Noname manuscript No.
(will be inserted by the editor)

A Multi-agent Architecture Based On The BDI Model For Data Fusion In Visual Sensor Networks

Federico Castanedo · Jesús García · Miguel A. Patricio · José M. Molina

the date of receipt and acceptance should be inserted later

Abstract The newest surveillance applications attempt more complex tasks such as the analysis of the behavior of individuals and crowds. These complex tasks may use a distributed visual sensor network in order to gain coverage and exploit the inherent redundancy of the overlapped field of views. In this article, a Multi-agent architecture based on the Belief-Desire-Intention (BDI) model for processing the information and fuse data in a distributed visual sensor network is presented.

Instead of exchanging raw images between the agents involved in the visual network, local signal processing is performed and only the key observed features are shared. After a registration or calibration phase, the proposed architecture performs tracking, data fusion and coordination. Using the proposed Multi-agent architecture, we focus on the means of fusing the estimated positions on the ground plane from different agents which are applied to the same object. This fusion process is used for two different purposes: (1) to obtain a continuity in the tracking along the field of view of the cameras involved in the distributed network, (2) to improve the quality of the tracking by means of fusion techniques, and by discarding non reliable sensors.

Experimental results on two different scenarios show that the designed architecture can successfully track an object even when occlusions or sensor's errors take place. The sensor's errors are reduced by exploiting the inherent redundancy of a visual sensor network with overlapped field of views.

Keywords Distributed Visual Sensor Networks, Multi-agent Systems, Data Fusion.

This work was partially supported by projects CICYT TIN2008-06742-C02-02/TSI, CICYT TEC2008-06732-C02-02/TEC, SINPROB, CAM MADRINET S-0505/TIC/0255 and DPS2008-07029-C02-02

Applied Artificial Intelligence Group
www.giaa.inf.uc3m.es
Department of Computer Science
University Carlos III of Madrid
Spain
E-mail: {fcastane,jgherrer,mpatrici}@inf.uc3m.es, molina@ia.uc3m.es

1 Introduction

Automatic capture and analysis of computer vision scenes is a highly active area of research due to both the number of potential applications and its inherent complexity. The number of potential applications, the scientific complexity, the speed and price of the current hardware, and the efforts made in the field of security issues have had a direct impact on the development of intelligent visual systems [1]. Computer vision applications can be roughly grouped under three categories [1]: (1) surveillance applications, (2) control applications and (3) analysis applications. In this article, we will focus on surveillance applications. The newest surveillance applications attempt more complex tasks such as the analysis of the behavior of individuals and crowds. These complex tasks may use a distributed visual sensor network in order to gain coverage and exploit the redundancy. In this article, a distributed visual sensor architecture developed using a BDI agent model is presented. A BDI agent model is a type of agent architecture containing explicit representations of beliefs, desires and intentions.

On the one hand, sensor networks are related to spatially distributed multi-sensor environments. In the case of camera sensors, they cope with distributed computer vision techniques known as Visual Sensor Networks (VSNs) [2]. A distinguishing feature of visual sensors, as compared with other types of sensor such as pressure sensors, microphones or thermometers, is the considerable amount of data which they generate, which makes it essential to have local processing which delivers the information on a conceptualized level. This distributed information must be integrated and fused with data fusion[1] techniques with the aim of obtaining a global view of the environment being monitored.

On the other hand, Multi-agent systems are composed of several intelligent software agents. A software agent [3] is a computational process that has several characteristics: (1) *"reactivity"* (agents can perceive and respond to a changing environment), (2) *"social ability"* (the means by which agents interact with other agents) and (3) *"proactivity"* (through which agents have a goal-directed behavior).

As a result, Multi-agent VSNs have been introduced in an attempt to achieve more robust, flexible and adaptable computer vision systems. In Multi-agent VSNs, visual sensing is the mechanism or process whereby the system can be influenced by the environment around it. In order to develop a process for the automatic capture and analysis of computer vision scenes, we present a distributed visual sensor network based on the BDI Multi-agent model. The use of a Multi-agent architecture for processing the information in a VSN allows the development of an open architecture which is easy to scale. We could easily add new agents (with different or the same goals) in the Multi-agent system following the proposed BDI specification and the communication messages. This issue is demonstrated in the results of the second scenario which show how the error is reduced when more visual sensors are employed. Also a standard based Multi-agent architecture, would allow us to inter-operate with third party developers. Since each agent has a unique name and direction the communication with other agents could be easily performed by messages.

The Multi-agent architecture presented in this paper, provides several advantages which could improve a VSN:

[1] The term data fusion is widely used in the literature and appears also as information fusion, data combination or data integration. In this paper, we refer to multi-sensor data fusion, and the concept means how to combine redundant data from multiple visual sensors (looking at the same scene) in an optimal way.

- The improvement of accuracy and performance due to cooperation, which could be easily implemented with messages. For example, a better tracking accuracy as a result of the agents cooperation.
- Usually agents solve problems which are part of a global issue. For example, most of them perform local tracking using local image processing, but we also need a global view of the tracking being performed by all of the agents. In a VSN, this global view consist in the surveillance process of the environment being monitored which is achieved using a data fusion process with techniques available in the computer vision literature.
- The easily sharing of information between agents, thus allowing them to correct errors by their ability to make an explicit coordination.
- The exploitation of the overlapping field of view of the visual sensors, in order to achieve better global tracking.
- As tracks[2]or objects are commonly lost, a data fusion process is justified to avoid tracking inconsistency. This data fusion process is also implemented in a BDI agent and the required information is obtained by other agent's messages.

In this paper, we show an implementation of a visual sensor network using the BDI Multi-agent model. The proposed architecture performs tracking, data fusion and coordination. Using this architecture, we focus on how to fuse the tracks from different agents which pertain to the same object. This fused information is used for two different purposes: (1) to obtain continuity in the tracking along the field of view of the cameras involved in the distributed network, which provides extended coverage equivalent to that of a bigger sensor, and (2) to improve the quality of tracking using data fusion techniques, by discarding non reliable sensors. In the experiments section, results in two different scenarios with the objective of validating the previous purposes are presented. This article describes the general steps which are necessary to take into account for processing the information in a VSN: (1) local processing on the visual sensors, (2) registration and camera calibration and (3) data fusion. These steps which are common to traditional multi-camera environments are defined using the BDI Multi-agent model as beliefs, desires and intentions.

The main contribution of this article is to define a complete architecture for implementing a VSN using the Multi-agent paradigm, specifically the BDI agent model.

The outline of the article is organized as follows: The following section describes related studies in the area of Multi-agent systems as applied to visual sensor networks. Section III addresses the tasks of image and data processing in the VSNs. Following from this, the details of the architecture of Cooperative Sensor Agents are described in section IV. Section V presents experiments carried out by using the proposed architecture in two different scenarios: an indoor scenario with 3 cameras and a basketball match scenario with 5 cameras. Finally, section VI concludes the paper.

2 Related Works

The work on VSNs can be divided into two categories based on visual configuration: non-overlapping fields of view [28] and overlapping fields of view. In this work, we assume an overlapping camera configuration and focus on fusing the local coordinate frames of multiple cameras into a global view by the use of a BDI Multi-agent system.

[2] A track is an ordered set of positions all of them generated from the same object.

The application of Multi-agent systems in computer vision have been explored in several works. Berge-Cherfaoui and Vachon [4] proposed a Multi-agent approach based on the blackboard model. The blackboard model is one of the first Multi-agent communication models, although it is less flexible than current communication models based on FIPA-ACL[3] [29].

There are several coordination models available for Multi-agent systems. An approach based on the model of information-subscription coordination is used in the European project MODEST [5]. The MODEST traffic surveillance system deployed four cameras along a bridge in Brussels and it is FIPA [6] compliant[4]. The coordination mechanism uses the directory facilitator (DF) of the FIPA platform [7]. The designers proposed an extension of Semantic Language (SL) in order to take into account uncertainty and MPEG-7 descriptors.

A different coordination model based on a contract network protocol was used by Graf and Knoll [8]. They distinguish between two types of agents: masters and slaves which are connected using a contract network. However, the main difference in their approach is that they use a proprietary communication language, not a standard one, which makes interoperability with other systems problematic.

There are also other papers without an explicit coordination scheme. In [9], a Multi-agent architecture and Hidden Markov Models are used, in order to understand scene dynamics, by merging the information streamed by multiple cameras. However, they use a very simple communication mechanism and do not adopt any communication standard. The system also produces a computational overload due to classification issues. In Monitorix [10] the authors use neural networks and Multi-agent systems as a classification approach. They use a FIPA platform [6] and a FIPA ACL communication language. In this project, the fields of view of the cameras do not overlap. The work of [11], which is very similar to our work, proposes an architecture for implementing algorithms which understand the scene in the visual surveillance domain. The aim of their work is to obtain a high level description of the events observed by multiple cameras. As in our work, their architecture associates each camera to a software agent and the tracking is performed on the ground plane. However, they create one agent for each object detected in the scene. Dynamic memory architecture [12] is proposed by Matsuyama et.al to coordinate agents in a real time cooperative multi-target tracking system [13] using Pan-Tilt-Zoom[5] (PTZ) cameras. In their framework they use a group of active vision agents which represent a logical model for a network-connected computer for each active camera and each agent controls a PTZ camera. They designed a three-layered interaction architecture in which each agent can either be searching or tracking. They also defined protocols for cooperative tracking between the agents which perform 3D tracking, but do not use a standard Multi-agent architecture or communication language. The work of Bakhtari et. al [14] also propose a Multi-agent system for online sensing system reconfiguration of an active vision system To combine data from a homogeneous network of sensors for tracking purposes, Marchesotti et.

[3] FIPA is the acronym of *The Foundation for Intelligent Physical Agents* (http://www.fipa.org), an IEEE Computer Society standards organization that promotes agent-based technology and the interoperability of its standards with other technologies. ACL is the acronym of *Agent Communication Language*, the communication language proposed by FIPA.

[4] Which means that is conformed with the FIPA standards.

[5] Pant-Tilt-Zoom is the name given to cameras with zooming, horizontal (Pan) and vertical (Tilt) movement capabilities.

al [15] propose an agent-based architecture. They use the color and position of each detected target in order to identify it. A paper focused on the selection of the most appropriate camera for each task, taking into account occlusion is presented in [16]. Their work differs from ours, as objects are tracked by only one camera at any given time.

However, most of the related work focuses on how to solve different visual sensor problems; there are fewer papers focusing on how to build an intelligent VSN architecture which takes into account all the related problems (tracking, data fusion, coordination).

3 Image and Data Processing Tasks in the VSN

In a VSN, the images generated in every visual sensor, which have huge amount of useless raw data, must be processed with image and video analysis algorithms. A mono-sensor approach generally suffers from limited coverage and low performance in tracking targets in certain conditions of occlusion, initialization, lighting changes, and shadows. Therefore the main objectives of our proposed BDI Multi-agent architecture are to reduce these problems of limited coverage and low performance. We describe three different types of agents which together are able to make a visual sensor network operate successfully: (1) surveillance-sensor agents, (2) fusion agents and (3) interface agents.

Although computer vision research has achieved great advances in recent decades in building robust and sophisticated solutions for image analysis, the task of fusing the processing results of different nodes in an accurate, reliable and decentralized schema has been less studied.

