Hindawi Publishing Corporation
Advances in Multimedia
Volume 2014, Article ID 962015, 10 pages
http://dx.doi.org/10.1155/2014/962015

Research Article

Execution Behavior Modeling Methodology for Large Scale Surveillance System Design and Evaluation

Jung-Min Oh,[1] Kyung Hoon Kim,[1] Sangjin Hong,[1] and Nammee Moon[2]

[1] *Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794, USA*
[2] *Department of Computer Engineering, Hoseo University, Asan 336-795, Republic of Korea*

Correspondence should be addressed to Nammee Moon; mnm@hoseo.edu

Received 21 May 2014; Revised 30 September 2014; Accepted 1 October 2014; Published 14 October 2014

Academic Editor: Young-Sik Jeong

This paper presents a performance and evaluation environment for complex surveillance system design. The system consists of environment model, execution model, and application and evaluation model. The environment model interprets the script and creates objects in a surveillance environment so that various situations can be evaluated. The execution model modifies generated data with the perspective of each sensor and reflects algorithm execution behavior. The application model allows building large scale collaborative operations. The system behavior is parameterized for simple representations. The feasibility of the proposed method is illustrated through the case studies for improving the prototype surveillance system.

1. Introduction

Many researchers have been interested in surveillance systems with heterogeneous sensors. They usually consist of various sensors to complement each other, such as visual sensors, identification sensors, motion sensors, and acoustic sensors. The visual sensors are used to monitor moving objects and identification sensors are used to identify them [1–4]. The motion sensors or the acoustic sensors are usually used to detect many abnormal behaviors by any movements or sounds in the surveillance region [5, 6]. The association between heterogeneous sensors is also to identify and track an object [7, 8]. Due to the characteristics of a sensor's detection, the performance of the surveillance system is heavily affected by the environmental factors [9, 10].

In order to evaluate surveillance system design and its internal algorithms, simulation based methods have been widely used in the research community [11, 12]. They are used for verifying the adequacy of the surveillance system designed and the visualization and modification of large amounts of data before the costly system is physically installed [13]. Some researchers [14, 15] present a virtual environment to test and compare developed algorithms, but they usually focus on the visual surveillance system only. It is limited to evaluating the most recent surveillance system using heterogeneous sensors [16]. Also, most simulation based approach is performed in limited environment, to highlight the novelty and characteristics of the proposed design [17–19]. Such simulation is inadequate to measure the performance of the system design in a realistic environment. More importantly, the common problem in all the simulation based approaches lies in their complexity [20].

In this paper, we present a modeling methodology for easily and flexibly representing a variety of surveillance applications as well as dynamic environments. The environment model for simulating realistic environments is specified with simplified representation. It covers a surveillance environment, objects with scheduled or random motions, and sensors with dynamic reconfiguration and time-varying coverage. In order to model sensor's processing and collaboration, we provide an execution model. Its key idea is to model the execution behavior of an actual surveillance system in a simplified format instead of an actual implementation. The parameterized models enable us to test various surveillance designs in a short amount of time. All scripts are parameterized in the proposed systems. The proposed modeling allows the designer to evaluate the target system through rapid prototyping.

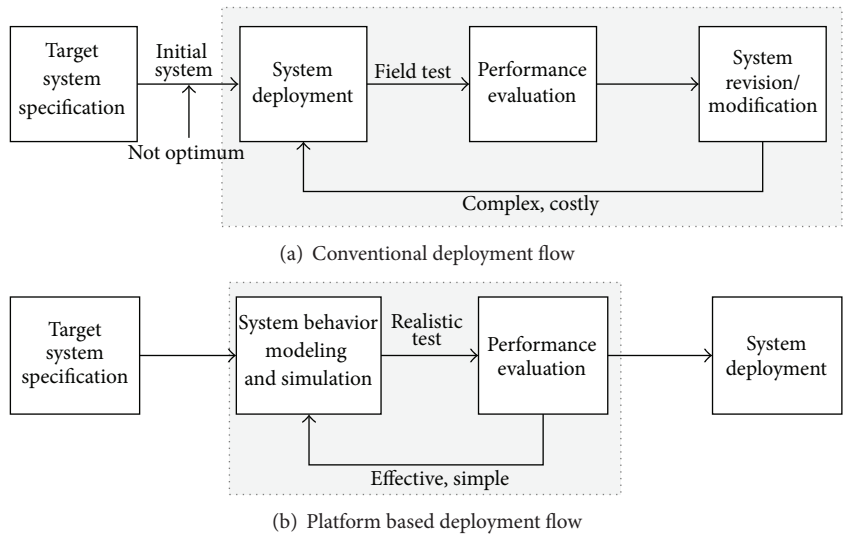

Figure 1: Comparison of two different surveillance system design and deployment methods.

