Multi-agent System for Tracking and Classification of Moving Objects

Sergio Sánchez[1], Sara Rodríguez[1], Fernando De la Prieta[1], Juan F. De Paz[1], and Javier Bajo[2]

[1] Computer and Automation Department, University of Salamanca, Plaza de la Merced s/n, 37008, Salamanca, Spain
{sergio_sg,srg,fer,fcofds}@usal.es
[2] Artificial Intelligence Department, Polytechnic University of Madrid, Campus Montegancedo, Boadilla del Monte, Madrid 28660, Spain
jbajo@fi.upm.es

Abstract. In the past, computational barriers have limited the complexity of video and image processing applications but recently, faster computers have enabled researchers to consider more complex algorithms which can deal successfully with vehicle and pedestrian detection technologies. However, much of the work only pays attention to the accuracy of the final results provided by the systems, leaving aside the computational efficiency. Therefore, this paper describes a system using a paradigm of multi-agent system capable of regulating itself dynamically taking into account certain parameters pertaining to detection, tracking and classification, to reduce the computational burden as low as possible at all times without this in any way compromise the reliability of the result.

Keywords: Computer Vision, Agents, Multi-Agent System, Vehicle Detection, Vehicle Counting, Pedestrian Detection, Classifiers, Video-Surveillance.

1 Introduction

Computer Vision is a field of Artificial Intelligence that has captured the attention of many researchers from diverse communities in recent years. This field is intended to achieve a computational model of the human sense of vision in order to produce numerical or symbolic information to enable decision making . For this, the field includes methods for the acquisition, processing, analysis and understanding of real-world images, whether maps dimensional pixels, combining images from stereo to obtain 3D models cameras and even multidimensional data to work with any type dynamic scene. In this sense, intelligent approaches based on multi-agent systems (MAS) combined with information fusion process have been recently emerging [14]. MAS [20] are increasing in importance in the research line of distributed and dynamic intelligent environments. These intelligent systems offer a high-level tool to support a framework for intelligent information fusion and management. There are now many

S. Omatu et al. (eds.), *Distributed Computing & Artificial Intelligence, 12th Int. Conference*, Advances in Intelligent Systems and Computing 373, DOI: 10.1007/978-3-319-19638-1_8

lines within the field of artificial vision in which it is necessary to manage different information of different kind of sensors such as systems video surveillance and monitoring, robotics and autonomous navigation, instrumentation for medicine, security systems, satellites and military applications, image processing, data mining large volumes of images or quality control in manufacturing chain[7][17][14].

For all these reasons, it is possible to say that MAS are an ideal option to create and develop the open and heterogeneous systems such as those normally found in the computer vision and information fusion process. This paper presents an intelligent multi-agent system that incorporates new information fusion techniques to automatically locate objects through devices such as surveillance cameras to facilitate tracking or location in a dynamic environment. This article is structured as follow: the next section describes the general background in vehicle detection and tracking focus on optimizing computational efficiency, information fusion and multi-agent systems. The third section is a description of the components of the proposed system. Finally, the fourth section presents the results and conclusions obtained, measured mainly from the point of view of the computational savings offered by the use of agents.

2 Background

The computational techniques used in this work are described in the following paragraphs, focusing on Pedestrian and Vehicle tracking, Multiagent systems and Computational Efficiency Applied to Computer Vision.

Regarding **Pedestrian and Vehicle Tracking**, in this work, we use Optical flow algorithm [8] to detect coming traffics. Given the two consecutive frames, we find corner points with corner detector [15] and these features are matched by the Optical flow algorithm, which assumes that important features are detected in both frames. We fuse some features into a rectangle if the Euclidian distance of two features (the location of feature and the direction of optical flow) are small. The optical flow algorithms find the correspondence within a reasonable time so that we can use the algorithms for real-time application. However, some of optical flows are generated by some cracks of the road and road signs. Haar-like feature detector [18] is used to detect traffics in the same direction, but the optical flow algorithm is not so appropriate for this goal because they do not have any salient movement in most case. The shapes mostly show the rear side of the car. To detect the rear shape we choose to use Haar-like feature detector because it is fast and efficient.