The main advantages of having a visual sensor network are the coverage of an extended area and the improvement in the process of inferring visual information which takes advantage of the overlapped fields of view of different sensors and the redundant information.

In order to achieve these goals, some of the basic factors which need to be taken into account are: (1) the transformation of the information into a common coordinate frame (global coordinates), known as calibration or registration, (2) synchronization to a common time, (3) the removal of corrupted or erroneous objects by analysis and (4) the combination of estimations obtained by different sensors.

3.1 Acquiring Information. Local Processing

In our proposal, each visual sensor (camera) is associated using a frame grabber with a software agent, which acquires video frames to estimate local tracks and interpret its local scene. A typical configuration of processing modules in a surveillance-sensor agent is arranged in a pipe-line structure consisting of several modules [17][18]. It directly interfaces with the image stream coming from a camera and extracts the tracking information of the mobile objects in the current frame. The interface between adjacent modules is symbolic data and it is set up in a such a way that for each module there are different interchangeable algorithms.

The main modules of the application are: (1) a detection process for moving objects; (2) an association process; (3) a prediction process; (4) a track manager; (5) a track

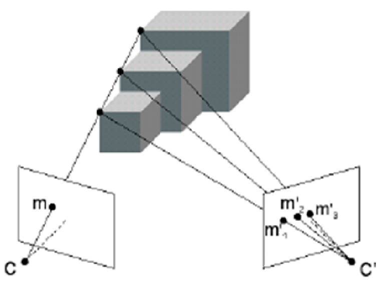

Fig. 1 When the scene is projected on the image plane information on the depth of the objects is lost.

updater. The detection process (1) for moving objects must give a list of blobs[6] that are found in a frame. This list must contain information about the position and size of each blob. The pixels contained in a blob must be locally distinguishable from pixels which are not part of the blob. Within the tracking process and continuing from the list of blobs obtained by the previous module, the association process (2) will solve the problem of blob-to-track multi-assignment, where several (or zero) blobs may be assigned to the same track and simultaneously several tracks could overlap and share common blobs. Therefore, the problem we need to solve here is the decision on the most appropriate grouping of blobs and the way they are assigned to each track for every processed frame [19]. The prediction process (3), uses the association made by the tracking process and estimates on where each track will move to during the next frame; this prediction will be used by the tracking process in order to make the association. The track manager module (4) eliminates those blobs that have not been associated with any track and do not initialize new ones, since they are considered to be noise. The last main module, the track updater (5), updates the tracks obtained in the latest frame with the information obtained from the previous modules for this frame. The symbolic data between adjacent modules is normally filtered using the contextual information of the scene in order to improve the tracking system in complex situations [20].

3.2 Registration and Camera Calibration

The process of combine different field of views and generate a global view of the area under observation is known as registration. The registration process involves the geometric camera calibration which consist in the problem of estimating the intrinsic and extrinsic parameters of a camera.

At the fusion stage, the first step carried out by the fusion agent is the time-space alignment. Spatial alignment means using a common coordinate frame to represent the objects, which is done with the registration process to project the coordinates from a local image plane to global coordinates. Visual sensor calibration requires a certain number of correspondences in order to establish a geometric transformation between different views. The calibration process, inspired by the human capability of inferring 3D structures from 2D information, uses a geometrical projection known as the pinhole

[6] A blob is a connected set of pixels in an image which are assumed that correspond to an object.

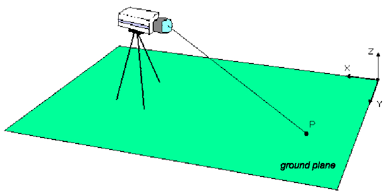

Fig. 2 Planar projection on ground plane.

model or perspective projection model [25]. In this model, the creation of a digital image is divided into two processes in order to obtain the digital image itself: (1) The scene projection on the image plane (sensor), and (2) the sample and digitalization of this plane. Each process can be stated as a change in the coordinate system.

The algorithms that reconstruct the 3-D structure from an image, or calculate the position of the objects' in space need equations that can match the 3-D points with their corresponding 2-D projections. Even though those equations are defined by the projection, it is normal to suppose that the 3-D points can be defined by a different coordinate system than that of the camera and in addition, it is necessary to relate the coordinates from a point in the image with the corresponding points in the projection system given by the camera. For this task, a set of points with known positions in some fixed world coordinates are used. Thus, camera calibration can be modeled as an optimization process, where the difference between the image field of view point and the point in the global coordinate system is minimized with respect to the camera's intrinsic and extrinsic parameters. The task of obtaining the value of these parameters is known in computer vision as the camera calibration process.

Some camera calibration techniques have been proposed, being Tsai [25], Heikkilä [26] and Zhang [27] the most used and important. In this work we employ the Tsai's camera model [25] for calibration issues.

Tsai's camera model is based on the pinhole model of 3D-2D perspective projection with 1st order radial lens distortion. The model has 11 parameters, five intrinsic or internal parameters (which relate the reference system of the camera to the image) and six extrinsic or external parameters (those which relate both 3-D reference systems). On one hand, the five intrinsic or internal parameters consist of:

– f: effective focal length of the pinhole camera.
– kappa1: 1st order radial lens distortion coefficient.
– sx: scale factor to account for any uncertainty in the framegrabber's resampling of the horizontal scan-line.
– Cx, Cy: coordinates of the center of radial lens distortion and the piercing point of the camera coordinate frame's Z axis with the camera's sensor plane.

On the other hand, the six extrinsic or external parameters consist of:

– Rx, Ry, Rz: rotation angles for the transform between the world coordinate and the camera coordinate frames.
– Tx, Ty, Tz: translational components for the transform between the world coordinate and camera coordinate frames.

The pinhole model establishes the mathematical relation between the spatial points and their equivalents in the camera image. The image making process is the projection

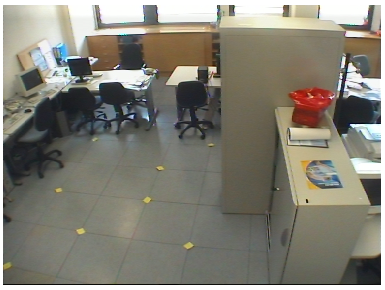

Fig. 3 Calibration template used for camera 2 in the first scenario of the experiments section.

of a three-dimensional world onto a two-dimensional image. Therefore, through perspective transformation, the correspondence between the real world and the image can be obtained. Nevertheless, with inverse perspective transformation there is more than one possible correspondence (because in the direct transform depth information is lost) and for each image point there is an infinite set of three-dimensional environmental points (see Fig. 1). Additional information is necessary to obtain a certain space point from the image which gives the depth loss suffered in the direct transformation. Binocular disparity (the angular difference between images of the same features in different image planes) has been considered one of the most important sources in 3-D information recovery [21]. Stereo vision systems are able to infer information about the 3-D structure and the distance of a scene obtained from two or more images taken from different angles.

Although stereo-vision is the most common method for inferring 3D information in computer vision systems, there are approximations which simplify the pin-hole model by adding some restrictions as to the initial knowledge of the environment. In our experimentation, we have followed the suggestion of the authors of [22], and we have developed a simplified calibration model using only one camera and adding the constraint that all the world points are in the same plane (see Fig. 2), usually called the ground plane.

Tsai's calibration algorithm has two variants: coplanar and non-coplanar data. For coplanar data, Tsai's algorithm requires the z component of the 3D coordinates to be 0. Since, we are interested on tracking the objects on the ground plane this requirement could be accomplished. Every visual sensor is calibrated using its own calibration template. Therefore, we obtain the image-plane to ground-plane homography which consist on the relation between the 3D (x, y, z) world coordinates of a feature point and the corresponding coordinates (X_f, Y_f) (in pixels).

The constraint in planar motion is used to estimate a homographic correspondence between views and avoid the use of more complex stereo-vision processes. In order to obtain the global coordinates of the local tracks, surveillance-sensor agents compute the inverse transformation of the middle-bottom point of the bounding box of the detected objects in camera coordinates (see Fig. 4).

When spatial alignment is achieved, the next steps carried out by the data fusion process are track-to-track association and vector combination, which are described in the next section.

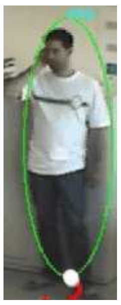

Fig. 4 Local projection on the ground plane (white point) of the middle point of the detected object.

3.3 Data Fusion Problem

In a VSN the information processed by each visual sensor should be integrated or fused accordingly, providing a global picture of the scene which is being monitoring. Thus, data fusion techniques became a key phase of a modern VSN system.

A feasible data fusion implementation in a VSN should take into account several restrictions. Firstly, sending the raw images acquired from each visual sensor to a central processor is not feasible due to the high amount of bandwidth requirements. Secondly, the system should be robust against highly noisy inputs or errors and autonomously correct the errors.

In this paper we only use 2-dimensional tracking positions on the ground plane in common global coordinates; however, different features could be used. Each feature is derived from both a local tracking as well as a projection of the ground-plane positions on the global coordinates.

The local processing task of each visual sensor agent could be formulated as follows. Let a sequence of n frames captured from a visual sensor by an agent S_j which corresponds to n time instances $1 \leq i \leq n$, we can define the set of m tracked features expressed in global coordinates, from each detected object (T_k) in a frame F_i as: $\hat{X}^{S_j}_{T_k}(F_i) = \{\hat{x}^{S_j}_{T_1}(F_i), \hat{x}^{S_j}_{T_2}(F_i)\ , ..., \hat{x}^{S_j}_{T_m}(F_i)\}$.
Notice that the number of detected objects in each frame could not be the same. We can define a track $Tr^{S_j}_{T_k}(l)$ of an object T_k from a sensor S_j as a vector which resumes the observations received until instant l. Since each feature is obtained it is time-stamped, therefore we can align them by time.

In a distributed visual sensor network with overlapped fields of view, more continuity in the tracking process can be provided, as a result of the integration of the information from different visual sensors. However, due to several types of errors (detection errors, illumination changes, etc) a visual sensor could provide inconsistent tracking information [23].

Regarding inconsistent tracking information, let us suppose a set of tracks of the same object T_k from n sensors, $Tr = \{Tr^{S_1}_{T_k}\ , Tr^{S_2}_{T_k}\ , ..., Tr^{S_n}_{T_k}\}$. The fused track $Tr^{F}_{T_k}(l)$ should provide the best estimation until instant l given the visual sensors observations.

So, we argue that when fusing tracking information of the same target from different sensors, some tracking errors could be ruled out, with the aim of obtaining the best estimation. In addition, as a loss of tracks usually occurs, a fusion process is justified in order to avoid tracking discontinuity. This idea is the main objective of the data fusion techniques, which receives noisy redundant information and generates a global otuput.

4 Cooperative Sensor Agents. Architecture Details

In this section, the design and implementation approach of the development of a prototype for Cooperative Sensor Agents (CSA) with the previously defined objectives is described.