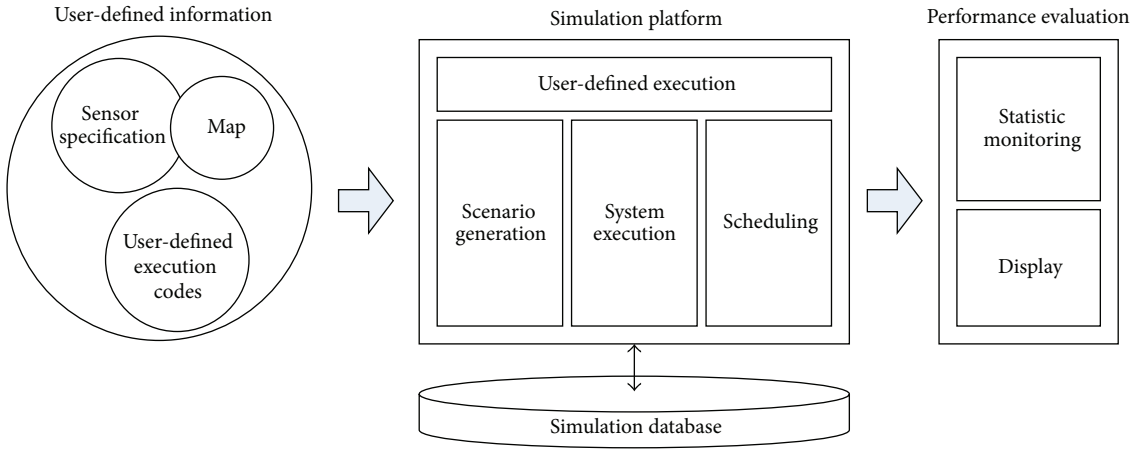

Figure 2: Simulation overview and system platform structure.

2. Motivation and Problem Description

This work is motivated by the necessity of evaluation tool for cost-effective surveillance system design and for easy comparison of diverse surveillance system structures. The effective surveillance deployment flow based on our proposed model is compared to the conventional one in Figure 1. In a conventional deployment flow, it is very costly to design and deploy a large scale surveillance system (i.e., the final target system must be fully implemented in order to evaluate it). As the complexity of surveillance environment actually changes, the deployed system may not operate efficiently or may not guarantee sufficient accuracy. With the proposed deployment flow, we can effectively design and deploy surveillance systems even as the complexity increases. Thus, the proposed deployment method allows evaluating system even before the system implementation.

Figure 2 shows an overview of the proposed platform. When the large scale surveillance system is described, there are three main components in the structure. First, the user-defined information must be provided (as a script file). These include map information, type of sensors used, and the object trajectory information. The user-defined information defines a specific target system. Once the information is defined, the simulation platform executes the behavior of the target applications. Note that when we describe the behavior, the actual implementations are not necessary. Since we can obtain a complete system simulation information, visualization and monitoring are possible for average long-term performance analysis such as an effect of algorithmic changes or environment changes.

The overall platform operation requires an interaction between the scenario generation and behavior execution as illustrated in Figure 3. When the user-defined information is provided, the platform scheduler generates all object motion information. This information is then used by the subsequent execution model. In order to model the dynamic behavior of the sensors, the interface mechanism is provided such that

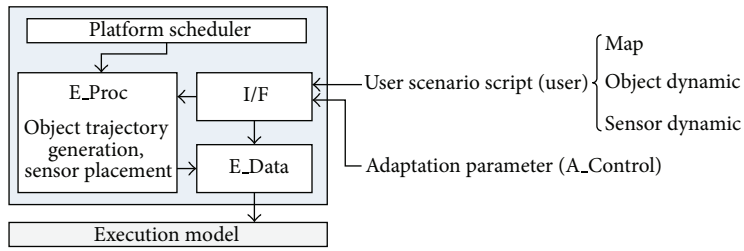

Figure 3: The scenario generator generates an application-specific data and the data is processed by the execution model.

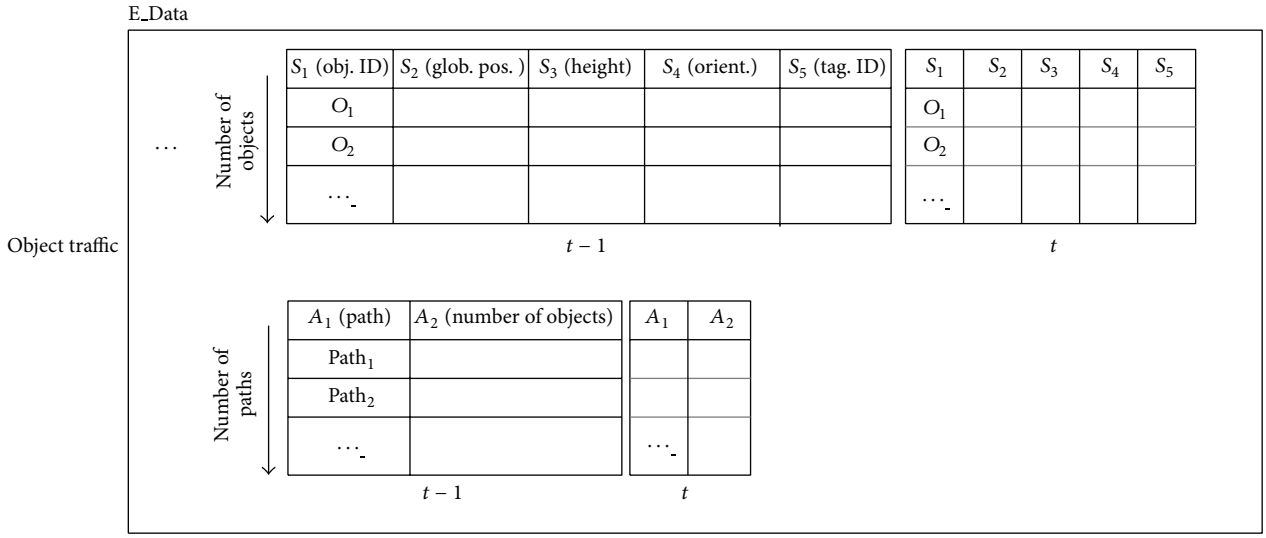

Figure 4: Environment data format for object management.

the sensor characters may be changed by the application. In the execution model, the topological representation of heterogeneous sensors and the collaborative signal processing among them are easy to be modeled as a parameterized script.

