Event Detection for Video Surveillance Using an Expert System

Barbara Krausz
Fraunhofer IAIS
Schloss Birlinghoven
53754 Sankt Augustin, Germany
barbara.krausz@iais.fraunhofer.de

Rainer Herpers
FH Bonn-Rhein-Sieg
Grantham-Allee 20
53757 Sankt Augustin, Germany
rainer.herpers@fh-bonn-rhein-sieg.de

ABSTRACT

Video Surveillance is in the center of research due to high importance of safety and security issues. Usually, humans have to monitor an area and often they have to do this for 24 hours a day. Thus, it would be desirable to have automatic surveillance systems that support this job automatically. The system described in this paper is such an automatic surveillance system that has been developed to detect several dangerous situations in a subway station. This paper discusses the high-level module of the system. Herein, an expert system is used to detect events.

Categories and Subject Descriptors

I.5.4 [**Pattern Recognition**]: Applications—*Computer Vision*; I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis

General Terms

Algorithms

Keywords

Event Detection, Video Surveillance, Expert System

1. INTRODUCTION

The final step in automatic video surveillance is the analysis of results of the low-level modules. For example, classification and tracking results are analyzed. The objective of the event detection module is to interpret the behavior of the observed entities (humans, vehicles etc.) and to generate a description of the events taking place in the scene or to raise an alarm if a dangerous situation is detected. It takes results of the low-level vision modules as input and tries to understand the semantics, that is it tries to understand what is happening in the scene. Since the results of the previous modules in a surveillance system are often inaccurate or even erroneous, the event detection module often has to cope with poor tracking or classification data. Another problem the event detection is concerned with is that activities which have to be detected may appear very similar. For example the activities "stealing an item" and "buying an item" are similar: a person enters a shop, picks up an item and exits the shop [13].
Event detection approaches can be distinguished as follows:

- *Detected events*: There are systems that try to detect abnormal activities. For example, in [15] probabilistic models for normal motion trajectories are learned from video sequences. Abnormal trajectories like cars driving into the opposite direction are detected by comparing the trajectories with the learned models.
  On the other hand, there are systems that look for predefined events. These events are mostly defined by domain experts. Hence, the system should provide an easy means of defining events that should be detected, so that domain experts can describe these events on their own without the help of computer scientists.

- *Scene context*: In most cases surveillance systems are applied in certain environments, so that the context of the environment can be used for inference. For example in [19] the scene context of a subway station is taken into account by modeling the equipment like seats, doors and ticket machines in 3D. In some cases simpler context information is provided to the system by specifying two dimensional polygonal zones connected to semantic information like entrance zone, platform and tracks [19]. In contrast to that, [15] does not take any context information into account.

- *Learning*: In many cases, video sequences are available showing normal activities. These sequences can be used to learn models, which are then utilized to distinguish normal activities from unusual activities. Video sequences showing abnormal activities can also be used to learn models for those abnormal activities with which such events can be detected directly.

This paper discusses an automatic surveillance system that analyzes tracking results as well as classification results in three steps. It generates events and asserts facts to the knowledge base of a rule-based expert system. The rules of this expert system represent seven predefined events that have to be detected. An inference engine analyzes the facts in the knowledge base and the rules and generates an alarm if necessary.

AREA'08, October 31, 2008, Vancouver, British Columbia, Canada.

The described system has been developed to detect dangerous situations in a subway station such as persons crossing the rails, dropping object on the tracks or being trapped by the door of a moving train. The video sequences this system is working on have been provided by the largest European public transportation company (RATP, France) within the International Conference on Advanced Video and Signal based Surveillance 2005, see [1]. Further details about the system can be found in [12]. The paper is organized as follows: The next section presents related work. Thereafter, section 3 briefly discusses which types of knowledge have to be represented in an surveillance system. Section 4 then describes the event detection approach implemented in our system. Before a conclusion is drawn, section 5 presents obtained results.