Much of systems using Artificial Vision only pays attention to the accuracy of the final results provided by the system, leaving aside the **computational efficiency**. Research suggests that the computational efficiency and the accuracy of the system are directly linked, stating that it is impossible to have reliable results in dynamic scenarios if the computational effort of the system is reduced; but not impossible in controlled settings [10]. Finally, these studies end up veering towards parallelization hardware for increased performance. Focusing on our work, there are a large number of studies that make comparisons on specific parts of similar systems, such as comparisons between different methods of "Background subtraction" [11], but never come

to relate it to the scene, no even with the other phases of a typical detection and tracking system. The work presented in this paper will demonstrate that it is possible to have a system of collaborative **agents** that fuse certain information to suit the status of each scene and may be reduced at all times the computational effort to a minimum where accuracy does not decrease, by regulating a number of parameters described in the following sections.

The information obtained from multiple vision sensors needs to be fused because no single sensor can get all the information, and the information from different sensors may be uncertain, inaccurate, or even conflicting [9]. This is the reason why information fusion is a fundamental part of a computer vision management. There is a considerable variety of sensors that can observe user contexts and behaviours and multi-agent architectures that utilize data merging to improve their output and efficiency [7]. The adequacy of **MAS** applied to information fusion and dynamic environments has been deeper discussed in our previous work [13]. Moreover, various works inspire the proposed architecture, among which we mention [4].

On this background, in this work, the above techniques are combined, using a cascade of stages on the geometric characteristics of the objects to detect, track, count and classify the detected features, and the influence of various parameters on the behavior of the system is studied.

3 System Overview

Our goal is to develop a core technology on which to build custom applications that require the use of computer vision for detecting, tracking and counting people and vehicles. To do this, we have set the following technical objectives: (i) Develop an optimal algorithm for detecting, tracking, counting and classification of moving objects, which combine both static and dynamic techniques in image processing; (ii) Develop a graphical user interface for visualizing and modifying some parameters of the image processing phases; (iii) Develop a module for capturing images from IP camera to ensure interoperability with existing video-surveillance systems.

The use of MAS provides a mechanism that allows individual units called agents to perform tasks concurrently. These agents can be software units that undertake simple tasks to reach a common goal of the overall system. This provides a number of important advantages. First, it implies a decentralization of the complete system, so detection calculations tasks, based on a complex Gaussian Filter, could be performed in another computer. It also provides great modularity and scalability, which makes it possible to add new functionality by adding more phases, or combine the input of a few cameras. Finally, the MAS provide a communication system that allows each of the agents to exchange information following the protocol defined by the platform.

There are multiple platforms when making a practical application related to these MAS, such as JADE (Java Agent DEvelopment framework)[2] or PANGEA [20] (*Platform for Automatic Construction of Organizations of Intelligent Agents*). The latter has been used for the deployment and communication of agents that will perform the full functionality needed for the case study.

3.1 Phases of the System

To accomplish the above objectives, it was decided to design a system comprising a number of cascade-connected stages. Each of these phases is in charge of a task, so the output of one stage is the input of the next stage. In addition, each phase is associated with a type of agent that communicates with the agents associated with other phases of the system. The purpose of this communication between agents is dynamically set the appropriate values for a number of parameters, which depend of the state of the scene. This will reduce the computational effort with respect to set all this parameters statically.

The agents involved and therefore the computational details of these phases are described below. As will be shown throughout this article, the average improvement of the computational efficiency at each stage is between 24.90% and 78.34% depending on the three-dimensional structure and congestion level of each particular scene.

Processing Cascade consists of the following phases: stabilization, detection, tracking, classification and group. Below it is explained in details how each of the last four phases work, making an especial emphasis to the improvements made at each stage.

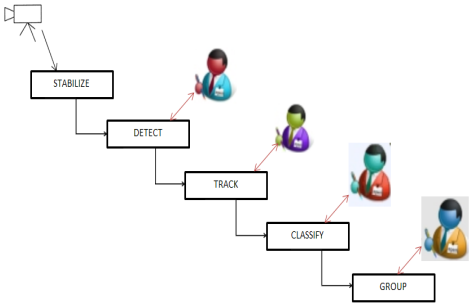

Fig. 1. Block diagram of the Processing Cascade System

Detection Agent

Once the camera has been calibrated and the image stabilized, the next step is the detection of changes in the sequence of frames that would indicate some kind of movement. Because the cameras that provide images are not moving, a background subtraction method for detection motion is used. The operation of this type of methods is based on the error between an estimate of the image without moving objects and the current image. The numerous approaches to this problem differ in the type of background model used and the procedure used to update the model.