4.1 BDI Multi-agent Model

The BDI model provides a way to conceptualize the system and structure its design. It is based on a philosophical model of human practical reasoning, originally developed by M. Bratman [31] which reduces the explanation for complex human behavior to a *motivational stance* [32]. Practical reasoning involves two important processes: (1) deciding the goals to be achieved and (2) how to achieve them. The BDI model is a type of agent architecture containing explicit representations of beliefs, desires and intentions. Beliefs are the information an agent has about its environment, which may be false. Desires are those concepts that the agent would like to see achieved and intentions are those procedures the agent is committed to doing in order to achieve its desires.
Martha Pollack [34] divides the work in BDI agents into three categories:

1. General models of practical reasoning based on BDI concepts.
2. Computational models based on the Intelligent Resource-Bounded Machine (IRMA) architecture [45].
3. The computational model used in the Procedural Reasoning System (PRS) [35] [36]. The PRS system was one of the first implemented systems to be based on a BDI architecture. It was developed by the artificial intelligence center at SRI International and implemented in LISP. It is widely used in practice and also implemented in solutions like JACK [37], JAM [38], Agentis Adaptive Enterprise Suite [39] and more recently in JADEX [40].

Although the BDI model has been well established, issues such as how committed an agent should be to its intentions are still a research topic. It is clear that an agent can abandon its intentions on some occasions, for several reasons: (1) the agent believes that the intention will not be achievable, (2) the intention has been already achieved or (3) the reason for achieving the intention is no longer present.

In CSA architecture, intentions are abandoned as the environment evolves, or if the intention has been already achieved. For example, if the tracked objects disappear from the field of view, the tracking intention will not be achievable.

The proposed CSA architecture in this article is based on the PRS systems computational model [35] [36], specifically on JADEX [40]. It is composed of three different types of agents: (1) Surveillance-Sensor Agents, (2) Fusion Agents and (3) Interface Agents.

4.2 Surveillance-Sensor Agents

Images generated in every surveillance-sensor agent, with huge amounts of useless raw data, must be processed with image and video analysis algorithms running in the local processor associated to the camera. Each camera is attached to a local processor that generates output information according to the representation used (tracks, positions, sizes, etc.) and this information should be coherently adapted and fused to obtain a global view of the situation.

This type of agent tracks all the objects and sends the data to a fusion agent. It acquires environmental information through a camera and performs the local signal processing task. The tasks carried out are: object detection, data association, state estimation and communication with the fusion agent. These tasks are achieved entirely using local data and pixel coordinates and cope with the largest amount of data that can be captured from the video stream The vision algorithms are based on the OpenCv [42] library and coded in C programming language, since the agent code is implemented in Java it accesses them through Java Native Invocation (JNI) [43]. Each surveillance agent can freely communicate with other agents by exchanging ACL messages to make the surveillance process more effective.

The surveillance-sensor agent is a Java thread which runs inside a Jadex agent container. This thread has an execution process which updates the information on beliefs and chooses only those intentions which are needed in order to achieve the desires. Meanwhile, tracking information is obtained through a C++ thread, as the tracking algorithms are developed in C++. These algorithms are built in a dynamic link library (dll) and are accessed in run time through JNI [43].

As explained before, the first step in the surveillance-sensor agent tracking algorithms is the detection of blobs. This is a critical step which directly influences the quality of the global tracking.

In a distributed multi-camera system, each camera has its own field of view. Consistency across multiple camera views (either with shared or disjointed fields of view) can only be maintained when spatial coherence is achieved. As we mentioned previously, this spatial coherence is achieved by performing a ground-plane calibration based on the pinhole camera model. Therefore, each surveillance-sensor agent knows the specific calibration values which it must use.

Each surveillance-sensor agent processes the environmental information gathered from its visual sensor. Previously, the internal clock of the machine where the surveillance-sensor agent runs must be synchronized with the clocks of the other machines. In the system, Network Time Protocol (NTP) [44] is used as an external clock to stabilize the local clock of each machine. Every new detected object or updated object measurement given by the tracking process are marked with a time-stamp as soon as possible. This time-stamp allows time synchronization between all surveillance agents.

The previous processes are embedded into the BDI reasoning cycle. The surveillance-sensor agent's beliefs are continually updated as more environmental data is sensed. Desires are determined by the objectives of the system and the utility (costs and rewards) of satisfying them. The intentions of a surveillance sensor agent determine the course of action that is to be undertaken.

An agent's *beliefs* correspond to the information the agent acquires from the environment and the other agents. It represents the knowledge of the state of the world. Surveillance-sensor agents have the following *beliefs*:

1. Environment Knowledge: knowledge about the objects detected in their field of view. This knowledge consist of the position of each detected object at each time frame in the agent's field of view. This information is stored internally in the agent's belief base.
2. Environment Updating Frequency (ms): an internal parameter which specifies the frequency in milliseconds for update the detected objects in the environment. The agent must have the necessary balance between the computational effort expended in obtaining the visual information and the execution of other activities. This balance is achieved in the system by using an updating frequency belief value.
3. Communication Frequency (ms): this type of agent sends information to the fusion agent periodically. This parameter is established in the system by using a specific belief. Of course, there must be a balance in this parameter to avoid network congestion and ensure communication.
4. Fusion Agent: each surveillance-sensor agent knows its respective fusion agent's name and address (FIPA agent name).
5. Foreground Algorithm: each agent knows the foreground algorithm used to detect moving objects. This algorithm must provide a list of blobs which are found in a frame as well as information about the position and size of each blob. It is possible to use different foreground algorithms in each agent.
6. Tracking Algorithm: each agent knows the tracking algorithm used to tracking the detected objects. Different agents could use different tracking algorithms, as for example, a kalman filter, an extended kalman filter, an unscented kalman filter, a particle filter, etc.
7. Camera Type: the knowledge about the type of camera, or video file which is being used for obtaining the sequence of images.

Desires capture the motivation of the agents. A desire represents the state of affairs that the agent would like to bring about. The surveillance-sensor agents have the following *desires*:

1. Tracking: this desire regards the tracking intentions which are performed continuously.
2. Looking: the looking desire allows the agent to observe the current tracks in the environment.
3. Communication: This type of agent has communication desires. In this proposed framework we communicate for each detected object an associated track vector of features and an error covariance matrix. Therefore, the granularity of the exchanged information is small, compared to the communication of higher level information such as for example trajectory trends. A constant rate of communication feature information, instead of trends, provides more realism when trying to achieve real time performance.

Intentions are the basic steps chosen by the agent to achieve its *Desires* and represent the desires an agent has committed to achieve. *Intentions* constrain the reasoning an agent is required to perform in order to select the action that has to be performed. Surveillance-sensor agents have the following *Intentions*:

1. Tracking: Each surveillance-sensor agent S_i acquires images $I(x, y)$ at a certain frame rate, V_i. The internal tracking process provides for each object X_{T_j}, an associated track vector of features $\hat{X}_{T_j}^{S_i}[n]$, containing the numeric description of their

features and state their location, speed, dimensions, etc. as well as the associated error covariance matrix, $\hat{P}_{T_j}^{S_i}[n]$.
2. To study the Environment: This intention observes the current objects in the environment at time t.
3. New Track Information: The intention of communicating the information about new tracks in the environment to their respective fusion agent using ACL messages.
4. Update Track Information: All the surveillance-sensor agents can communicate information about the new track features to their respective fusion agent using ACL messages.
5. Delete Track Information: This intention allows the surveillance-sensor agent to communicate information about a object that has disappeared using ACL messages.

The previous intentions are the basic steps that allow the architecture to obtain continuity in the tracking along the field of view of the cameras involved in the distributed network.

4.3 Fusion Agents

The fusion agent is a key component of the VSN architecture, as we said before, data fusion techniques provide two main benefits: (1) a continuity in the tracking process and (2) to improve the tracking quality by discarding data of non reliable sensors. The fusion process is carried inside this specific type of agent which receives the tracking information of each surveillance agent. As other BDI agents, the fusion agent has specific beliefs, desires and intentions.
The fusion agent *beliefs* are as follows:

1. Knowledge of surveillance-sensor agents: the fusion agent knows which surveillance sensor agents it manages, so it can understand its environment.
2. Consistency Belief: indicates when a track must be discarded due to it is inconsistency as regards the other tracks with a time-space analysis. This belief indicates a spatial (in centimeters) and temporal (in milliseconds) threshold for discarding local tracks.
3. Fusion Frequency (ms): a value in milliseconds which indicates the frequency on performing the data fusion. On every $Fusion\ Frequency$ the agent checks the data input queues of each surveillance-sensor agent and using the information received it performs the data fusion.

The *desires* of the fusion agent are defined as follows:

1. Fuse Data: the fusion agent has the desire to fuse the data received from several surveillance-sensor agents.
2. Receive Data: the fusion agent has the desire to receive ACL messages from other agents of the system.
3. Inform Data: the fusion agent may inform the interface agent using FIPA ACL messages.

Finally, the fusion agent *intentions* are defined as follows:

1. New Track Information: The intention of receiving information on new tracks, which would be received through an intranet using FIPA ACL messages.

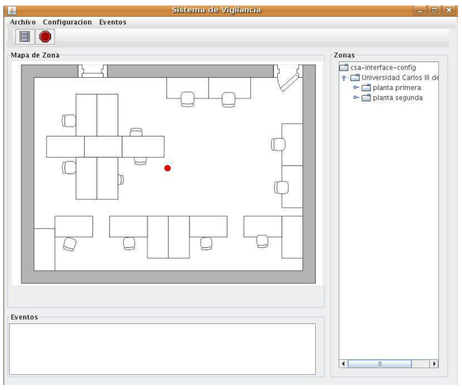

Fig. 5 Interface Agent.

2. Update Track Information: The intention of receiving FIPA ACL messages which contain new information about detected objects.
3. Delete Track Information: Receive information in FIPA ACL messages about deleted tracks.
4. Fuse Data: The fusion agent performs a technique to fuse the information received from the surveillance-sensor agents, which is received in FIPA ACL messages and is time stamped. The algorithm uses the most likely inference from each surveillance-sensor agent.
5. Send Information: The intention of sending the fused data to the interface agent.

The fuse data intention allows the fusion agent to improve the quality tracking by means of fusion techniques. The other intentions allow the fusion agent to obtain continuity in the tracking along the field of view of the cameras involved in the distributed network.

The algorithm used for the fusion process is described on detail in [23] and [24]. The algorithm is based on checking the consistency of the tracks and then applying a combination of each visual sensor values weighted according to its covariance. This method provides smooth movements and discards measurements which are affected by occlusions.

4.4 Interface Agents

This agent receives the fused data and shows it to the final user, which is also the user interface of the surveillance application. In this graphical user interface (Fig. 5) the fused tracking information is shown. The different areas of the building under surveillance are specified in a xml configuration file, where each area has an image or a bi-dimensional map which shows the current state of the zone under surveillance. At this stage, the application shows the information in a two dimensional view as a point in a map. However, further improvements of the interface agent includes a 3D visual interface.

5 Experiments

In this section we evaluate the suitability of the system in two different scenarios. The first one, is given by an indoor tracking in our laboratory using a recorded video file with 3 different cameras. The second one, is performed using the apidis dataset [46]. The apidis dataset is composed by 1500 frames of a basketball game acquired by seven 2-Mpixels cameras around and on top of a basket ball court.