3. Scenario Modeling

3.1. Object Motion Representation and Generation. We model an object as a 3-dimensional rectangular parallelepiped since the detection of a visual sensor, such as camera, depends on an object's height and orientation with respect to the camera. For each object, we provide a global identification which is uniquely used in an enterprise environment. In addition to the globally defined identification within the platform, there is additional information we need to maintain such as RFID tag identification. It is necessary for evaluating the performance characteristics by using object identification as ground-truth reference. Figure 4 illustrates the data structure maintained by the scenario generator about the object. The data structure describes the complete information about the object trajectories. In order to represent a trajectory path of each object, we define the spatial and temporal parameters about an object's movement. For the spatial parameters, we maintain a departure place and an arrival place for each object. However, for random generation of an object, this spatial information is randomly assigned by the simulation. In the environment, the map is represented by a graph with nodes and edges (i.e., the edges for a path in the map and the nodes for intersections and/or key places such as doors or elevators). In order to represent a path as staying in a specific position, we specify an identical node index. For the path's temporal parameters, we use a departure time range and an arrival time range. The start and end time are specified in a time range. In order to represent a path departing or arriving at a specific time, we specify the time identically to the start and end time. By setting a time range with the time gap, we represent a path departing or arriving at random time in the time gap. In order to represent a path without specific time requirements, we use "random" in a time range. In order to represent a path with velocity variation, we use the field about an object's velocity.

3.2. Sensor Characterization and Dynamics. In the environment model, any type of sensors can be incorporated including visual sensors, proximity sensors, and identification sensors. These sensors are specified before the system evaluation. There are two types of sensors: real sensors and pseudo sensors. The real sensors are the ones that are physically available such as cameras and RFID readers. The pseudo sensors are virtual sensors such as homographic lines to detect the events. Typically, these sensors can be dynamically configured during the operations in real time.

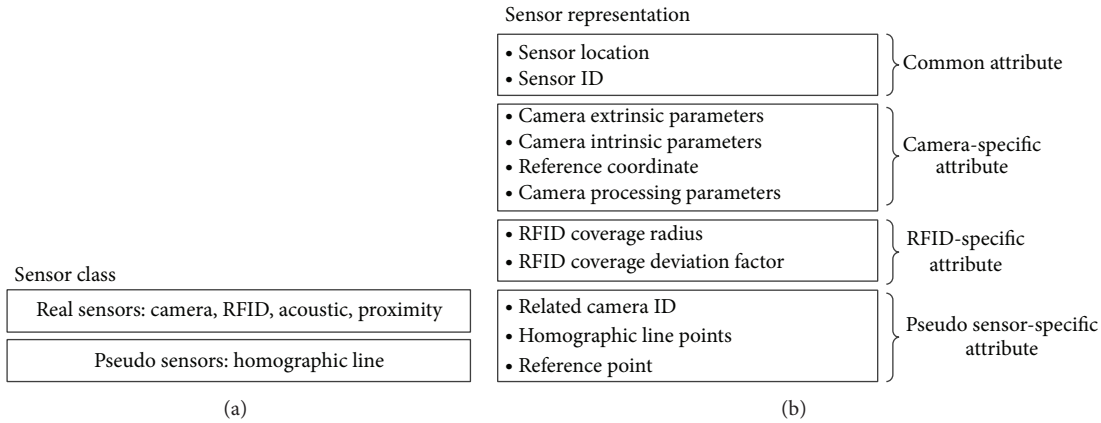

Figure 5: Sensor representation script format.

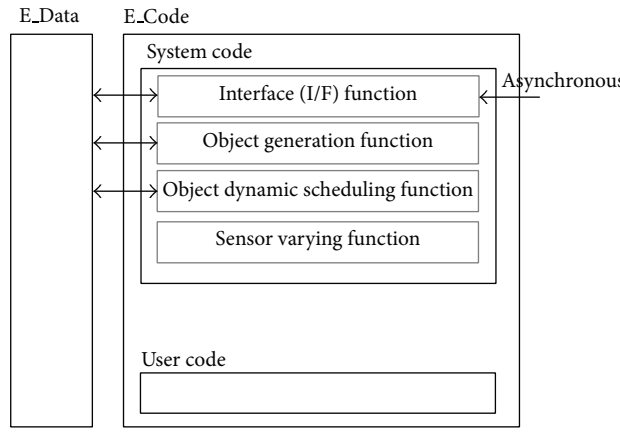

Figure 6: Interaction between environment and execution models.

Figure 5 illustrates the script of sensor characteristics provided as user-defined information. A visual sensor, such as a static camera, is configured with several parameters. The extrinsic information includes panning and zooming capability. The intrinsic parameters include distortion factors and so forth. In addition, various parameters such as resolution and sampling rate are also specified. In order to model the camera's image capture, we will use a mathematical camera model in modifier.

However, an active camera is dynamically configured by a surveillance application after it is initially configured. Since the camera's movement includes mechanical movement, the movement time is greater than the computation time, such that it cannot be ignored for the realistic modeling. Thus, we introduce an interface function. It uses camera index, panning, tilting, and zooming value from the application model as input. By comparing the value of corresponding parameters from the script to those input values, it obtains the delta values. By using the translation speed and zooming speed of the camera index in the script, it computes the time taken by the camera movement. An RFID sensor is one of the widely used identification sensors. We model the RFID sensor with a several parameters. We model RFID coverage as a circle centering the RFID location. Some application may change the coverage dynamically by reconfiguring RFID's radiation power in the middle of its operation. The interface function we discussed is simply extended for dynamic RFID coverage change by replacing RFID coverage radius in the script with the new value from the application model.