There is much research concerning the different background subtraction methods. This researches qualify numerically the behavior and robustness of each of this background subtraction methods according with the type of video input, memory needed or computational effort required.

It has been demonstrated that the static and simple background subtraction methods, as MinMax, whose operation is based on a threshold on a grayscale, offer worse results than any of the dynamic methods [3]. However, more sophisticated methods such as

KDE or dynamic Eigen not always produce more accurate results, especially in noisy images, and due to their consumption of CPU and memory are not suitable for image processing in real time [3], so we have chosen the Gaussian Mixture Model (GMM) method as the basis of this detection phase [16]. This filtering process consists of enclosing the area of mobile contours considered valid between a maximum and a minimum values to ensure that the areas in which motion has been detected by size may correspond with pedestrians or vehicles. If not, the frame will not go into other phases of the Processing Cascade and will not be unnecessarily processed if there were no eligible contours into it. The Detector Agent has the self-learning ability needed to establish the maximum and minimum values using information that other agents have processed and reported from other stages in the cascade as follows:

- When the system starts operating, because of the absence of historical information, all the frames in which motion has been detected will advance to the next stage of the Processing Cascade.
- When a target gets through all stages of the Processing Cascade, the Group Agent communicates with the Detector Agent, sending him information on the average area of this category.
- The higher the number of targets that went through all stages, then we will have more information to affirm with greater certainty that the area of potential targets in that category will be closest to the average area of all the targets that are already part of the historical of that category on this particular scene.

This allows the system to enclose the valid area more precisely when the Group Agent has stored more data. It means that we are providing the system with self-learning, since the longer the system is running, the greater efficiency we are having for this stage. According to the values shown in the training video datasets, the system has been modeled so that a single element in the historical of this category, the maximum possible area is twice that middle area and as you increase the number of historical data, the maximum allowable area will get closer to average historical area of the category.

N_C being the number of objectives that have been classified in a category C, we have the following máximum and mínimum área values for category C:

$$MaxArea_C = AverageArea_C * 1.1 + (AverageArea_C)/(N_C) \quad (1)$$

$$MinArea_C = AverageArea_C * 0.9 - (AverageArea_C)/(N_C) \quad (2)$$

The 10 percent additional margin of these two values is due to the Area of a track depends on its distance from the camera, which will vary over time. It has been demonstrated with the training datasets that this variation of the track relative size will never exceed 10 percent of its original value, so that adds and subtracts this value to the maximum area and minimum area respectively. It is considered that an objective O is eligible to be part of a category C if and only if the following condition is met, discarding the frame if it contains no eligible contours:

$$MinArea_C \leq Area_O \leq MaxArea_C \quad (3)$$

The following figure shows graphically the learning capacity of the system. As the size of the historical increases, the eligible area is bounded more severely and the efficiency of the system increases:

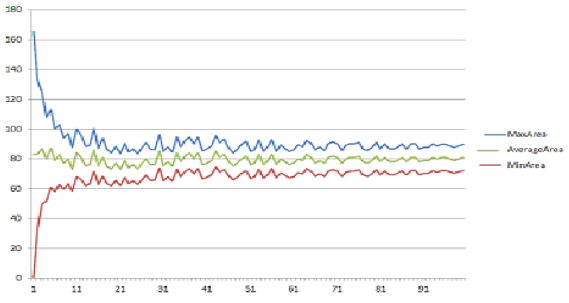

Fig. 2. Eligible area of Category C is bounded more severely while the amount of historical data increases

To demonstrate the computational savings offered by this improvement to the overall process, which will continue to provide the same result, in the following table it is shown the number of discarded frames, which ultimately depends on the degree of influx of pedestrian and vehicles passing in front of the camera:

Table 1. Improving the computational efficiency due to the filtering by area

Training Data: 12,000 Frames for each situation		
Congestion charge	**Frames that pass the filter**	**Frames that do not pass the filter**
High	10903	90.85%
Medium	4772	39.76%
Low	692	5.76%

Tracker Agent

Once detected every eligible target, at this stage the system will obtain their position over time through small changes in the image, which are tracked by the technique known as Optical Flow, described in the paper [1]. For this technique to work, it must be defined a zone of influence around the target with a suitable dimension so that there must be a real correspondence between small changes in the frames and the movement of the real targets. Thus if the new motion is detected at a distance greater than the limit of the former movement, the system will track it as another target, rather than treating it as a movement of the previous target. In short, this technique can only be applied reliably if the input video has a number of Frames per Second (FPS) enough to make this association in the area of influence.