In the first scenario we focus on evaluating the tracking continuity of the target obtained using the fused values and the mean absolute error of the fused values against manually anotated ground-truth values. In the second one, we evaluate the performance of the system when more visual sensors are used. Also in this scenario, since the monitored area is bigger, an evaluation of the calibration quality was performed. The obtained values provide an idea of the quality of the information provided by each visual sensor when the transformation to global coordinates on the ground plane are applied.

To evaluate the calibration quality we applied the most common accuracy evaluation methods:

1. Error of distorted pixel coordinates (E_d): is calculated by computing the difference between estimated pixel coordinates $(\hat{x}_{pi}, \hat{y}_{pi})$ obtained from the measured world coordinates by the camera model with lens distortions and the observed pixel coordinates (which are usually obtained manually) (x_{pi}, y_{pi}):

$$E_d = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(\hat{x}_{pi} - x_{pi})^2 + (\hat{y}_{pi} - y_{pi})^2} \tag{1}$$

   where n is the number of input points to the calibration algorithm.
2. Error of undistorted pixel coordinates (E_u): is calculated similarly than E_d but without using lens distortion in the estimated pixel coordinates and by removing lens distortion from observed pixel coordinates:

$$E_u = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(\hat{x}_{upi} - x_{upi})^2 + (\hat{y}_{upi} - y_{upi})^2} \tag{2}$$

3. The object space error or the distance with respect to the optical ray (E_o): is computed between 3D points in camera coordinates (X_{ci}, Y_{ci}, Z_{ci}) and the optical rays back-projected from the undistorted image points on the image plane (x_{ui}, y_{ui}):

$$E_o = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(X_{ci} - x_{ui}\cdot t)^2 + (Y_{ci} - y_{ui}\cdot t)^2 + (Z_{ci} - t)^2}, \tag{3}$$

   $t = (X_{ci}x_{ui} + Y_{ci}y_{ui} + Z_{ci}/(x_{ui}^2 + y^2ui + 1)$

4. Normalized calibration error (NCE): it is obtained by normalizing the discrepancy between estimated and observed 3D points with respect to the area each pixel covers at a given distance from the camera, as follows

$$NCE = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{(\hat{X}_{ci} - X_{ci})^2 + (\hat{Y}_{ci} - Y_{ci})^2}{Z_{ci}^2(\alpha^{-2} + \beta^{-2})/12}\right)^{1/2} \tag{4}$$

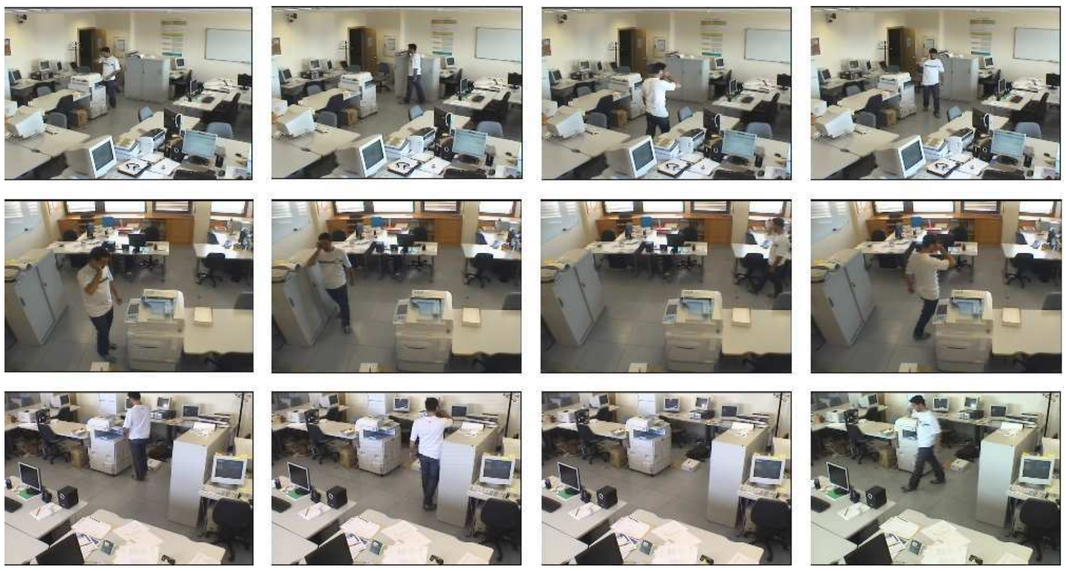

Fig. 6 Example of input images (frames 75, 150, 225 and 300) for camera 1, 2 and 3 of the first scenario. From top to bottom, camera 1, camera 2 and camera 3 images are shown. In the first row we can see the input images of frame 75, the second row shows the images of frame 150, the third one shows the images of the frame 225 and the last one show the images from frame 300.

The first three accuracy measurements (E_d, E_u, E_o) are sensitive to digital image resolution, camera field of view and object to camera distance. The NCE metric proposed by Weng et. al. [47] is insensitive to digital image resolution and provide less biased accuracy results.

To evaluate the global accuracy, the fused results are compared against the ground-truth values on the ground plane. The ground-truth values were obtained in the local camera plane and then transformed to the global coordinates using the obtained calibration projection.

5.1 First Scenario: An Indoor Tracking in Our Laboratory

An indoor evaluation of the proposed architecture was undertaken in our laboratory. Three Sony EVI-100 Pan-Tilt-Zoom cameras were connected to a matrox morphis frame grabber. Despite the pan and tilt capabilities of the cameras, in this experiment they were used as static cameras. Fig. 7 illustrates the camera layout in this experimental environment. The positions of the three cameras and the calibration origin (0,0) can be observed in the axis reference. Each PTZ camera is controlled by a surveillance-sensor agent which runs on a dedicated personal computer. There is also a fusion agent which receives each surveillance-sensor agent's tracking information. A record of 300 frames (with a resolution of 768x576 pixels) from one person randomly moving (in a zone of 660cm x 880cm) is performed.

In Fig. 6 an example of the input images from frames 75, 150, 225 and 300 are shown. In this experimental indoor scenario many occlusions occur (mainly caused by

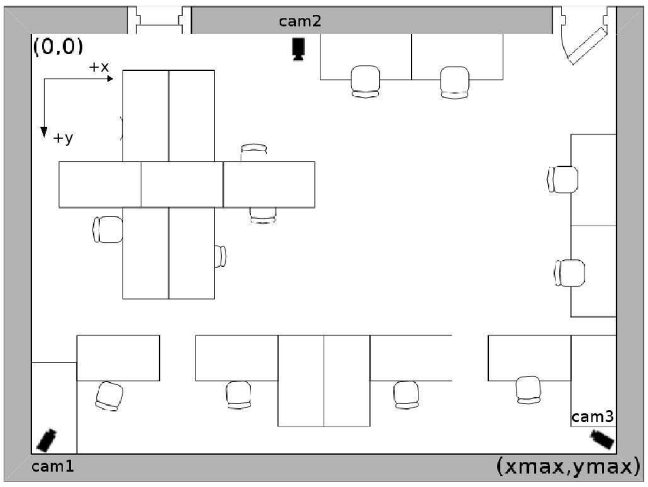

Fig. 7 Camera layout of the first scenario environment (660x880 centimeters).

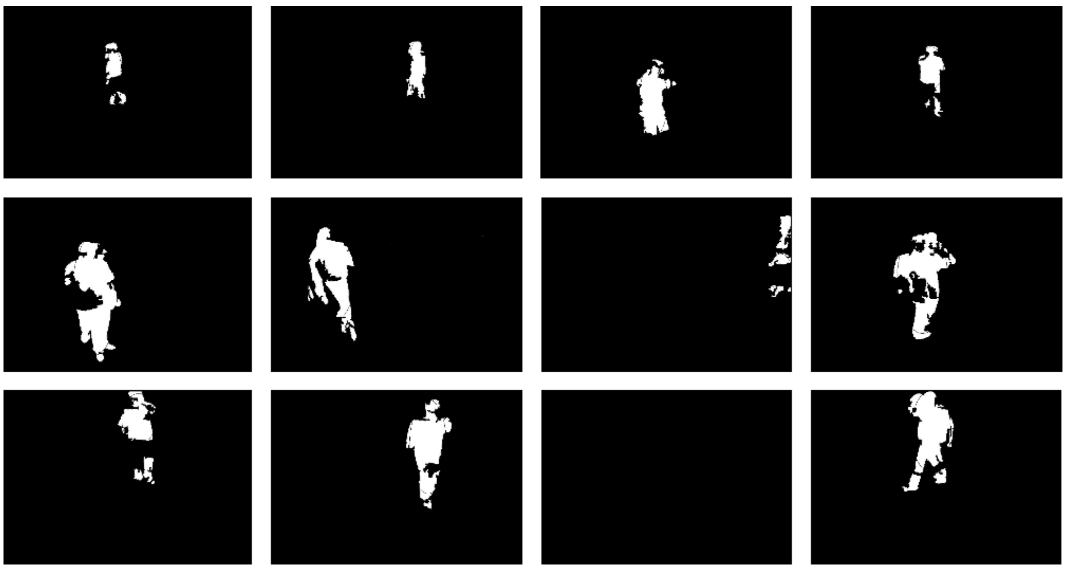

Fig. 8 Example of foreground images (frames 75, 150, 225 and 300) for camera 1, 2 and 3 of the first scenario. From top to bottom, camera 1, camera 2 and camera 3 are shown. In the first row we can see the input images of frame 75, the second row shows the images of frame 150, the third one shows the images of frame 225 and the last one shows the images from frame 300.

the tables and the printer machine) which affect the local tracking of each surveillance-sensor agent.

The image sequences of each camera were analyzed by a surveillance-sensor agent. The specific configuration was adjusted for the surveillance-sensor agent and the fusion agent beliefs. Inside each surveillance-sensor agent, the perception process is triggered every 10 milliseconds. Regarding the fusion agent, the fusion process was activated every 10 milliseconds, with a spatial difference set to 140 centimeters and the temporal difference set to 20 milliseconds. This means that the data fusion only fuse tracks with less time-stamp difference than 20 milliseconds and less location difference than 140 centimeters. For each detected object the coordinates of the bounding box centroid was obtained. This position was projected on the ground plane, and then the information in terms of global coordinates (after the application of the calibration transformation) was sent to the fusion agent. Because of the initialization of the tracking algorithms, there were some initial frames where the person was not detected. For example, camera 1, detected the person in frame 25. As a loss of tracks usually occurs, a fusion process between the overlapped cameras is justified in order to avoid a tracking discontinuity.

For illustrative purposes we present the output of the phase of blob detection (foreground images) for the input images 75, 150, 225 and 300 in Fig. 8.

