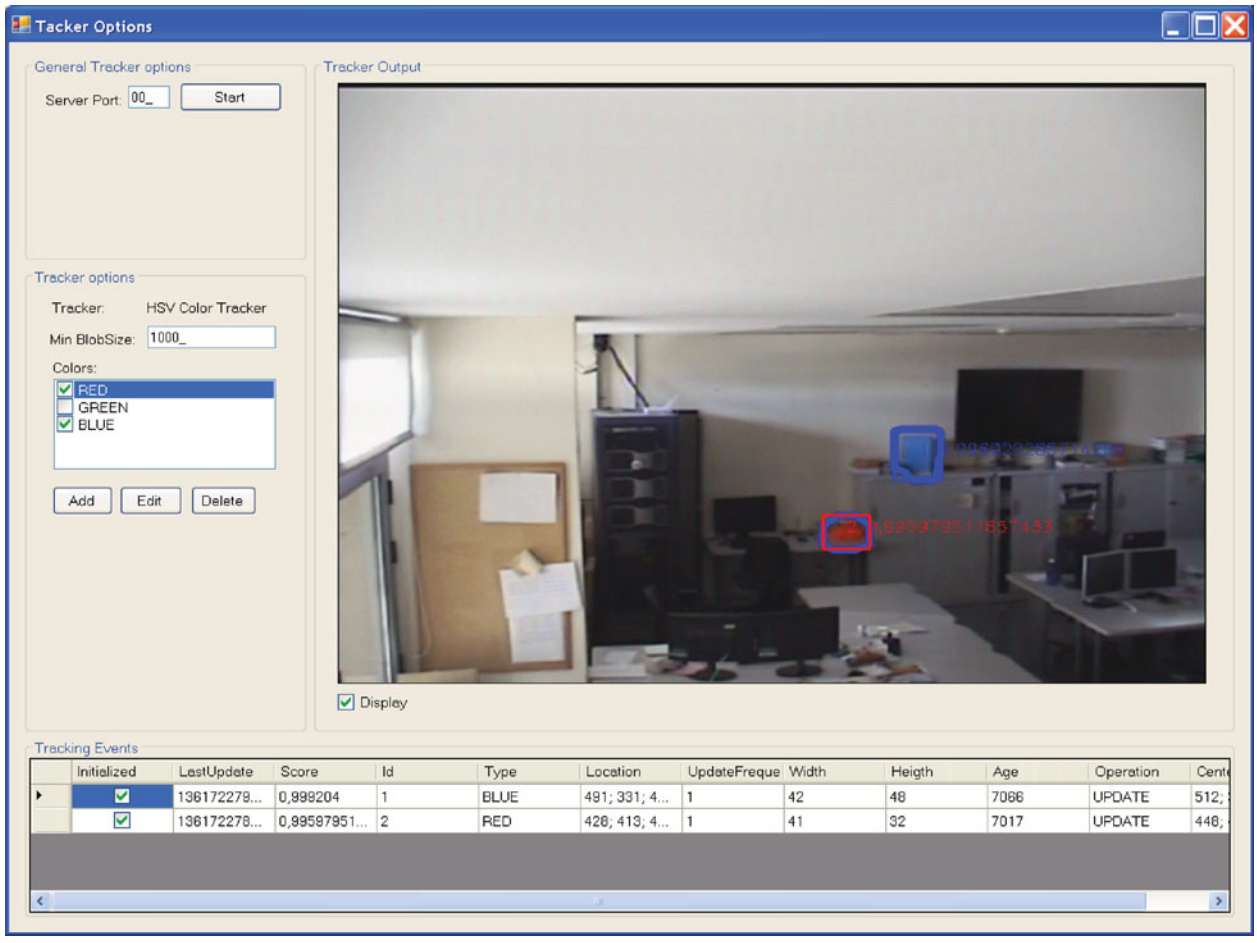

Figure 8. The *Sensor Controller* middleware is responsible for providing tracking information to *Sensor Agent*. The tracking information is delivered to the respective *Sensor* and *Fusion Agents*.

algorithms are likely to be harder to develop and tune, so we use a simple colour-based tracking algorithm for this preliminary stage. As discussed earlier, the tracking information is provided by the *Sensor Controller* middleware, which enables agents to access tracking information without processing images, among other things. Figure 8 illustrates an example of this controller running on one server to provide the colour-based tracking. The agents receiving the tracking information can start to manage sensors and coordinate with other agents. In this case, coordination will consist of tracking different targets when possible. So, the expected system behaviour is for each camera to track a single target.