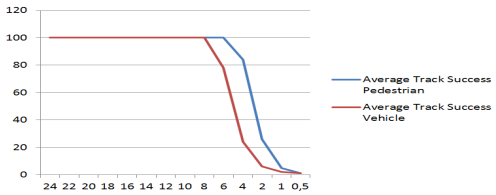

Fig. 3. Average Track Success for pedestrian and vehicles in relation with the number of FPS

The previous graph shows the success rate in relation with the number of FPS for pedestrians and vehicles with a training dataset of 12000 frames. From the above figure can be drawn that the result of the system will be equally reliable if is reduced from 16 FPS, typically offered by the video surveillance cameras [19], up to 8 FPS in the case of vehicles and 6 FPS in the case of pedestrians, because the movement speed is lower. While it is true that setting a static limit of 8 FPS would have a reliable result for both cases, the use of a Tracker Agent to set the FPS rate for each scenario dynamically bring further improvement in performance. The operation of the Tracker Agent is based on reducing the FPS rate to a minimum, so that in each frame the new position of the track approach as much as possible the edge of the area of influence of that track in the previous frame, without never exceed that limit, because it would led to an erroneous result.

The way in which the tracking algorithm is programmed to provide valid results, if the area of a track of a pedestrian is W * H, then its influence area would be a circle whose area would be 0.6*W*H:

$$\text{Influence radius} = \text{Sqrt}((W*H)/(pi)) * 0.6 \quad (4)$$

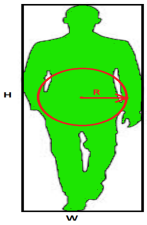

Fig. 4. Influence Radius of a track while using the Optical Flux Method

For a given track, we can immediately calculate the Influence Radius, and compare it with to the amount of movement from the previous frame, which could be defined as:

$$\text{Movement} = \text{sqrt}\ (\Delta x^2 + \Delta y^2) \quad (5)$$

By linking the two measures, the Tracker Agent can modify the FPS Rate that each track needs, in order to minimize the computational effort as follows:

$$FPS_T = \text{Desplazamiento/RadioMovimiento} * FPS_{T-1} \quad (6)$$

This measure is directly proportional to the speed of the different tracks, so the Tracker Agent set for this moment the highest FPS rate of all the active tracks belonging to the scene. Ultimately, as the aim of the system is not seeing a fluid video to the human eye, but it can be processed reliably to extract the desired information. The Tracker Agent allows us to improve the efficiency of the system as shown in the following table:

Table 2. Improved system efficiency due to the dynamic regulation of the number of FPS

Training Data 12,000 Frames in the initial video			
Congestion charge	**Average FPS**	**Standard FPS**	**Percentage of FPS needed.**
High	7.58	16	47.37%
Medium	6.31	16	39.43%
Low	4.42	16	27.62%

Classifier Agent

At this point of the cascade stages, you must perform a classification of the target to determine whether it is a pedestrian, a vehicle or unclassifiable goal. This phase is reached only if there is a possible moving target that has not yet been definitively classified in the category of pedestrian or vehicle at the scene. Because the technique classifiers cascade introduces noise in the form of false positives [12], it is not enough that the target is classified in one of two possible categories in a particular frame, because it could be a false positive.

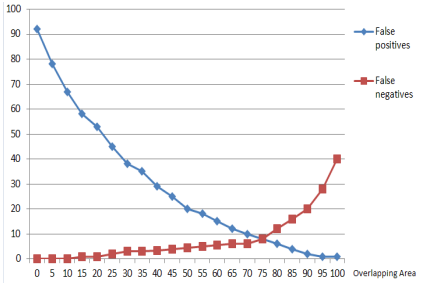

Fig. 5. Point balance between false negatives and false positives

To decide whether a goal has not yet been rated definitely belongs to one of the two categories in a given frame, once the image has been processed by the classifier, the rectangle associated with track and the rectangle in which was detected a person or vehicle must have a certain percentage of their areas overlapped. The ideal percentage overlap was determined by testing with a total of 12000 frames of real video sequences. If the percentage of overlapping is too small, the system will get many false positives, but if that percentage is too large some tracks will not be taken as positive, but they are actually positives.