2. RELATED WORK

In the following an overview of works is given in which predefined events are detected and scene context is taken into account:

- *Finite State Machines*: [3] manually builds up deterministic automata that consist of states representing certain sub-scenarios like "the car enters zone A". The whole scenario is recognized if the automaton reaches a final state.
  [4] also employs finite state automata to recognize simple scenarios. For example, the automaton for the scenario "a person jumps over a barrier without validating his ticket" is composed of five states: a person is tracked, the person is at the beginning of the validation zone, the person moves fast, the person is beyond the barrier and the person is at the end of the validation zone.

- *Hidden Markov Models*: [18] uses Hidden Markov Models to analyze motion of a hand to recognize American sign language. With Parameterized-HMMs, a more complex approach is used in [21] to interpret human gestures. The greatest disadvantage of approaches using HMMs is the need for training samples. Furthermore, the topology and the number of states of the model have to be determined.

- *Bayesian Networks*: In addition to finite state machines, [4] uses Bayesian Networks for the recognition of more complex scenarios. This approach was proposed in [7], where for example a Bayesian Network is created and trained for the scenario "object A slows down toward object B". More recently, [13] applies Dynamic Bayesian Networks. Similar to Hidden Markov Models, training samples are needed to determine the parameters of the network.

- *Stochastic Context-Free Grammars*: [9] analyzes tracking results and generates discrete events like "car-enter" or "person-exit", together with likelihood values. These events form the vocabulary of a stochastic context-free grammar and are processed by a parser to recognize events.

- *Logical Predicates*: There are several works using logical predicates in combination with an inference mechanism. [16] and [17], for instance, define rules for activities like "stealing". In each frame the system analyzes the results of the low-level layers and generates facts, which are then put into a knowledge base. A Prolog inference engine is then used for reasoning.
  A very interesting approach can be found in [5], where Fuzzy Metric Temporal Horn Logic (FMTHL) is used for representing facts and rules. FMTHL extends conventional Horn Logic by a temporal and a fuzzy component. Additionally, a tool is provided for modeling activities that should be detected.

- *Petri Nets*: [6] makes use of Petri Nets, which have been shown to be similar to rule-based expert systems. A transition represents a rule, whereas markings correspond to facts. [6] describes an ontology for event recognition and shows the use of Petri Nets in the domain of a parking lot.

Besides these approaches, there are many other methods like Dynamic Time Warping or Self-Organizing Neural Networks, see for example [8] for an overview.

3. KNOWLEDGE REPRESENTATION

Usually, surveillance systems, which are designed to detect certain events in video sequences, rely on knowledge. Three main types of knowledge can be distinguished:

1. *Scene context*: Surveillance systems operate in certain environments. Mostly, knowledge about this environment is used for inference. Knowledge about the geometry of the environment can be provided to the system in form of 3D models (see for example [19]).

2. *Current state*: The surveillance system has to represent the current state of the scene in a formal way. For example it has to be aware of the humans' positions, their velocity and moving direction.

3. *Detected events*: As a surveillance system has the job of monitoring an area and raising an alarm, it has to know which events should be detected.

Knowledge must be provided to the system in a formal way. Its representation should be easy to express and to understand for humans, as often domain experts define activities the system has to recognize. Of course the representation of knowledge often depends on the chosen activity analysis approach in the surveillance system. For example in [4] knowledge about the activities that should be detected in the video sequences are provided to the system through the states and the topology of the finite state automaton.
An example for a knowledge representation language in the context of video surveillance is ERL (Event Recognition Language)[14]: ERL aims at formalizing the third kind of knowledge. It provides a language for representing events that should be detected by the surveillance system. Such a representation of an event has the following form:

```
EventType EventName (ObjType ObjName,...)
  Event description;
```

The `EventType` is `PRIMITIVE` for simple events, which are performed by a single actor, `SINGLE_THREAD` for a combination of primitive events or `MULTI_THREAD` for a combination of multiple single-thread events, which have some temporal, spatial or logical relations and even can involve multiple

scene objects. `Event description` contains the representation of the event. The description of multi-thread events can include boolean expressions or temporal constraints. The parameters `ObjType ObjName,...` include all objects that are involved in this event. This representation language is independent of the event recognition approach. In [14] an event is translated to a tree structure, which is passed to an inference mechanism. [14] proposes to recognize primitive events by using Bayesian Networks. Single-thread events are recognized by employing a variation of HMMs and multi-thread events can be detected by evaluating all sub-events and their relationships.
A similar knowledge representation language can be found in [20].