In this experiment, there are three different targets (T_1, T_2, and T_3), with blue, red and green colours, respectively. T_1 appears in the environment at an early stage (t_1), and, being the only target, is tracked by all the cameras. At a given time (t_2), the second track T_2 enters the scenario, forcing cameras to collaborate. In this case, each agent has colour priorities, so they collaborate to maximise both the overall priority and targets covered when there are shared tracks in the FOV. So, the track-to-camera association is forged in real time when the fusion events indicate that there are cameras looking at the same targets. At this time, C_1 and C_3 switches their FOV to the new target T_2, and C_2 keeps its gaze on T_1. Finally, t_3, T_3 changes the reasoning and re-associates the targets, where each camera should now look at a single target. According to the colour priorities, C_1 to T_2, C_2 to T_1 and C_3 to T_3. Figure 9 illustrates an example of this behaviour. Also a video demonstration with this experiment is available at http://www.giaa.inf.uc3m.es/miembros/alvaro/ptz_mas_2013.mov for a better understanding.

This *a priori* apparently simple experiment covers many distributed processes working together, ranging from the *Sensor Controller* acquiring frames, performing tracking and allowing PTZ movements, to the *Sensor Agent* receiving tracking and fusion information to determine which track to follow and then drive the PTZ camera. The results in the proposed scenario suggest that the agents can