As seen in the above graph, facing the percentage of overlap in relation to false positives and negatives, the balance point is found in an overlapping area of 70%. Regarding the treatment of false negatives, after a frequency analysis with videos in which appear both pedestrians and vehicles, the following results were obtained:

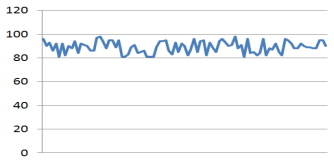

Fig. 6. Percentage of success in the classification of each detected track

According to this analysis, we can ensure that the number of times the track is classified incorrectly will never exceed 20%. Based on this data, the Classifier Agent dynamically set the total number of times each track must be classified into one of two categories to definitely belong to that category. Using this Classifier Agent will stop executing this phase if all the targets of the scene are definitely classified, again causing computational savings. The number of positive enough will be calculated by the Classifier Agent, that communicate with the Group Agent to know at all times the average number of frames that the targets of this category remain on the scene.

As the noise in the classification will never be higher than 20%, a frame definitely belongs to one of the two categories if the number of times it has been classified in that category exceeds the following threshold:

"Classification threshold" = 0.2 * Average frames on stage for that category. (7)

This means that in the best case we would be saving the classification process in 80% of cases, but this would only be true if there were never more than two targets on stage simultaneously. For actual results, in the following table you can see how the computational savings in the classification process is greater with decreasing the degree of congestion.

Table 3. Improved system efficiency due to dynamic threshold for classification

Training Data: 12,000 Frames in the initial video			
Congestion charge	**Frames in which the classifier is used**	**Frames with detected activity**	**Percentage required to use the classifier**
High	10557	10903	96.82%
Medium	3112	4772	65.21%
Low	286	692	41.33%

Group Agent

In the last phase of the Processing Cascade, the Group Agent is responsible for the processing and storage of historical data of the system. The goal is to communicate with other agents who need some historical data, and the creation of social role models for studying the behavior of the targets in each scenario using data mining techniques.

4 Results and conclusions

Below, the proposed improvements at each stage will come together to see what the average computational savings in each case would be through the use of intelligent agents that lead to a non-linear optimized system.

Table 4. Improved system efficiency of each phase

Training Data: 12,000 Frames in the initial video				
Congestion charge	**Computation in tracking phase.**	**Computation in detection phase.**	**Computation in classiffication phase.**	**Percentage relative to unoptimized system**
High	47.37 %	90.85%	96.82%	**78.34 %**
Medium	39.43 %	39.76 %	65.21%	**48.13 %**
Low	27.62 %	5.76 %	41.33%	**24.90 %**

The main conclusion drawn is that the improvement of the system compared to algorithmically correct system in the treatment of the images but without any parametric regulation depends heavily on the "Congestion Charge", although in all cases significant improvements occur with regard to computational effort. We should also note that when cascaded stages, we would be leading to a greater reduction in computational effort because the improvements would chaining.

In short, although the general increase in computer performance allow us to run the system without improvements, with the architecture and the behavior of the agents proposed the system will stop committing inconsistencies as to process frames in which there are no eligible targets by size, try to classify all the frames when all objects have already been definitively classified or process a number of FPS, much larger than needed to obtain a reliable result. With these improvements, it has been possible to run the system satisfactorily in a Single-Board Computer (SBC) with a 700MHz processor and 512Mb of Random Access Memory (RAM), capable of running a Linux distribution.

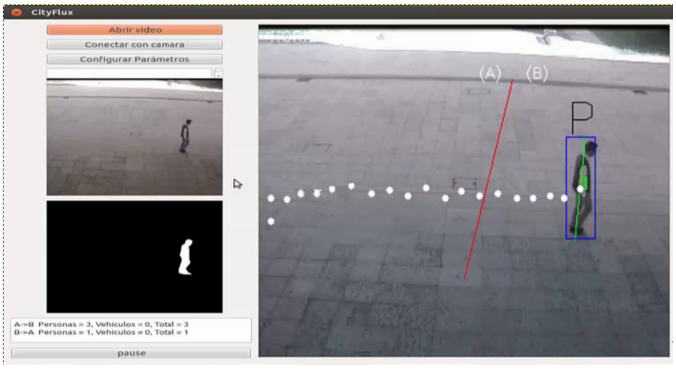

Fig. 7. Screenshot of the developed software running in a SBC

Acknowlegements. This work has been carried out by the project Sociedades Humano-Agente en entornos Cloud Computing (Soha+C). SA213U13. Project co-financed with Junta Castilla y León funds.