However, in the actual deployment of the RFID sensors, the RFID coverage has characteristics of its boundary fluctuating in time. Because of this characteristic, even though an object continues to be in a location within the coverage, it is not always detected. To model the fluctuation, we use RFID coverage deviation factor as an attribute. Thus, we introduce the time-varying fluctuation model to model the realistic coverage fluctuation. The model uses RFID coverage radius as an input and outputs a set of grids which an RFID reader actually reaches. It produces the output every simulation time. Such operation is illustrated in Figure 6.

4. Execution Behavior Modeling Platform

4.1. Overall Modeling Structure. The execution model plays an important role in specifying the behaviors of the target surveillance system component processing. Figure 7 illustrates the overall structure of the proposed execution model interacting with the environment model and the application

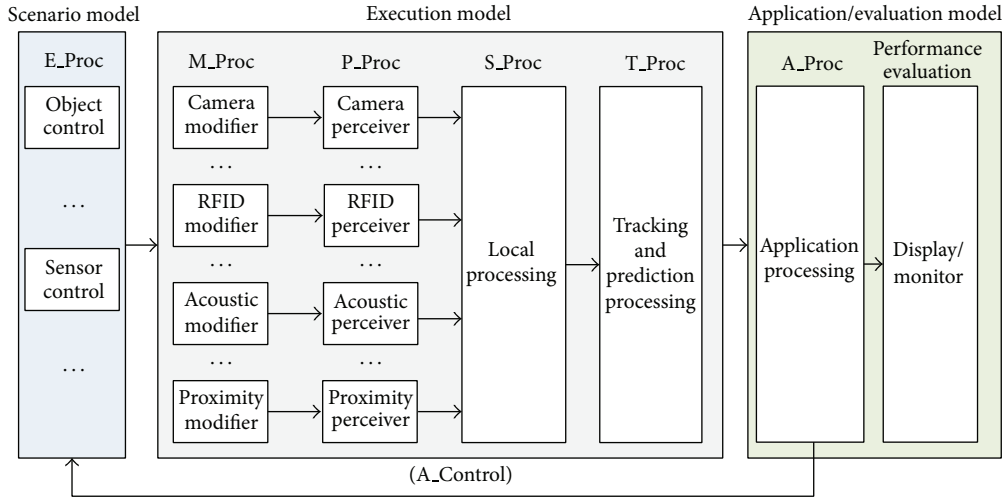

Figure 7: The overall structure of execution model interacting with environment model and application model.

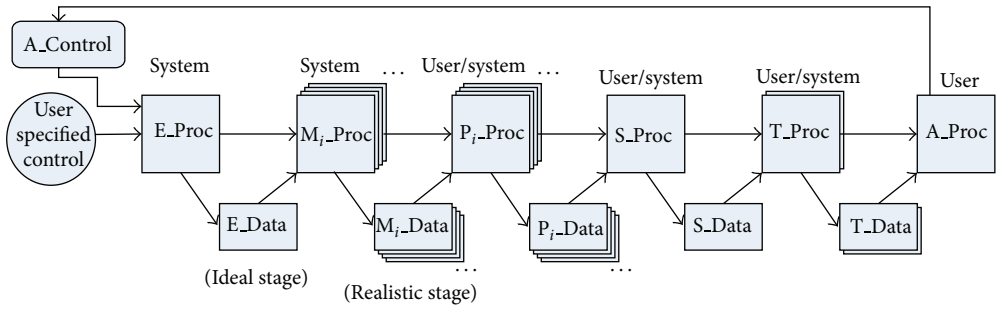

Figure 8: Illustration of the structure of the overall process and data.

model. The execution model takes the environment data from the environment model as input, and then it generates behaviorally processed low-level data which is system dependent and is commonly used for surveillance applications. The application model simulates a specific surveillance application by transforming the low-level data to synthetic high-level data which is application-dependent. In the end, the application model evaluates the surveillance application under the simulated environment by using the low-level and high-level data.

We categorize components of the execution model as sensor based and object based. The sensor-based execution models model the sensor's detection behavior of surveillance system. Modifier and perceiver shown in Figure 7 belong to the sensor-based models. A pair of modifier and perceiver are needed for each sensor. In the data perspective, the modifier represents ideal sensor-to-object relationship as an output, while perceiver gives realistic relationship by altering the ideal information by incorporating the algorithm behavior.

On the other hand, object-based component is for modeling behavior of system based on multiple sensor's collaboration. Local processing and tracking/prediction processing models are examples. Both models represent object-to-sensor relationship and pass new information made by multiple sensor's collaboration. However, the local processing model is under complete detection information about an object, while tracking/prediction processing model is under partial detection.

The overall platform operates as a sequential execution of different models as shown in Figure 8. The process of every model has code to execute and data to produce. The platform executes in the order of E_Proc, M_Proc, P_Proc, S_Proc, T_Proc, and A_Proc. In each model, a code is composed of system code and user code. The system code processes and generates data that are not application specific. The user code is application specific and must be provided as a template code to complete the system simulation. The system code in the process keeps generating system data necessary for the user code of the process or other processes to decide when and how to make a decision. On the other hand, the user code of a process generates user data by using the system data of the process and system/user data of other processes. At any given simulation iteration, all processes can use the data generated by the previous model as well as all of data obtained in the previous simulation iterations. A history of the data for all processes is also necessary for final evaluation.