In Fig. 9 the tracking information obtained by the three surveillance-sensor agents at frames 75, 150, 225 and 300 are shown. As we said before, the information sent to the fusion agent were the calibrated (X_g, Y_g) positions of the projection of the blob centroid onto the ground plane. Positions are expressed in two dimensional coordinates, being X_g the position of the horizontal tracking projection on the ground plane and Y_g

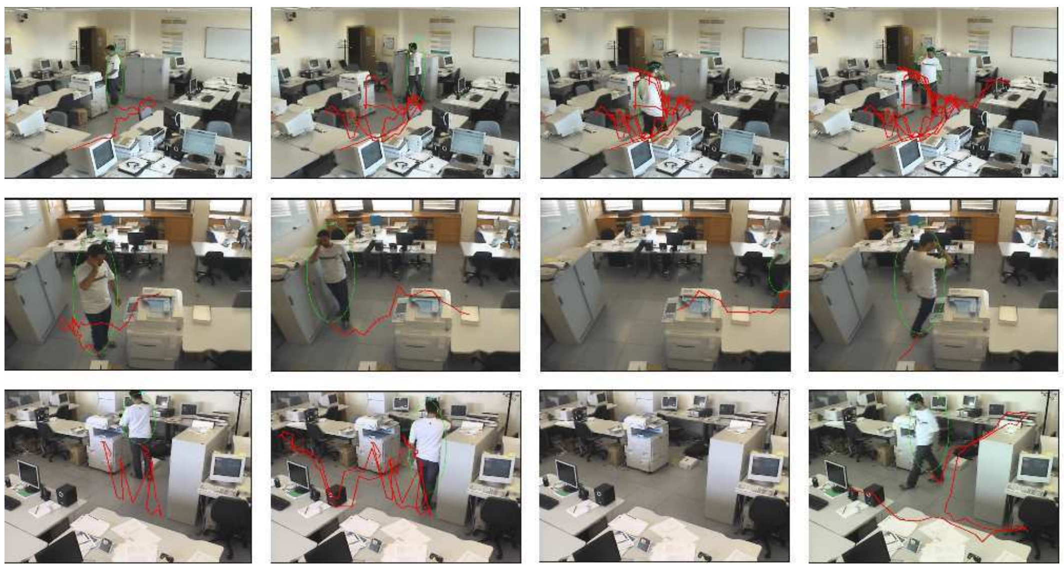

Fig. 9 Example of tracking images (frames 75, 150, 225 and 300) for camera 1, 2 and 3 of the first scenario. From top to bottom, camera 1, camera 2 and camera 3 are shown. The first row are the input images of frame 75, the second row shows the images of frame 150, the third one shows the images in frame 225 and the last one shows the images from frame 300.

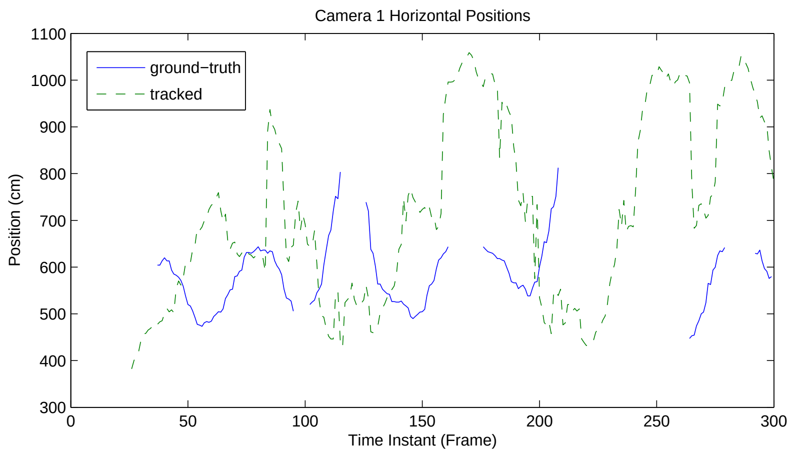

Fig. 10 Camera 1, horizontal ground-truth positions vs horizontal tracked positions.

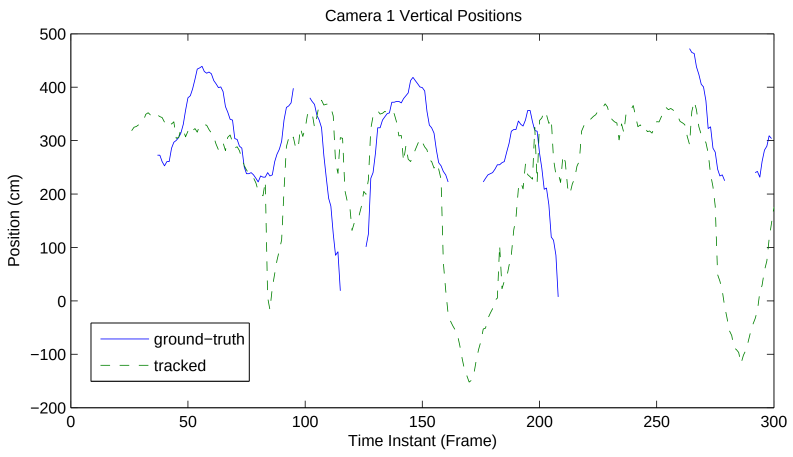

Fig. 11 Camera 1, vertical ground-truth positions vs vertical tracked positions.

the vertical coordinate of the tracking projection on the ground plane (see Fig.7). Each position is compared against manually annotated ground-truth values. In the process of obtaining the ground-truth values, the (X_f, Y_f) local coordinates of the person's feet in those images that are completely visible were annotated and then projected onto the global coordinates. Therefore the ground-truth values are expressed in global or real world ground-plane coordinates using the same transformation values as in the experiments.

In order to evaluate the accuracy of the system the tracking values were compared against the manually annotated ground-truth values. The metric used to evaluate the tracking accuracy is the mean absolute error of the tracked positions against the ground truth values. Tracking positions obtained from each surveillance-sensor agent compared to the ground-truth values are shown in Fig. 10 and Fig. 11 for camera 1, Fig. 12 and

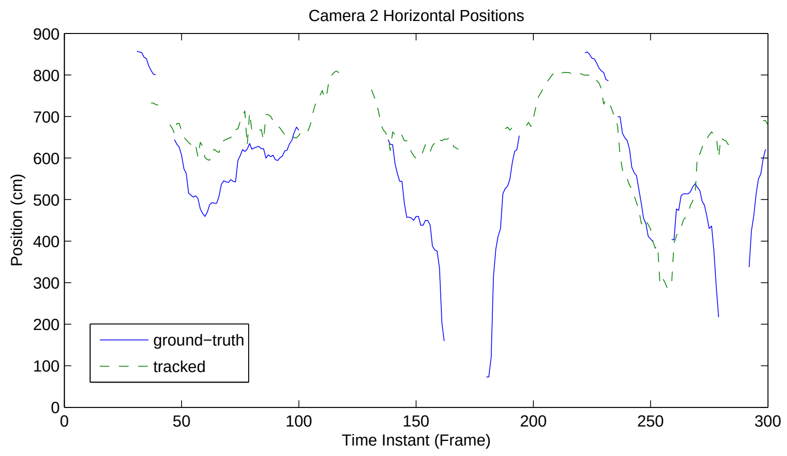

Fig. 12 Camera 2, horizontal ground-truth positions vs horizontal tracked positions.

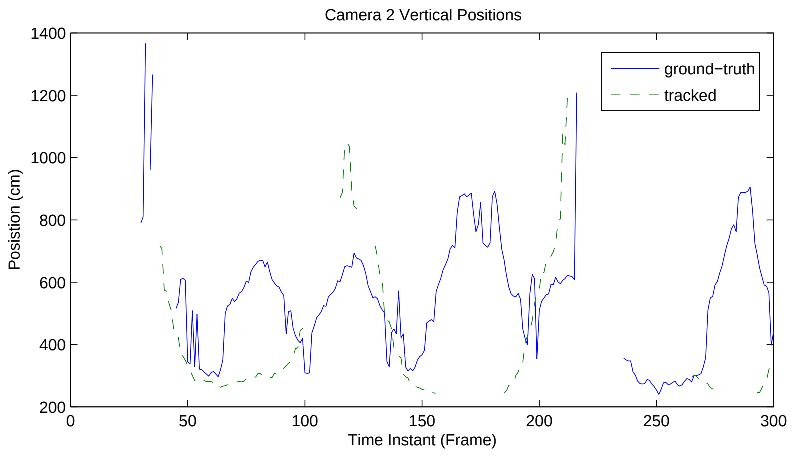

Fig. 13 Camera 2, vertical ground-truth positions vs vertical tracked positions.

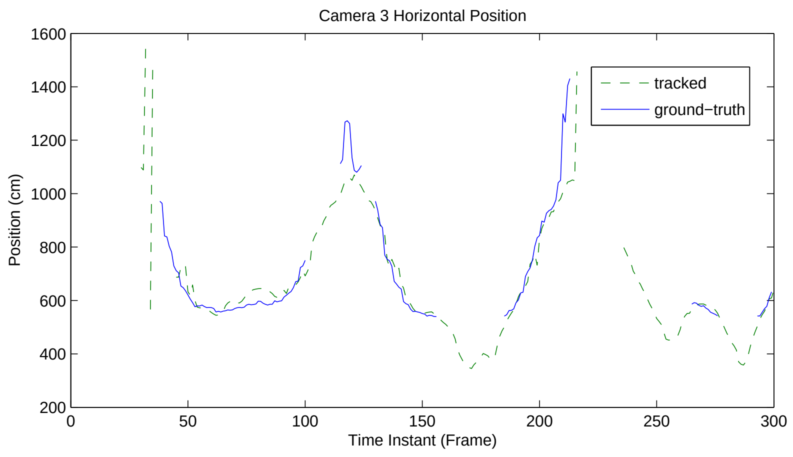

Fig. 14 Camera 3, horizontal ground-truth positions vs horizontal tracked positions.

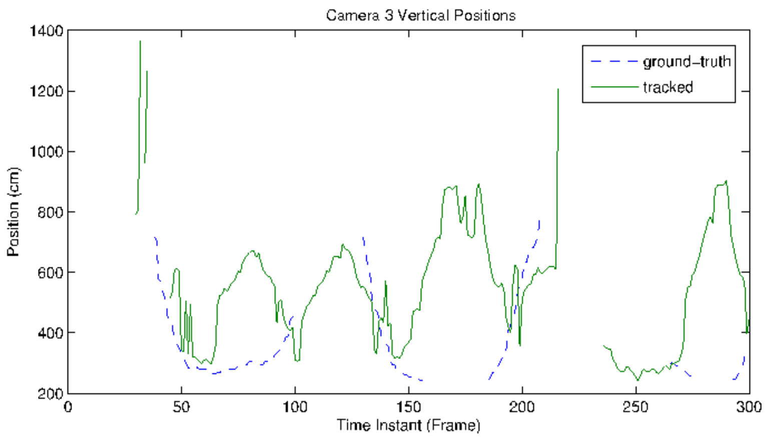

Fig. 15 Camera 3, vertical ground-truth positions vs vertical tracked positions.

Table 1 Mean absolute error between ground truth and tracked positions (centimeters).

	Camera 1	Camera 2	Camera 3	Fusion Agent
Horizontal axes (X)	580.94	100.19	93.2	94.1
Vertical axes (Y)	111.2	140.9	208.23	122.3

Fig. 13 for camera 2, Fig. 14 and Fig. 15 for camera 3. The output fused positions compared with the ground-truth are shown in Fig. 19 and Fig. 20. Finally, the data fused values compared to each surveillance-sensor agent value are shown in Fig. 21 and Fig. 22.