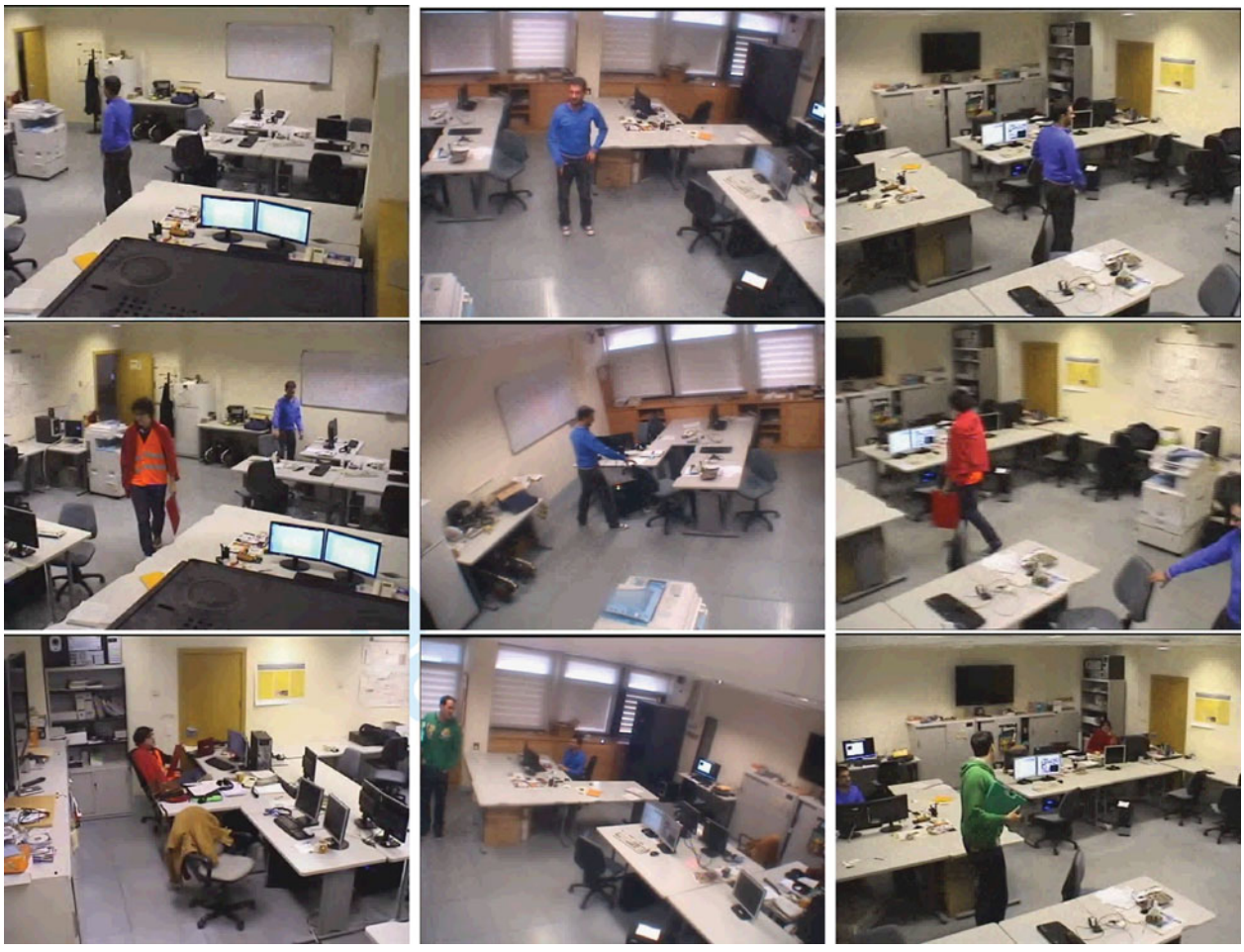

Figure 9. Example of multi-agent behaviour as new targets appear in the environment. The columns show the FOV of each camera, C_1, C_2 and C_3, from left to right. Rows represent a given moment in time, t_1, t_2 and t_3, from top to bottom, with the different targets followed by each camera.

follow the desired targets and perform reasoning when there is more than one track in the same FOV, so the proposed multi-agent architecture is suitable for this kind of environments. Moreover, the fusion layer affords a more flexible solution, as it is able to trigger reasoning between agents when necessary.

The next step for validating the architecture and gathering quantitative results typically involves measuring tracking capabilities and goal achievement success.

For this purpose, a training set of situations (mainly videos) is necessary to validate the system under different conditions. Many data sets are available for this purpose, and the Performance Evaluation of Tracking and Surveillance (PETS; http://pets2012.net/) conference provides a data set that is well known in the visual surveillance community. Such data sets consist of a group of (relatively well) synchronised videos of everyday situations taken from different cameras. Along with the video streams, they contain the calibration parameters of each camera required to reference camera location, and sometimes the ground truth information of each track. This information is very helpful when designing a tracking system, as results are comparable, based on the same conditions, with research community outcomes. However, when dealing with a system mainly focusing on the management of active cameras as the main part of the video surveillance system, the plot thickens. Such data sets become unusable for the simple reason that an active camera can change its FOV, and each camera movement will provide different video feeds. This will throw out the calibration parameters, and the location-fixed video sequences will now be meaningless. So neither the standard data sets nor the real world can provide enough repeatability for achieving quantitative results.

As the traditional techniques for validating video surveillance systems are deficient, we have to search for alternatives capable of satisfying the complex active camera scenario. The research community has proposed several approaches, but the use of virtual worlds is perhaps

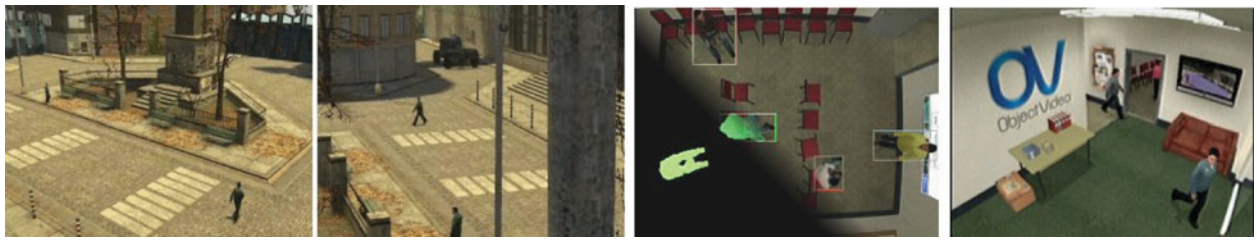

Figure 10. The use of virtual environments can provide synchronised video streams from multiple cameras (including active cameras), low-level features like automatic ground truth generation, different ambient conditions, automatic simulation of pedestrians, etc.

the best suited (Qureshi and Terzopoulos 2007). The ObjectVideo Virtual Video Tool (OVVV), which is available online for free at http://development.objectvideo.com/ (Taylor, Chosak, and Brewer 2007), serves this purpose. It is basically able to create virtual environments in which you can place different types of cameras, including active, still and fisheye cameras, or even cameras placed in unmanned aerial vehicles (UAVs). It will generate synthetic video feeds for processing. It may also provide information like automatic ground truth generation for each visible track and its three-dimensional world location, so you can focus on high-level coordination algorithms rather than trying to tune your real-world environment. Figure 10 shows some images generated by this tool.