4.2. Sensor-Based Execution Models

4.2.1. Modifier. As we discussed, the modifier takes the data from the scenario model and performs the idea sensor processing. The modifier keeps its local data structure, M_Data,

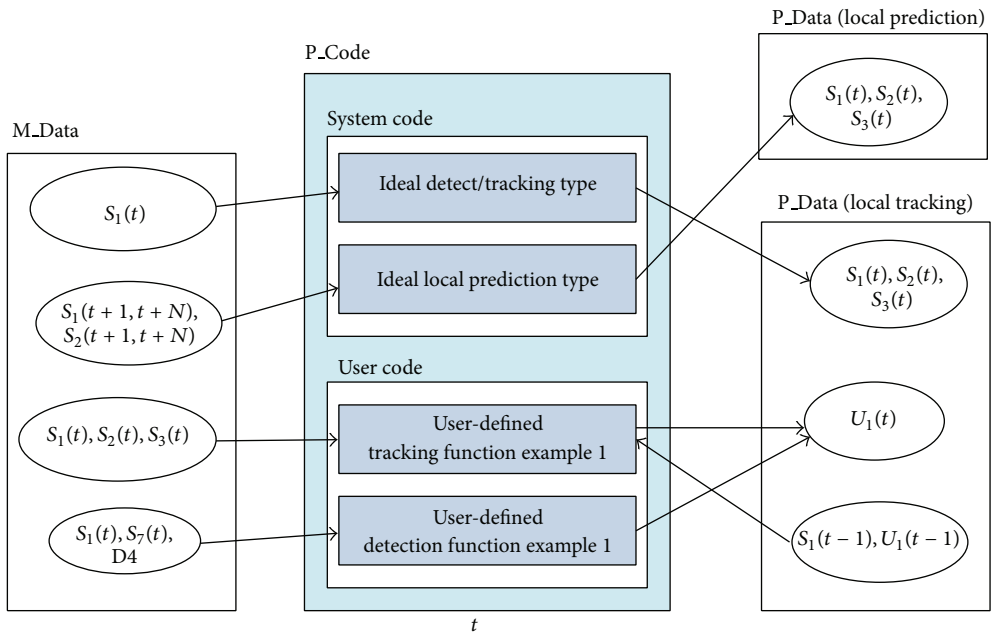

Figure 9: Perceiver code operation.

with five attributes, object ID, object position, neighboring object IDs, sensor ID, and object data. The object ID is for globally distinguishing objects which appear in the simulation. The object position is for the object's location with respect to a sensor. The neighboring object IDs are a list of objects IDs which are within a certain distance from the object ID with respect to a sensor. The object data is static information about an object which is related to a sensor but does not depend on object's status. In this paper, we consider camera, RFID reader, proximity sensor, and acoustic sensor for the modifier. For instance, when a modifier models a camera, the object position is about object's pixel position on an image. If a modifier models RFID reader, then the object data is a tag ID.

In order to model sensor's ideal detection, we consider two types of functions in the system code of M_Proc. Function type 1 has the characteristic of generating new object's information every simulation time. It is because the detection information depends on the relationship between object's global location and sensor's global location and detection condition. Function type 2 has characteristic of copying the static detection data from E_Data. Whenever an object is detected, the detection result is always constant. For instance, a camera and acoustic sensor are representative examples for function type 1, and RFID reader and proximity sensor are for function type 2.

4.2.2. Perceiver. The purpose of the perceiver is to modify the ideal information generated by the modifier into realistic information according to the algorithmic behavior specified as a user code. Even though an object can be detected in the modifier, the perceiver can change the detection status. The internal operation is illustrated in Figure 9.

In order to model object's local position prediction, P_Proc maintains P_Data. It has attributes object ID and track ID. The system code of perceiver is responsible for local tracking and prediction for which we define two types of functions. The first type for local tracking simply models ideal tracking by storing track ID and tracking result for each object in M_Data in P_Data. The track ID is generated if an object is detected at time t but was not at time $t-1$. For this, it checks if an object ID in M_Data appeared in the previous P_Data. The second type of function is for prediction. It applies the future object information in M_Data from $t+1$ to $t+N$ to the above first type of function. Thus, it stores the result in P_Data.

However, the perceiver process usually depends on the user code. It is because the perceiver is for modeling behavior of realistic detection/tracking system according to the parameters affecting the system. In fact, those parameters are given by a user. Thus, we categorize two types of functions in the user code of P_Proc. The detection function type makes object's actual detection data by injecting error or failure based on current object status. On the other hand, the tracking function type uses previous object status as well as current object status for making the actual detection data.

4.3. Estimation and Prediction Behavior Modeling

4.3.1. Sequence Model. While sensor-based execution models provide all necessary information from the sensor perspective, this information must be transformed into object perspective in order to incorporate the target algorithm behavior. The perspective transformation is performed in the sequence model. The internal data structure for the sequence model is illustrated in Figure 10. Once the information is transformed, the sequence model models the execution

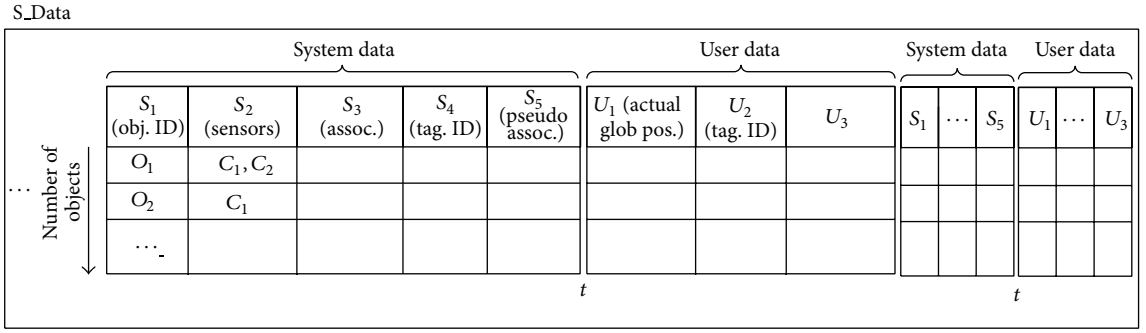

Figure 10: Information perspective transformation.

behavior of multiple sensor collaboration system. It especially focuses on the collaboration between homogeneous sensors.