In table 1, we show the mean absolute error of the tracked positions against the ground-truth values for each camera on the horizontal (X) and vertical (Y) axes and the mean absolute error of the fused values against the ground-truth values. As we can observe, the fused values provide a better result than that of each of the sensors taken separately (except for camera 3 horizontal axes). Therefore, the fused values practically achieve the best sensor accuracy for each coordinate. Most of the tracking errors are produced by foreground/background segmentation errors. By segmentation errors we mean the errors generated in those frames where the lower part of the blob is not correctly detected. Therefore when the calibration projection is applied the obtained values are incoherent, because only the ground plane is calibrated. This is the main reason of the good horizontal values of the camera 3, since this camera have a good view of the ground plane when the person is moving (see Fig. 6), because, without any object hiding a person, the foreground/background segmentation relies in a good blob detection stage (see Fig. 8).

5.2 Second Scenario: An Indoor Basketball Match (Apidis Dataset)

In this second scenario, results with the apidis dataset [46] using the previous VSN architecture, are presented. Apidis dataset is composed of a basketball game acquired by seven 2-Megapixels cameras around and on top of a basket ball court. The images of all cameras were captured by a unique server, therefore acquired frames are pseudo synchronized in time. Apidis dataset provides two different video files, one in the native format (1600x1200 image size) and other one pseudo synchronized (800x600 image size). For these experiments, the pseudo synchronized video files of all the cameras except the top view ones (camera 3 and camera 5), recorded at 25 frames per second in 800x600 MPEG-4 resolution, was chosen. A screen-shot of the 5 different views of the apidis dataset and the image of the ground-plane used for calibration purposes is shown in the Fig. 16, the image shown the bounding-box of one specific player. In the next section, the registration phase performed on the apidis dataset is explained.

5.2.1 Evaluation Of The Registration Error

Registration phase receives for each camera a set of points which relate the global 3D (x, y, z) point with the image point (x_{pi}, y_{pi}) in pixel coordinates. The objective is to

Table 2 Estimated values of the intrinsic parameters in the apidis dataset.

camera	**f** (mm)	**kappa1** ($1/mm^2$)	**Cx** (pixels)	**Cy** (pixels)	**Sx**
camera 1	19.403642	9.913269e-04	407.30926	274.804934	1.0
camera 2	19.199602	8.873476e-04	473.95128	204.872783	1.0
camera 4	26.300296	-4.107186e-05	457.79424	-179.912847	1.0
camera 6	20.569473	7.014055e-04	395.10578	264.560713	1.0
camera 7	33.042031	3.644292e-04	182.35262	393.347814	1.0

Table 3 Estimated values of the extrinsic parameters in the apidis dataset.

camera	**Rx** (deg)	**Ry** (deg)	**Rz** (deg)	**Tx** (mm)	**Ty** (mm)	**Tz** (mm)
camera 1	-69.769	21.125	4.219	-3347.818	-1802.199	29096.422
camera 2	-48.16	-69.298	-42.391	-9505.713	2237.553	9324.672
camera 4	-105.49	-46.033	13.605	-10639.491	7367.315	18124.522
camera 6	-70.695	-20.103	-2.691	-22388.609	4.145	20168.512
camera 7	-62.928	23.431	5.752	1646.596	-4522.187	27700.724

obtain the intrinsic and extrinsic parameters for each camera. In this case, we employed the provided basket court image which contain the global position of 69 points on the ground plane (establishing z=0). The known global points in world coordinates are manually matched with the points in image coordinates of each camera. For these experiments, the top view cameras (camera 3 and camera 5) with the fish eye lenses are rule out, therefore we have 5 different views of the scene. We performed a Tsai [25] camera model calibration and the obtained intrinsic parameters are shown in table 2 while the extrinsic ones are shown in table 3. For evaluate the calibration quality we applied the most common accuracy evaluation methods previously described.

The first three accuracy measurements (E_d, E_u, E_o) are shown in table 4 and the obtained NCE values are shown in table 5. Results show that camera 2 and camera 4 have the worst camera calibration. If we have a look at Fig. 16, we notice that camera 2 and 4 (in fig. 16 the second and the third image from left to right) have a field of view with high depth information, which makes more difficult to obtain an accurate calibration which is coherent with the obtained values of calibration accuracy.

5.2.2 Evaluation of Accuracy

The dataset provides an xml file with a bounding box in pixel coordinates for each detected player in each camera view. We employed the provided bounding box values as the output information of each surveillance-sensor agent. So, this means that the image processing tasks are not testing under these experiments, instead the bounding box information of each player in pixel camera coordinates was used. Since each player bounding box, provided in the dataset, also gives the frame number information, the time-space alignment could be easily performed.

At each frame instant the surveillance-sensor agents obtain the projection on the ground plane of the bounding box (see Fig 4), performs the tracking using a kalman filter and then applied the calibration transformation in order to obtain the position in

Table 4 Calibration accuracy measurements of the apidis dataset obtained values.

measurement	**camera1**	**camera2**	**camera4**	**camera6**	**camera7**
E_d mean (pix)	1.188333	3.419681	3.022088	2.453974	1.980312
E_d stddev (pix)	0.587034	6.210263	3.150429	4.308640	2.133353
E_d max (pix)	2.692261	48.36222	18.59863	26.32054	11.40111
E_d sse (pix^2)	64.65491	2826.343	714.287	891.1312	215.7424
E_u mean (pix)	1.237540	3.562748	2.985860	2.606630	2.034450
E_u stddev (pix)	0.606474	6.279122	3.099667	4.752018	2.131429
E_u max (pix)	2.757642	48.88498	18.285212	28.87088	11.40313
E_u sse (pix^2)	69.90688	2931.443	694.277255	1064.337	221.1883
E_o mean (mm)	30.492083	88.80409	54.638430	58.28041	28.5351
E_o stddev (mm)	15.837112	225.7637	60.26738	100.9322	27.6851
E_o max (mm)	63.263249	1741.838	324.40383	616.1429	149.05
E_o sse (mm^2)	43430.692	3303790.5	247833.45	492417.7	40332.18

Table 5 Normalized Calibration Error (NCE) of the apidis dataset obtained values.

	camera1	**camera2**	**camera4**	**camera6**	**camera7**
NCE	3.031341	8.726915	7.313833	6.384913	4.983363

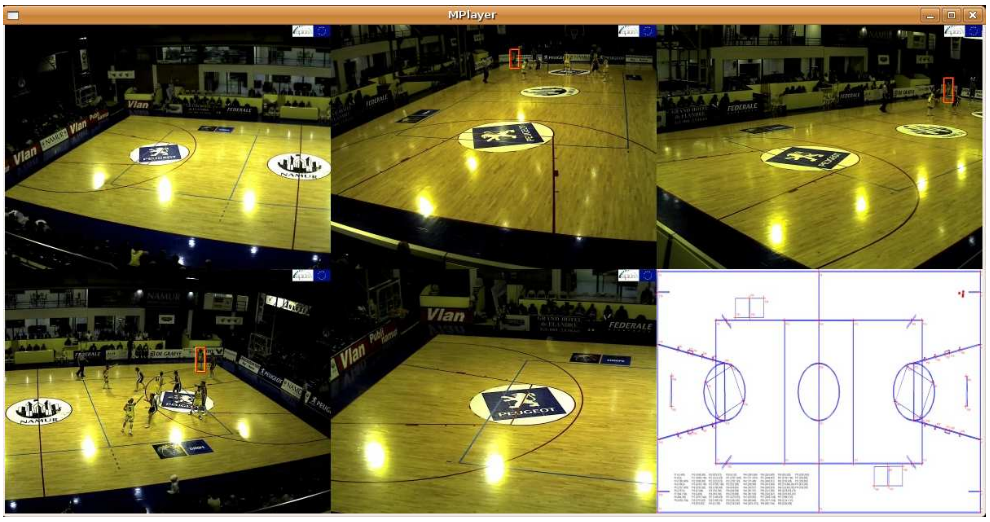

Fig. 16 A screen-shot of the 5 different views of the apidis dataset and the ground-plane used to plot the fused results.

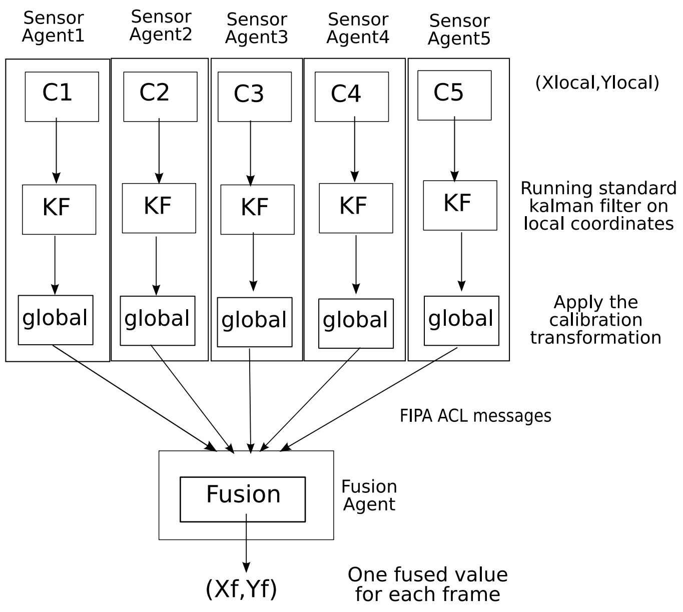

Fig. 17 The process followed to obtain the ground-truth values for testing the robustness of the system against errors in one visual sensor.

global coordinates. This information with the covariance matrix is send to the fusion agent every frame. At the fusion agent, received messages are processed and information is fused using the technique proposed in [23]. Figure 17 show the process followed to obtain the fused values which are later used as the ground-truth information for the experiments.

The objective of the performed experiments was to verify the improvement of the system when more cameras were employed and also testing the robustness of the system. The 5 different cameras were divided into three sets: (1) camera 1, camera 2 and camera 4; (2) camera 1, camera 2, camera 4 and camera 6; (2) camera 1, camera 2, camera 4, camera 6 and camera 7 (see Fig 18).

For simulating sensor and image processing problems in order to test the robustness of the system, experiments with two different pseudo-randomly errors: one between random errors equally distributed in the range of (1,500) millimeters and another one in the range of (1,1500) millimeters; these random errors modify the tracking information provided by camera 1 on global coordinates, after the calibration was performed. The figure 18 shown the followed schema to obtain the fused values affected by these errors.

For each player and each set of cameras the mean absolute error (in millimeters) and the root mean square error (in millimeters) of the fused positions with induced errors against the fused positions without induced errors are obtained (ground-truth). The obtained results are shown in table 6 and table 7 for the first interval and in table 8 and table 9 for the second interval; it is clearly observed in these tables how the tracking errors are reduced when the number of cameras are increased. This results shown the robustness and how the system behaves when the numbers of visual sensors are increasing.