4. EVENT DETECTION

The event detection module in our system takes the results of the low-level vision modules as input. For each object detected in the scene the following information is available:

- centroid
- bounding box
- trajectory
- velocity
- moving direction
- classification (human, train or unknown object)

The module raises an alarm as output if one of seven predefined events takes place.
In the ideal case, the system should be flexible, so that it can be extended to detect further events. The knowledge representation should be very easy to understand, as domain experts should be able to formulate the events the system has to detect. It is very important for a surveillance system to analyze the results of the vision modules in real-time. Furthermore, knowledge about the scene context should be taken into account.

The event detection module is inspired by the works of [16] and [5]. As a detailed description of the events that have to be detected is provided, it seems natural to choose a rule-based approach. Figure 1 shows an overview of the event detection module. The core consists of a rule-based expert system. In the following section the basics of expert systems are introduced.

4.1 Expert Systems

We have utilized the rule-based expert system CLIPS[1]. Such an expert system offers a simple syntax for writing rules and facts so that even domain experts are able to understand and formulate rules on their own. Nevertheless, the language is very powerful and the expert system offers a ready-to-use inference mechanism, which works in real-time. In contrast to the system VidMAP in [16], which employs Prolog, our expert system offers forward chaining and therefore works data-driven, whereas Prolog works goal-driven. For that reason, VidMAP contains a reasoning thread that is invoked every 5 seconds to insert facts into the knowledge base and query the Prolog engine.

[1]CLIPS: http://www.ghg.net/clips/CLIPS.html

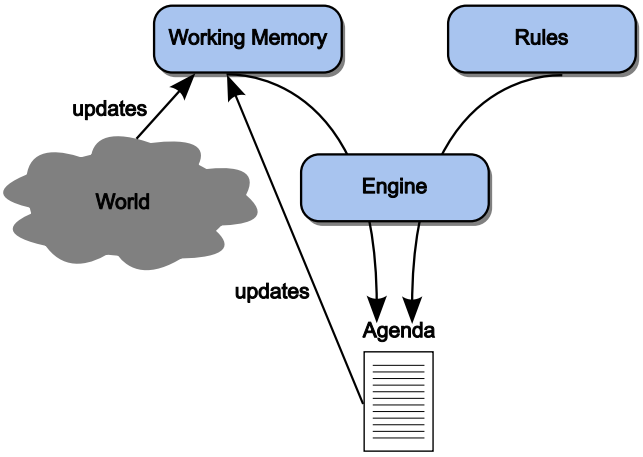


Figure 2: Expert system with forward chaining [11].

Figure 2 shows the general layout of an expert system with forward chaining. Such a system consists of a *working memory*, a *rule set* and an *inference engine*. The working memory contains all facts. A fact represents a piece of information and is the basic unit of data in an expert system [2]. In CLIPS the structure of (non-ordered) facts can be defined using the `deftemplate` construct, for example

```
(deftemplate trackobject
        (slot label)
        (slot classification)
        (multislot centroid)
        ...
)
```

For example, a fact like `(trackobject (label 1) (classification human)(centroid 55 103))` can be added to the list of facts. These facts in the working memory represent the knowledge about the current state. New facts are added to the working memory if the current state changes. The working memory is also updated by rules. In CLIPS rules have the following form [10]:

```
(defrule <rule-name>
      <premise 1>
      ...
      <premise m>
      =>
      <action 1>
      ...
      <action n>
)
```

The left-hand side of a rule consists of a set of conditions. The right-hand side defines actions that are executed if the rule fires. Often new facts are added to the working memory, existing facts are modified or retracted. The inference engine determines which rules are applicable and puts them onto the *agenda*. If multiple rules are applicable, the order of the rules is determined by considering a conflict resolution strategy. All applicable rules are fired according to the conflict resolution strategy until no applicable rule remains.