References

1. Baker, S., Roth, S., Scharstein, D., Black, M.J., Lewis, J.P., Szeliski, R.: A Database and Evaluation Methodology for Optical Flow. In: IEEE 11th International Conference on Computer Vision, ICCV 2007, pp. 1–8 (2007), doi:10.1109/ICCV.2007.4408903, ISSN 1550-5499
2. Bellifemine, F.L., Poggi, A., Rimassa, G.: Developing multi-agent systems with JADE. In: Castelfranchi, C., Lespérance, Y. (eds.) ATAL 2000. LNCS (LNAI), vol. 1986, pp. 89–103. Springer, Heidelberg (2001)
3. Benezeth, Y., Jodoin, P.-M., Emile, B., Haurent, H., Rosenberg, C.: Comparative Study of Background Subtraction Algorithms. Journal of Electronic Imaging 19 (2012)
4. Boissier, O., Demazeau, Y.A.: An architecture for social and invidivual control and its application to Computer Vision. In: European Workshop on Modeling Autonomous Agents In a Multiagent World, pp. 107–118 (1994)
5. Brooks, R.A.: Intelligence without representation. Artificial Intelligence 47, 139–159 (1991)
6. Liu, Y.H., Wang, S.Z., Du, X.M.: A multi-agent information fusion model for ship collision avoidance. In: Proceedings of the International Conference on Machine Learning and Cybernetics, pp. 6–11 (2008)
7. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proc. of the 7th IJCAI, Vancouver, Canada, pp. 674–679 (1981)
8. Luo, H., Yang, S., Hu, X.: Agent oriented intelligent fault diagnosis system using evidence theory. Expert Systems with Applications 39(3), 2524–2531 (2012)
9. Sundaram, N.: Making computer vision computationally efficient. EECS Department University of California, Berkeley Technical Report No. UCB/EECS-2012-106 (May 11, 2012)
10. Piccardi, M.: Background subtraction techniques: a review. In: 2004 IEEE International Conference on Systems, Man and Cybernetics (2004)
11. Zhu, Q., Avidan, S., Yeh, M.-C., Cheng, K.-T.: Fast Human Detection Using a Cascade of Histograms of Oriented Gradients. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 1491–1498 (2006), doi:10.1109/CVPR.2006.119, ISSN:1063-6919
12. Rodríguez, S., De Paz, Y., Bajo, J., Corchado, J.M.: Social-based planning model for multiagent systems. Expert Systems with Applications 38(10), 13005–13023 (2011)
13. Rodríguez, S., De la Prieta, F., García, E., Zato, C., Bajo, J., Corchado, J.M.: Virtual Organizations in Information Fusion. In: 9th International Conference on Practical Applications of Agents and Multiagent Systems - Special Session on Adaptive Multiagent Systems, Salamanca, Spain, pp. 195–202 (2011)
14. Shi, J., Tomasi, C.: Good features to track. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR 1994), Seattle (June 1994)
15. Stauffer, C., Grimson, W.E.L.: Adaptive background mixture models for real-time tracking. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2 (1999), doi:10.1109/CVPR.1999.784637, ISSN:1063-6919

16. Tapia, D.I., de la Prieta, F., Rodríguez González, S., Bajo, J., Corchado, J.M.: Organizations of Agents in Information Fusion Environments. In: Antunes, L., Pinto, H.S. (eds.) EPIA 2011. LNCS, vol. 7026, pp. 59–70. Springer, Heidelberg (2011)
17. Viola, P., Jones, M., Snow, D.: Detecting pedestrians using patterns of motion and appearance (2003)
18. Wang, X.: Intelligent multi-camera video surveillance: A review. Pattern Recognition Letters 34(1), 3–19 (2013), doi:10.1016/j.patrec.2012.07.005
19. Zato, C., Sanchez, A., Villarrubia, G., Rodriguez, S., Corchado, J.M., Bajo, J.: Platform for building large-scale agent-based systems. In: 2012 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS), pp. 17–18 (May 2012)