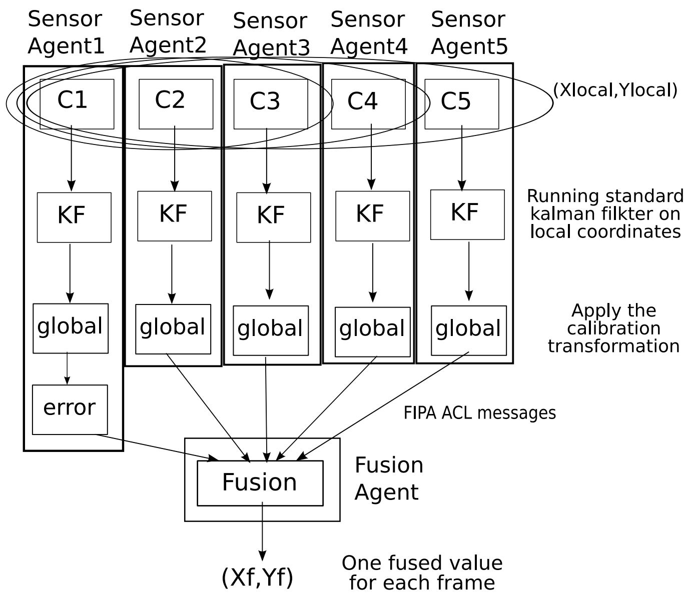

Fig. 18 The process followed to obtain the error values for testing the robustness of the system and the improvement when more visual sensors are used.

Table 6 Root Mean Square Error (RMSE) in millimeters of ground-truth fused values against values with a random error of (1, 500) millimeters in camera 1.

player-team	3-cameras-x	3-cameras-y	4-cameras-x	4-cameras-y	5-cameras-x	5-cameras-y
p10-A	448.58	225.94	344.10	221.84	336.65	218.28
p14-A	275.05	157.54	196.49	152.30	189.75	144.81
p04-A	365.23	174.46	351.25	180.83	344.33	173.31
p05-A	290.29	243.68	249.46	234.84	235.70	216.33
p09-A	309.03	164.08	244.00	159.38	235.43	147.91
p15-B	422.37	187.25	313.60	185.29	295.92	173.72
p06-B	350.50	199.73	296.40	191.60	284.86	182.56
p04-B	344.03	144.69	329.23	165.37	319.62	162.53
p07-B	282.22	164.66	223.42	157.69	214.39	156.09
p09-B	298.62	191.51	280.64	193.72	276.27	190.03

Table 7 Mean Absolute Error (MAE) in millimeters of ground-truth fused values against values with a random error of (1, 500) millimeters in camera 1.

player-team	3-cameras-x	3-cameras-y	4-cameras-x	4-cameras-y	5-cameras-x	5-cameras-y
p10-A	298.80	168.66	228.86	168.54	220.21	164.45
p14-A	195.22	120.74	149.95	117.92	143.72	112.36
p04-A	209.22	135.21	202.04	134.42	191.48	125.78
p05-A	219.94	174.48	192.13	167.95	183.05	154.71
p09-A	208.93	128.18	181.60	125.77	175.06	115.27
p15-B	298.42	137.27	224.22	139.44	214.29	127.70
p06-B	257.56	154.48	227.01	150.72	217.46	143.99
p04-B	231.11	108.56	201.75	114.51	190.36	111.04
p07-B	200.29	125.92	165.89	119.55	159.97	116.94
p09-B	213.05	141.19	191.06	143.24	187.08	139.86

Table 8 Root Mean Square Error (RMSE) in millimeters of ground-truth fused values against values with a random error of (1, 1500) millimeters in camera 1.

player-team	3-cameras-x	3-cameras-y	4-cameras-x	4-cameras-y	5-cameras-x	5-cameras-y
p10-A	498.89	307.81	409.53	301.11	377.49	272.81
p14-A	353.13	269.82	292.96	256.79	261.74	220.25
p04-A	425.88	279.34	416.71	284.34	381.44	238.30
p05-A	367.33	350.43	341.34	339.69	286.83	278.57
p09-A	371.71	268.86	319.48	262.99	284.15	216.04
p15-B	487.02	306.95	399.41	302.99	353.06	251.32
p06-B	444.75	315.39	399.54	314.05	337.66	240.68
p04-B	401.00	255.01	392.17	260.77	357.36	227.44
p07-B	348.42	260.88	303.32	244.69	270.89	220.18
p09-B	357.30	272.89	337.85	265.72	321.10	236.60

Table 9 Mean Absolute Error (MAE) in millimeters of ground-truth fused values against values with a random error of (1, 1500) millimeters in camera 1.

player-team	3-cameras-x	3-cameras-y	4-cameras-x	4-cameras-y	5-cameras-x	5-cameras-y
p10-A	366.97	231.36	300.56	227.35	267.34	207.97
p14-A	278.81	206.01	229.23	195.29	205.03	169.57
p04-A	293.33	226.75	289.55	226.24	251.06	187.06
p05-A	285.49	256.12	260.66	248.40	228.52	211.35
p09-A	276.20	211.45	246.25	203.72	219.80	171.77
p15-B	369.31	213.08	297.47	218.13	267.08	185.07
p06-B	331.71	235.66	300.83	227.18	265.96	189.58
p04-B	297.58	186.15	267.94	186.05	236.99	160.06
p07-B	264.52	195.22	230.06	180.20	208.60	164.10
p09-B	261.00	196.76	236.12	190.71	220.63	173.18

6 Conclusions and Future Works

In this article, a visual sensor network based on the BDI Multi-agent architecture is presented. Using the proposed Multi-agent architecture, we have focused on how to fuse a set of tracks from different agents which belong to the same object. This fused information was used for two different purposes: (1) to obtain a continuity in the tracking along the field of view of the cameras involved in the distributed network, (2) to fuse only the information which provides an accurate tracking, by discarding those visual sensors which have tracking errors.

The results of the first objective are shown experimentally, as the fused agent receives the tracking information of all the other agents involved in the surveillance process. In order to achieve the second objective at every fuse instant the fusion agent selects the more coherent tracking information which have a small variance. This assumption is made taking into account that a person does not make a large movement in a few milliseconds.

The details of the implementation of the proposed architecture were stated. In order to evaluate them, experiments in two different scenarios were performed: (1) an indoor experiment with three surveillance-sensor agents and one fusion agent and (2) using the basketball data provided by the apidis dataset. The results of the first scenario show an improvement in the tracking continuity along the overlapped field of view. The errors obtained in the final tracking positions compared to the average ground-truth positions was 94,19 centimeters in the horizontal plane and 122,03 centimeters in the vertical plane. Results of the second scenario show how the system behaves when the number of the visual sensors increase and how random errors are reduced.

In addition to these results, the use of a multi-agent architecture for visual sensor tasks gives several advantages with respect to traditional systems. It allows us to easily scale the system and to inter-operate with third party developments.

In future studies we are interested on how the system behaves in response to changes in some of the parameter values. Furthermore, issues such as time-delay, communication/sensor noise and more specific models of uncertainty will also be taken into account.

Acknowledgements The authors would like to thank the anonymous reviewers for their valuable comments that help to improve the manuscript.

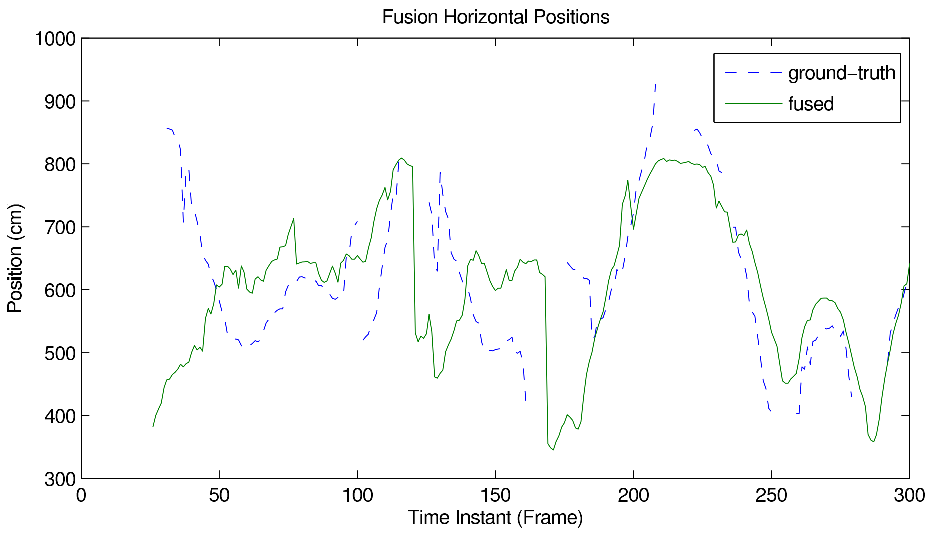

Fig. 19 Data fused, horizontal ground-truth positions vs horizontal fused positions.

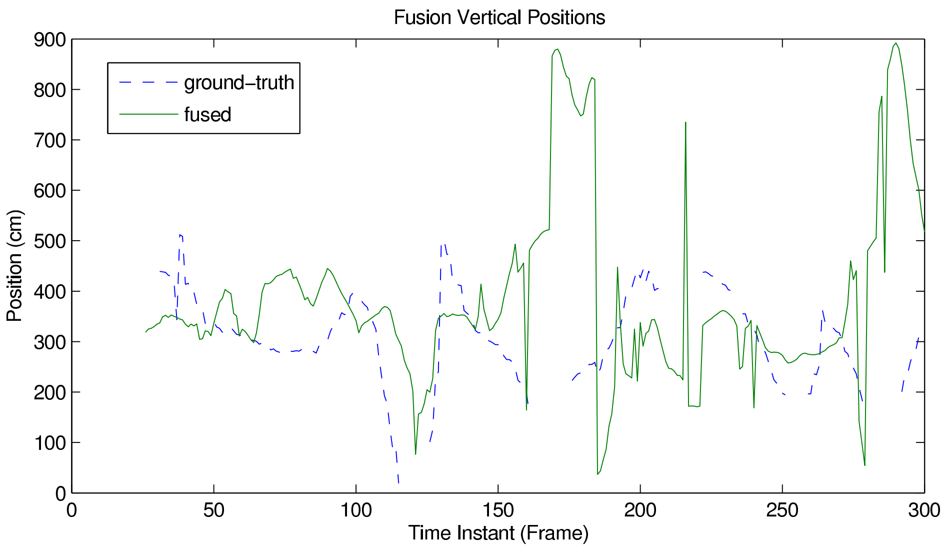

Fig. 20 Data fused, vertical ground-truth positions vs vertical fused positions.

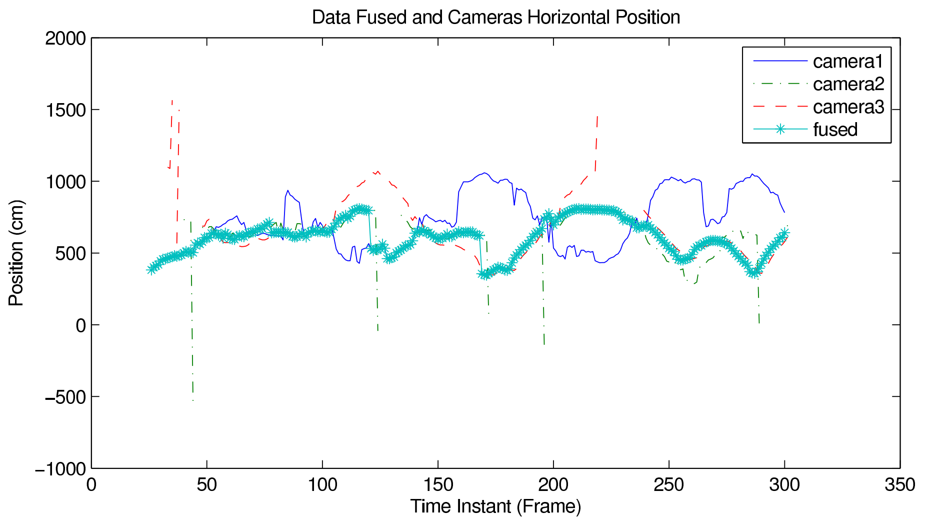

Fig. 21 Data fused, horizontal camera values positions vs horizontal fused positions.