In order to easily model the execution behavior of collaboration algorithm, the sequence model transforms sensor-based data which each perceiver maintains for its related sensor to object-based data. Thus, for each object, the system code of the sequence model maintains S_Data with a list of sensor IDs which sense the object, detection result according to sensor types including sensor ID, event type, and sensed data. For each simulation, it creates S_Data by copying the above data from system data of each perceiver.

Then, the sequence model generates algorithm result according to parameterized sensor collaboration algorithms. The association algorithm precedes localization algorithm. Thus, modeling for localization algorithm using two sensors requires that association algorithm using same sensors necessarily is defined. For the sequence order, the sequence model executes the association algorithm model first and then the localization algorithm model later. It also clearly specifies used sensors with the result as an output. For association algorithm, when sensor IDs in S_Data for an object include two sensor IDs of an algorithm parameter, the sequence model determines the object that is associated with those two sensors.

4.3.2. Tracking and Estimation. The tracking/prediction model models the execution behavior of algorithm using multiple sensors requiring multiple cycles, while the sequence model models the execution behavior determining its output at the single (current) cycle time. An example modeled by the tracking/prediction model is the association algorithm between heterogeneous sensors. Since they use different sampling time and their sampling time is not synchronized, it is impossible to determine the association result in the cycle time. Thus, in the real implementation, a time window to associate data from different sensors is often used. Similarly, the tracking algorithm requires multiple cycles to guarantee object's seamless tracking. Due to the imperfect sensor detection, object is not detected by a sensor such that the global tracking using multiple sensors may fail. However, such failure is often recovered by using previous successful tracking information about an object.

The execution behavior of global tracking is modeled by maintaining a global ID for an object. For each global tracking algorithm, the tracking model considers objects whose sensor IDs include algorithm representation's sensor IDs, every simulation time.

The tracking/prediction model also models the prediction behavior of algorithm using multiple sensors estimating the future result of data. A surveillance application in need of this prediction mechanism is for the optimum sensor scheduling and the object coverage. By predicting existing object's global locations, the coverage of sensors is dynamically reconfigured.

4.4. Application Behavior Modeling and Interaction

4.4.1. Application Model. The execution behavior modeling described in the paper specifies a low-level local processing. In order to complete the surveillance system specification and modeling, all data in the platform must be utilized. In the proposed platform, the application behavior modeling is done through event-based processing. To support event-based processing, we represent an event source with the sensor type and event name. A list of event sources for an application model is defined in the user script. For each event source, the system code of the application model searches if an object in T_Data has a specified event name as output of a specified sensor type. Then, the application model copies T_Data of the object to A_Data such that the remaining processes of the application model operate based on the A_Data. As a result, by filtering data about predefined events, the event-based application is modeled in the data level.

4.4.2. Sensor Scheduling with Reconfiguration. In many surveillance systems, the sensors are scheduled according to the change in the environment, for example, the changes in object density distributions and traffics. Based on the object traffics and sensor availability, the system often performs sensor scheduling and reconfiguration (i.e., view modifications). This execution behavior can be easily modeled by providing a user code in A_Proc model. A user code models sensor's dynamic reconfiguration fed back by an application by manipulating sensor reconfiguration data. In many surveillance applications, sensors are often reconfigured for sensing an object of interest sophisticatedly in the middle of application operation. A user code in the application model simply updates target sensor's configuration values.

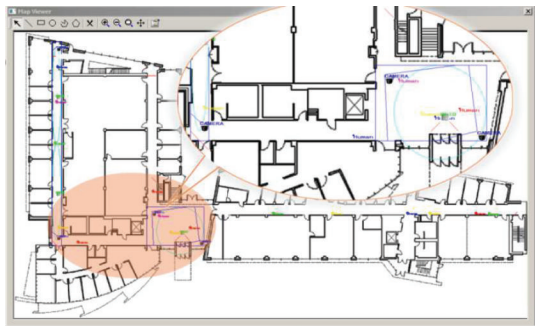

Figure 11: The snapshot of the proposed system during simulation of body detection application.

The sensor reconfiguration is modeled through a user code, not the system code. When a user code updates the target sensor configuration values in the sensor parameters, the interface function in the environment model adjusts the current sensor configuration to the target configuration according to sensor's reconfiguration characteristics.

4.4.3. Collaboration of Surveillance Applications. Modeling the behavior of collaboration within the surveillance applications is physically set up as separate systems or as a single system with multiple concurrent processes. In either case, the common behavior is that the application models exchange the information that they generate within their local domain. It is because the application model modeling surveillance applications maintains the user data for them together. Such collaboration between multiple concurrent local processing can be easily modeled in the proposed platform. The specification of their behavior is also modeled in A_Proc model. Since all necessary information is available, a user code is able to use the previous or current data exactly the same as the behavior of information exchange. Specifically, the application model is able to model for scheduling sensors and sensor load balancing in large scale surveillance domains. In order to properly model, the platform maintains all the objects which are globally identified through sensors by the execution model. The application model obtains a list of objects that each sensor captures from the execution model. Thus, it assigns a camera to each object.