So, we will use the OVVV tool for the next development stage of the MAS architecture in order to focus mainly on the problem of active camera coordination. This will provide for the necessary repeatability in the experiments in order to improve agents working under the same circumstances, and create several environments with different numbers of and arrangements of PTZ cameras. The *Sensor Agent* described in the architecture, which has been designed for real-world sensors, is still applicable for this kind of environments, as all the changes are made in the *Sensor Controller* through adaptation to the virtual camera interfaces.

9. Conclusions and further work

In this paper, we have defined a MAS integrating a fusion system in order to support the distributed control of an active camera sensor network. During the design of this architecture, we have found that there are several aspects to be defined or considered in order to get a real operational prototype, as this topic implies multiple disciplinary approaches. This paper deals with the problem of controlling active cameras, by both identifying the different low-level algorithms that can be used for active camera tracking, calibration, etc. and proposing a high-level architecture based on a MAS. The MAS is responsible for coordinating the active cameras to achieve different operator-specified goals. As the cameras can dynamically collaborate depending on their goals, we have proposed a fusion architecture integrated with the MAS that will allow the agents to know when collaboration is possible. This will enable the dynamic creation of coalitions with specialised agents for solving a particular problem. Also, we have identified possible practical concerns when developing this kind of architecture, like the use of a MAS framework, and how the communication should be defined. The experiments conducted suggest that the proposed architecture is suitable for this real-time environment, as the cameras can be handled by a set of cooperative agents.

In the future, we plan to extend the architecture definition for integration with multiple sensor types, like indoor/outdoor localisation systems, thermal cameras, other sensing devices like RFID or NFC, or also external devices performing activity recognition (like mobile phones). The integration of this kind of different inputs can improve the system by allowing another level of perception and coordination. For example, rather than just tracking information, camera agents can employ accurate fused location information provided by indoor localisation systems to track objects. Or in a health-care monitoring system, a remote device can detect a person that has a fall. The agents can then react to this event and start monitoring the target person. All these inputs can be easily adapted to the existing architecture by extending the *Control Agent* perceptions and creating new *Sensor Agents* to input information to the architecture.

Acknowledgements

This work was supported in part by Projects MINECO TEC2012-37832-C02-01, CICYT TEC2011-28626-C02-02 and CAM CONTEXTS (S2009/TIC-1485).

Notes on contributors

Alvaro Luis Bustamante received his B.S degree in computer science from Universidad Carlos III de Madrid in 2008, his M.S degree in artificial intelligence in 2009 and he is currently doing his Ph.D. degree in artificial intelligence at the same university as Research Fellow of the Applied Artificial Intelligence Group (GIAA). His main interests are artificial intelligence applied to image data processing, video surveillance automatic system, computer vision, and multi-agent systems.

Jose Manuel Molina Lopez received a degree in Telecommunication Engineering from the Universidad Politecnica de Madrid in 1993 and a Ph.D. degree from the same university in 1997. He joined the Universidad Carlos III de Madrid in 1993 where, actually, he is Full Professor in the Computer Science Department. Currently, he leads the Applied Artificial Intelligence Group (GIAA, http://www.giaa.inf.uc3m.es) involved in several research projects related with ambient intelligence, surveillance systems and context based computing. His current research focuses in the application of soft computing techniques (Multiagents Systems, Evolutionary Computation, Fuzzy Systems) to surveillance, ambient intelligence, air traffic management and e-commerce. He is the author of up to 50 journal papers and 200 conference papers.

M. A. Patricio received his B.S. degree in computer science from the Universidad Politecnica de Madrid in 1991, his M.S. degree in computer science in 1995, and his Ph.D. degree in artificial intelligence from the same university in 2002. He has held an administrative position at the Computer Science Department of the Universidad Politecnica de Madrid since 1993. He is currently Associate Professor at the Escuela Politecnica Superior of the Universidad Carlos III de Madrid and Research Fellow of the Applied Artificial Intelligence Group GIAA). He has carried out a number of research projects and consulting activities in the areas of automatic visual inspection systems, texture recognition, neural networks, and industrial applications.

References

Akyildiz, I., and Kasimoglu, I. (2004), 'Wireless Sensor and Actor Networks: Research Challenges', *Ad Hoc Networks*, 2(4), 351–367.