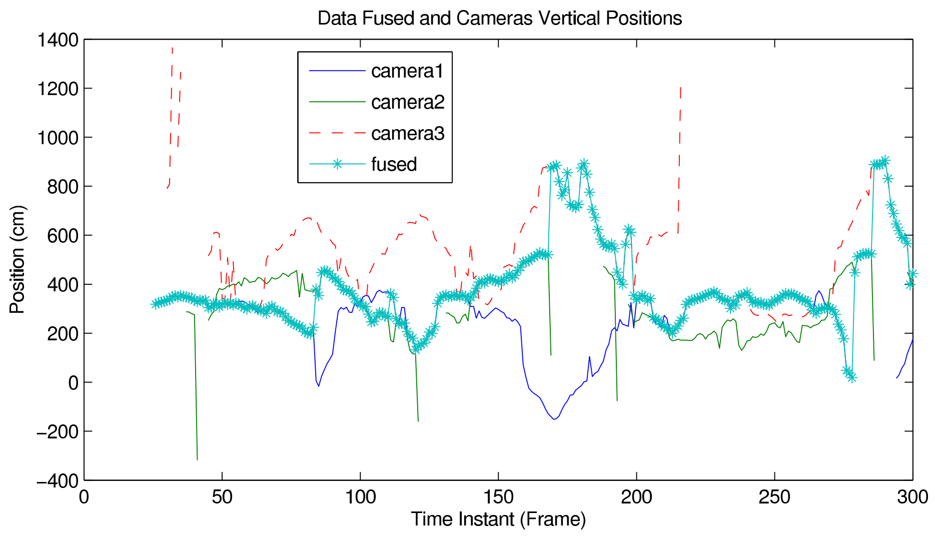

Fig. 22 Data fused, vertical camera values positions vs vertical fused positions.

References

1. T. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Computer Vision and Image Understanding*, vol. 104, no. 2-3, pp. 90–126, 2006.
2. M. A. Patricio and J. Carbo and O. Perez and J. Garcia and J. M. Molina, "Multi-Agent Framework in Visual Sensor Networks", *EURASIP Journal on Advances in Signal Processing*, 2007.
3. M. Wooldridge and N. Jennings, "Intelligent agents: Theory and practice," *The knowledge Engineering Review*, 1995.
4. V. Berge-Cherfaoui and B. Vachon, "A multi-agent approach of the multi-sensor fusion," *Fifth International Conference on Robots in Unstructured Environments*, pp. 1264-1259, 1991.
5. "Multimedia Object Descriptions Extraction from Surveillance Types". Accessed 18 September 2009. http://www.tele.ucl.ac.be/PROJECTS/MODEST/index.html
6. FIPA, "Fipa communicative act library specification," Accessed 18 September 2009. http://www.fipa.org/specs/fipa00037/SC00037J.pdf
7. L. Botelho, R. Lopes, M. Sequeira, P. Almeida, and S. Martins, "Inter-agent communication in a FIPA compliant intelligent distributed dynamic-information system," *Proc. 5th International Conference on Information Systems Analysis and Synthesis (ISAS99)*, 1999.
8. T. Graf and A. Knoll, "A Multi-Agent System Architecture for Distributed Computer Vision," *International Journal on Artificial Intelligence Tools*, vol. 9, no. 2, pp. 305–319, 2000.
9. P. Remagnino, A. Shihab, and G. Jones, "Distributed intelligence for multi-camera visual surveillance," *Pattern Recognition*, vol. 37, no. 4, pp. 675–689, 2004.
10. B. Abreu, L. Botelho, A. Cavallaro, D. Douxchamps, T. Ebrahimi, P. Figueiredo, B. Macq, B. Mory, L. Nunes, J. Orri, *et al.*, "Video-based multi-agent traffic surveillance system," *Intelligent Vehicles Symposium. Proceedings of the IEEE*, pp. 457–462, 2000.
11. J. Orwell, S. Massey, P. Remagnino, D. Greenhill, and G. A. Jones, "A multi-agent framework for visual surveillance," in *ICIAP '99: Proceedings of the 10th International Conference on Image Analysis and Processing*, (Washington, DC, USA), p. 1104, IEEE Computer Society, 1999.
12. T. Matsuyama, S. Hiura, T. Wada, K. Murase, and A. Toshioka, "Dynamic memory: architecture for real time integration of visual perception, camera action, and network communication," *IEEE Conference on Computer Vision and Pattern Recognition. Proceedings.*, vol. 2, 2000.
13. N. Ukita, "Real-time cooperative multi-target tracking by dense communication among active vision agents," *IEEE/WIC/ACM International Conference on Intelligent Agent Technology,*, pp. 664–671, 2005.
14. A. Bakhtari, M. Mackay and B. Benhabib, "Active-Vision for the Autonomous Surveillance of Dynamic, Multi-Object Environments", *Journal of Intelligent and Robotic Systems*, vol. 54, no. 4, pp. 567–593, 2009.
15. L. Marchesotti, S. Piva, and C. Regazzoni, "An agent-based approach for tracking people in indoor complex environments," *Image Analysis and Processing, 2003. Proceedings. 12th International Conference on*, pp. 99–102, 2003.
16. N. T. Nguyen, S. Venkatesh, G. West, and H. H. Bui, "Multiple camera coordination in a surveillance system," *Acta Automatica Sinica*, vol. 29, no. 3, pp. 408–421, 2003.
17. J. Garcia, J.M. Molina, J.A. Besada, and J.I. Portillo, "A MultiTarget Tracking Video System based on Fuzzy and Neuro-fuzzy Techniques," *EUROASIP Journal on Applied Signal Processing. Special Issue on Advances in intelligent vision systems: Methods and Applications*, no. 14, 2005.
18. M.A. Patricio, F. Castanedo, A. Berlanga, O. Perez, J. Garcia and J. M. Molina, "Computational Intelligence in Visual Sensor Networks: Improving Video Processing Systems", *Computational Intelligence in Multimedia Processing: Recent Advances Series: Studies in Computational Intelligence* , Vol. 96, 2008. ISBN: 978-3-540-76826-5.
19. M. A. Patricio, J. Garcia, A. Berlanga and J. M. Molina, "Solving Video-association Problem with Explicit Evaluation of Hypothesis Using EDAs". *Proceedings of the 2008 Congress on Evolutionary Computation CEC2008*, CEC 2008.
20. A.M. Sanchez, M.A. Patricio, J. Garcia and J.M. Molina, "Video tracking improvement using context-based information", *10th International Conference on Information Fusion*, 2007.

21. J. S. Tittle and J. T. Todd, "Perception of three-dimensional structure",*The handbook of brain theory and neural networks*, The handbook of brain theory and neural networks, MIT Press, Cambridge, MA, USA.
22. A. Quevedo and D. Maravall. "Wheelchair Tracking System in Hospital's Corridors by means of Video-surveillance Systems", *Technical report. Artificial Intelligence Department, Polytechnic University of Madrid*, May 2003.
23. F. Castanedo, M.A. Patricio, J. Garcia and J.M. Molina, "Robust data fusion in a visual sensor multi-agent architecture", *10th International Conference on Information Fusion*, 2007.
24. F. Castanedo, J. Garcia, M.A. Patricio and J.M Molina, "Analysis of Distributed Fusion Alternatives in Coordinated Vision Agents",*11th International Conference on Information Fusion*, 2008.
25. R. Tsai, "A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses",*IEEE Journal of Robotics and Automation*, Vol. 3, no. 4, pp. 323–344, 1987.
26. J. Heikkila, "Geometric camera calibration using circular control points," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1066–1077, 2000.
27. Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.
28. V. Kettnaker. and R. Zabih, "Bayesian multi-camera surveillance," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 253–259, 1999.
29. FIPA, ACL "Message Structure Specification". Accessed 18 September 2009. http://www.fipa.org/specs/fipa00061/
30. A. Rao and M. Georgeff, "BDI agents: from theory to practice," in *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95)*, (Cambridge, MA, USA), pp. 312–319, The MIT Press..
31. M. Bratman, *Intentions, Plans and Practical Reasoning.* Cambridge, Massachusetts: Harvard University Press, 1987.
32. D. Dennett, *The Intentional Stance.* Bradford Books, 1987.
33. C. Jo, G. Chen, and J. Choi, "A new approach to the BDI agent-based modeling," *Proceedings of the 2004 ACM symposium on Applied computing*, pp. 1541–1545.
34. M. Georgeff, B. Pell, M. Pollack, M. Tambe, and M. Wooldridge,"The Belief-Desire-Intention Model of Agency," *Proceedings of the 5th International Workshop on Intelligent Agents V: Agent Theories, Architectures, and Languages (ATAL-98)*, vol. 1555, pp. 1–10.
35. M. Georgeff and A. Lansky, "Reactive reasoning and planning," *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pp. 677–682.
36. F. Ingrand, M. Georgeff, and A. Rao, "An architecture for real-time reasoning and system control," *Expert, IEEE [see also IEEE Intelligent Systems and Their Applications]*, vol. 7, no. 6, pp. 34–44, 1992.
37. P. Busetta, R. Ronnquist, A. Hodgson, and A. Lucas, "JACK Intelligent Agents-Components for Intelligent Agents in Java," *AgentLink News Letter*, vol. 2, pp. 2–5, 1999.
38. M. Huber, "JAM: a BDI-theoretic mobile agent architecture," *Proceedings of the third annual conference on Autonomous Agents*, pp. 236–243, 1999.
39. D. Kinny and R. Phillip, "Building Composite Applications with Goal-Directed (TM) Agent Technology," *AgentLink News*, vol. 16, pp. 6–8, 2004.
40. A. Pokahr, L. Braubach, and W. Lamersdorf, "Jadex: A BDI reasoning engine," *Multi-Agent Programming: Languages, Platforms and Applications. Springer, Berlin*, 2005.
41. A. Pokahr, L. Braubach, and W. Lamersdorf, "Jadex: Implementing a bdi infrastructure for jade agents," *Search of Innovation (Special Issue on JADE)*, vol. 3, pp. 76–85, September 2003.
42. OpenCV, Accessed 18 september 2009. http://opencv.willowgarage.com/wiki/
43. JNI, "Java native method invocation specification v1.1," 1997.
44. D. Mills, "Internet time synchronization: the network time protocol," *IEEE Transactions on Communications,*, vol. 39, no. 10, pp. 1482–1493, 1991.
45. M.E. Bratman, D.J. Israel and M.E. Pollack,"Plans and resource-bounded practical reasoning," *Computational intelligence,*, vol. 4, no. 3, pp. 349–355, 1988.
46. Apidis, Accessed 18 september 2009. http://www.apidis.org/Dataset/
47. J. Weng, P. Cohen, and M. Herniou, "Camera calibration with distortion models and accuracy evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 10, pp. 965–980, 1992.