5. Case Study

In the case study, our objective is to demonstrate that the proposed modeling methodology is able to represent the large scale surveillance system, by reusing parameters of actual visual detection/tracking algorithm that we developed internally. The entire system is specified with scripts without the actual implementation. We simulate the surveillance system in CEWIT research center of Stony Brook University. The target application scenario is configured with 8 cameras and 8 RFID sensors as shown in Figure 11. The RFID coverage is modeled as a time-varying boundary. A face based detection algorithm is modeled in this study (i.e., an object is detected when the face with a specified minimum area is visible by the camera). A prior information based tracking is modeled (i.e., an object is tracked if previous detection history is available). The objects motions are pseudo randomly generated with a specified start position and time, end position, and average velocity. In case of crowding, the direction of motion is changed to avoid the collision (i.e., the object traffic models the employees of the center). The simulation has been conducted in order to see the effectiveness of the algorithm on object association performance (i.e., analysis of false association rate and miss association rate). The simulation has been executed for one hour real time but less than 3 minutes of simulation time.

The corresponding platform model is illustrated in Figure 12. In the platform model, interaction of data transfer and usages are illustrated. Since the target application requires extensive sensor collaboration, S_Proc and T_Proc heavily use their own local data. In addition, E_Data is accessed by the S_Proc directly in order to execute behavior of the sensor collaboration. As shown in the figure, S_Proc and T_Proc are extensively interactive for object association operations (associating visual information with tag information). Note that different execution behavior of the same environment can be easily modified for performance comparisons among different specifications. Since we have specified the detection with the face, the face detection user code and user data are generated in the model. Similarly, use codes and data are specified in the subsequent models.

Figure 13 illustrates overall visualization of the case study. The execution behavior of all sensors as well as the application can be evaluated. The visualization displays the status of all objects (i.e., indicating false and miss association). The performance of each sensor, the tracking performance, and the association performance can be visualized for effective verification of the specification. Based on this visualization capability, a long term performance can be investigated.

6. Conclusion

This paper presents a performance evaluation environment for effective surveillance system design. The platform is designed so that only the target application behavior specification is needed. The proposed modeling is organized with a chain of an environment model and a series of execution models so that an overall behavior of the system can be easily described and evaluated. All scripts are parameterized in

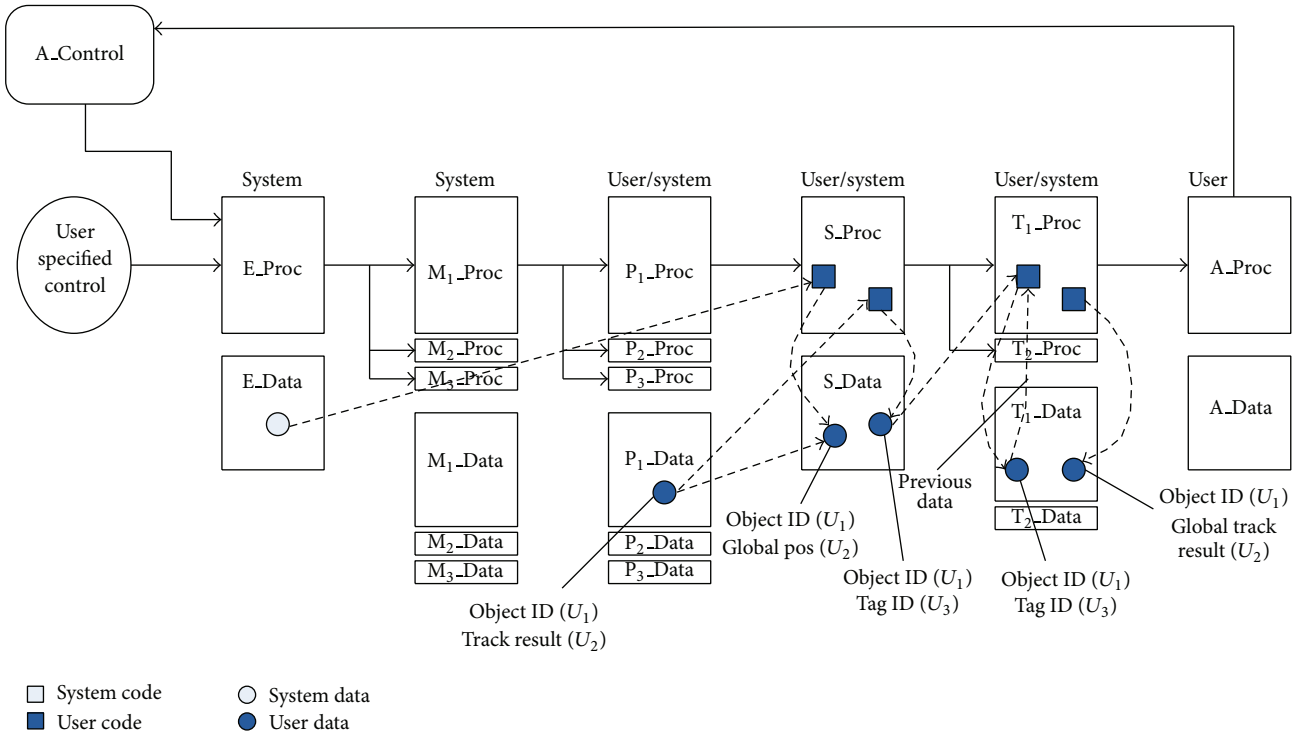

Figure 12: Illustration of modeling the execution behaviors where user codes are additionally defined for the realistic application modeling.