Bellifemine, F., Poggi, A., and Rimassa, G. (2001), 'JADE: A FIPA2000 Compliant Agent Development Environment', in *Proceedings of the Fifth International Conference on Autonomous Agents*, pp. 216–217.

Bevilacqua, A., Di Stefano, L., and Azzari, P. (2006), 'People Tracking Using a Time-of-flight Depth Sensor', in *IEEE International Conference on Video and Signal Based Surveillance*, pp. 89–89.

Braubach, L., Lamersdorf, W., and Pokahr, A. (2003), 'Jadex: Implementing a BDI-Infrastructure for Jade Agents', *EXP*, 3, 76–85.

Bustamante, A., and Molina, J. (2011), 'Multi-Camera Control and Video Transmission Architecture for Distributed Systems', *User-Centric Technologies and Applications*, 94, 37–45.

Bustamante, A.L., Molina López, J.M., and Patricio, M.A. (2011), 'MIJ2K: Enhanced Video Transmission Based on Conditional Replenishment of JPEG2000 Tiles With Motion Compensation', *Journal of Visual Communication and Image Representation*, 22(4), 332–344.

Castanedo, F., García, J., Patricio, M.A., and Molina, J.M. (2010), 'Data Fusion to Improve Trajectory Tracking in a Cooperative Surveillance Multi-Agent Architecture', *Information Fusion*, 11(3), 243–255.

Castanedo, F., Patricio, M.A., García, J., and Molina, J.M. (2006), 'Extending Surveillance Systems Capabilities Using BDI Cooperative Sensor Agents', in *Proceedings of the 4th ACM International Workshop on Video Surveillance and Sensor Networks*, New York: ACM Press, p. 131.

Collins, R.T., Lipton, A.J., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., Tolliver, D., Enomoto, N., Hasegawa, O., Burt, P., and Wixson, L. (2011), 'A System for Video Surveillance and Monitoring', *System*, 69(CMU-RI-TR-00-12), 573–575.

Comaniciu, D., and Ramesh, V. (2000), 'Robust Detection and Tracking of Human Faces With an Active Camera', in *Proceedings of the Third IEEE International Workshop on Visual Surveillance*, pp. 11–18.

Everts, I., Sebe, N., and Jones, G. (2007), 'Cooperative Object Tracking With Multiple PTZ Cameras', in *Proceedings of the 14th International Conference on Image Analysis and Processing*, pp. 323–330.

Galego, R., Bernardino, A., and Gaspar, J. (2012), 'Auto-Calibration of Pan-Tilt Cameras Including Radial Distortion and Zoom', *Advances in Visual Computing*, 7431, 169–178.

Garcia, J., Carbo, J., and Molina, J. (2005), 'Agent-Based Coordination of Cameras', *International Journal of Computer Science and Applications*, 2(1), 33–37.

Gascueña, J., and Fernández-Caballero, A. (2011), 'On the Use of Agent Technology in Intelligent, Multisensory and Distributed Surveillance', *Knowledge Engineering Review*, 26(02), 191–208.

Georgeff, M., Pell, B., Pollack, M., Tambe, M., and Wooldridge, M. (1999), 'The Belief-Desire-Intention Model of Agency', in *Proceedings of the 5th International Workshop on Intelligent Agents V: Agents Theories, Architectures, and Languages*, pp. 1–10.

Hall, D., and Llinas, J. (2001), 'Multisensor Data Fusion', in *Handbook of Multisensor Data Fusion*, eds. M.E. Liggins, D.L. Hall, and J. Llinas, Boca Raton, FL: CRC Press, pp. 1–14.

Hampapur, A. (2008), 'Smart Video Surveillance for Proactive Security', *IEEE Signal Processing Magazine*, 25(4), 136.

Han, J., and Bhanu, B. (2007), 'Fusion of Color and Infrared Video for Moving Human Detection', *Pattern Recognition*, 40(6), 1771–1784.

Heikkila, J. (2000), 'Geometric Camera Calibration Using Circular Control Points', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10), 1066–1077.

Horridge, M., Knublauch, H., Rector, A., Stevens, R., and Wroe, C. (2004), *A Practical Guide to Building OWL Ontologies Using the Protégé-OWL Plugin and CO-ODE Tools Edition 1.0*, Manchester: University of Manchester.

Jones, G., Renno, J., and Remagnino, P. (2002), 'Auto-Calibration in Multiple-Camera Surveillance Environments', in *Third IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pp. 40–47.