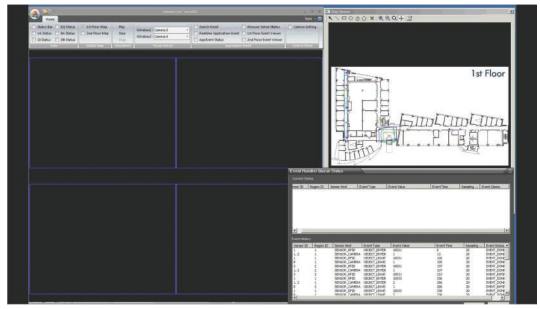

Figure 13: Illustration of modeling the execution behaviors where user codes are additionally defined for the realistic application modeling.

the proposed systems. Complex operations such as sensor collaboration can also be represented. The proposed modeling allows the designer to evaluate the target system through rapid prototyping. The proposed methodology is especially effective for large scale system design.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research was supported by the "Cross-Ministry Giga KOREA Project" of the Ministry of Science, ICT and Future Planning, Republic of Korea (GK14P0100, Development of Tele-Experience Service SW Platform Based on Giga Media).

References

[1] T. D. Räty, "Survey on contemporary remote surveillance systems for public safety," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 40, no. 5, pp. 493–515, 2010.

[2] C.-F. Shu, A. Hampaour, M. Lu et al., "Ibm sm art surveillance system (s3): a open and extensible framewor k for event based surveillance," in *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance (AVSS '05)*, pp. 318–323, September 2005.

[3] M. Valera and S. Velastin, "Intelligent distributed surveillance systems: a review," *IEE Proceedings—Vision, Image, and Signal Processing*, vol. 152, no. 2, pp. 192–204, 2005.

[4] S. H. Cho, Y. Nam, S. Hong, and W. Cho, "Sector based scanning and adaptive active tracking of multiple objects," *Transactions on Internet & Information Systems*, vol. 5, no. 6, pp. 1166–1191, 2011.

[5] W. Hu, T. Tan, L. Wang, and S. Maybank, "A survey on visual surveillance of object motion and behaviors," *IEEE Transactions on Systems, Man and Cybernetics C: Applications and Reviews*, vol. 34, no. 3, pp. 334–352, 2004.

[6] Y. Nam, S. Rho, and J. H. Park, "Intelligent video surveillance system: 3-tier context-aware surveillance system with metadata," *Multimedia Tools and Applications*, vol. 57, no. 2, pp. 315–334, 2012.

[7] S.-J. Oh, S. H. Cho, S. Hong, N. Moon, and P. Park, "Object association and identification in heterogeneous sensors environment," *Eurasip Journal on Advances in Signal Processing*, vol. 2010, Article ID 591582, 2010.

[8] S. H. Cho, S. Hong, and Y. Nam, "Association and identification in heterogeneous sensors environment with coverage uncertainty," in *Proceedings of the 6th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS '09)*, pp. 553–558, Genova, Italy, September 2009.

[9] C. Shahabi, S. H. Kim, L. Nocera et al., "Janus—multi source event detection and collection system for effective surveillance of criminal activity," *Journal of Information Processing Systems*, vol. 10, no. 1, pp. 1–22, 2014.

[10] D. Ghimire and J. Lee, "A robust face detection method based on skin color and edges," *Journal of Information Processing Systems*, vol. 9, no. 1, pp. 141–156, 2013.

[11] L. M. Brown, A. W. Senior, Y.-L. Tian et al., "Performance evaluation of surveillance systems under varying conditions," in *Proceedings of the IEEE PETS Workshop*, p. 18, 2005.

[12] J. Nascimento and J. Marques, "Performance evaluation of object detection algorithms for video surveillance," *IEEE Transactions on Multimedia*, vol. 8, no. 4, pp. 761–774, 2006.

[13] R. M. Hwang, S. K. Kim, S. An, and D.-W. Park, "The architectural pattern of a highly extensible system for the asynchronous processing of a large amount of data," *Journal of Information Processing Systems*, vol. 9, no. 4, pp. 567–574, 2013.

[14] F. Z. Qureshi and D. Terzopoulos, "Surveillance in virtual reality: system design and multi-camera control," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '07)*, June 2007.

[15] D. Terzopoulos, "Perceptive agents and systems in virtual reality," in *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST '03)*, pp. 1–3, ACM, New York, NY, USA, 2003.

[16] P. K. Atrey, M. Kankanhalli, and A. Cavallaro, *Intelligent Multimedia Surveillance: Current Trends and Research*, Springer, 2013.

[17] N. Howard and E. Cambria, "Intention awareness: improving upon situation awareness in human-centric environments," *Human-Centric Computing and Information Sciences*, vol. 3, article 9, 2013.

[18] Y. Lu, D. He, and H. Yao, "Optimal MLAT tracking based on improved simulation for airport surveillance," in *Proceedings of the 2nd International Conference on Systems Engineering and Modeling (ICSEM '13)*, pp. 972–975, 2013.

[19] T. J. Ellis, "Performance metrics and methods for tracking in surveillance," in *Proceedings of the 3rd IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS '02)*, Copenhagen, Denmark, June 2002.

[20] J. C. Augusto, V. Callaghan, D. Cook, A. Kameas, and I. Satoh, "Intelligent environments: a manifesto," *Human-Centric Computing and Information Sciences*, vol. 3, article 12, 2013.