Krahnstoever, N., Yu, T., Lim, S., Patwardhan, K., and Tu, P. (2008), 'Collaborative Real-Time Control of Active Cameras in Large Scale Surveillance Systems', in *Workshop on Multi-Camera and Multi-Modal Sensor Fusion Algorithms and Applications*.

Liggins, M., Hall, D., and Llinas, J. (2008), *Handbook of Multisensor Data Fusion: Theory and Practice* (Vol. 22), Boca Raton, FL: CRC Press.

Lim, S., Elgammal, A., and Davis, L. (2003), 'Image-Based Pan-Tilt Camera Control in a Multi-Camera Surveillance Environment', in *ICME*, 3, 645–648.

Manyika, J., and Durrant-Whyte, H. (1994), *Data Fusion and Sensor Management: A Decentralized Information-Theoretic Approach*, Upper Saddle River, NJ: Prentice Hall.

Marchesotti, L., Messina, A., Marcenaro, L., and Regazzoni, C. (2003), 'A Cooperative Multisensor System for Face Detection in Video Surveillance Applications', *Acta Automatica Sinica*, 29(3), 423–433.

Meer, P. (2003), 'Kernel-Based Object Tracking', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5), 564–577.

Molina López, J.M., García Herrero, J., Jiménez Rodríguez, F.J., and Casar Corredera, J.R. (2003), 'Cooperative Management of a Net of Intelligent Surveillance Agent Sensors', *International Journal of Intelligent Systems*, 18(3), 279–307.

Patricio, M.A., Carbó, J., Pérez, O., García, J., and Molina, J.M. (2007), 'Multi-Agent Framework in Visual Sensor Networks', *EURASIP Journal on Advances in Signal Processing*, 2007, 1–22.

Pavón, J., Gómez-Sanz, J., Fernández-Caballero, A., and Valencia-Jiménez, J.J. (2007), 'Development of Intelligent Multisensor Surveillance Systems With Agents', *Robotics and Autonomous Systems*, 55(12), 892–903.

Petrov, P., Boumbarov, O., and Muratovski, K. (2008), 'Face Detection and Tracking With an Active Camera', in *4th International IEEE Conference on Intelligent Systems*, pp. 14–34.

Possegger, H., Ruther, M., Sternig, S., Mauthner, T., Klopschitz, M., Roth, P., and Bischof, H. (2012), 'Unsupervised Calibration of Camera Networks and Virtual PTZ Cameras', in *17th Computer Vision Winter Workshop*.

Puwein, J., Ziegler, R., Ballan, L., and Pollefeys, M. (2012), 'PTZ Camera Network Calibration from Moving People in Sports Broadcasts', in *2012 IEEE Workshop on Applications of Computer Vision (WACV)*, pp. 25–32.

Qureshi, F., and Terzopoulos, D. (2007), 'Surveillance in Virtual Reality: System Design and Multi-Camera Control', in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8.

Qureshi, F., and Terzopoulos, D. (2011), 'Proactive PTZ Camera Control', in *Distributed Video Sensor Networks*, eds. B. Bhanu, C.V. Ravishankar, A.K. Roy-Chowdhury, H. Aghajan, and D. Terzopoulos, London: Springer, pp. 273–287.

Shibata, M., Yasuda, Y., and Ito, M. (2008), 'Moving Object Detection for Active Camera Based on Optical Flow Distortion', in *Proceedings of the 17th IFAC World Congress*, pp. 14720–14725.

Singh, V.K., Atrey, P.K., and Kankanhalli, M.S. (2007), 'Coopetitive Multi-Camera Surveillance Using Model Predictive Control', *Machine Vision and Applications*, 19(5–6), 375–393.

Snidaro, L., Foresti, G., Niu, R., and Varshney, P. (2004), 'Sensor fusion for video surveillance', in *Proceedings of the Seventh International Conference on Information Fusion*, pp. 739–746

Taylor, G., Chosak, A., and Brewer, P. (2007), 'OVVV: Using Virtual Worlds to Design and Evaluate Surveillance Systems', in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8.

Wesson, R., Hayes-Roth, F., Burge, J., Stasz, C., and Sunshine, C. (1981), 'Network Structures for Distributed Situation Assessment', *IEEE Transactions on Systems, Man and Cybernetics*, 11(1), 5–23.

Zhang, Z. (2000), 'A Flexible New Technique for Camera Calibration', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11), 1330–1334.