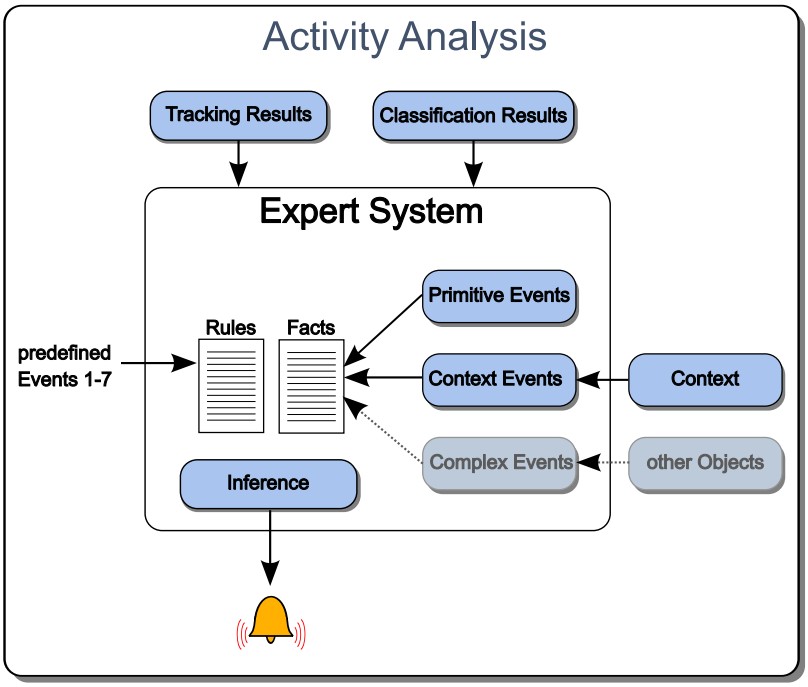


Figure 1: Event detection. This module uses an expert system to analyze the activities taking place in the scene. For that purpose, tracking and classification results are analyzed and facts are inserted into the knowledge base. Firstly, tracks are considered independently to generate primitive events. Secondly, the context of the environment is taken into account and context events are generated. The last step in the activity analysis module (complex events) is skipped in this system, because no interactions between objects in the scene have to be analyzed.
An inference engine analyzes the facts in the knowledge base and the rules, which model the seven predefined events that have to be detected. If such an event is detected, an alarm is raised.

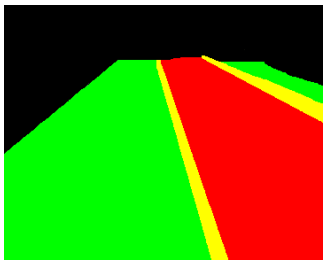
4.2 Knowledge Representation

As stated before, there are three main types of knowledge a surveillance system is concerned with. Our system represents this knowledge as follows:

1. *Scene context*: Knowledge about the scene context is provided to the system by the definition of zones. Figure 3 shows the zones of interest. The green zones correspond to the platforms, the yellow zones represent the white lines and the red zone stands for the tracks. If three dimensional information about the scene context (tracks, ticket machines, seats) was available, this information could be represented by VRML models.

2. *Current state*: The current state of the scene is represented by the facts that are asserted to the knowledge base of the expert system.

3. *Detected events*: Predefined events our system should be able to detect are represented by the rules of the expert system.



(a)

(b)

Figure 3: (a) Zones of interest for the subway station in (b). The green zones correspond to the platforms, yellow zones represent the white lines and the red zone corresponds to the tracks.

4.3 Facts: Event Generation

Similar to [5] the objects detected by the vision modules are analyzed in multiple stages. Knowledge about the current state of the scene is then represented by asserting facts to the knowledge base. Firstly, a single object is considered independently: the results of the vision modules are analyzed and events are generated for this object. As [5] points out, there are three levels that can be used to analyze an object: When examining the trajectory, events like "run" or "turn around" can be generated. Considering the body posture of a person, it can be concluded, if a person sits down, stands or moves the head up and down. If the face of a person is analyzed, events like "concentrated" or "worried" could be generated.
Secondly, an object is analyzed with respect to its environment. Now events are generated for this object by taking knowledge about the scene context into account. In the last step interactions between objects in the scene are examined by considering all detected objects. Figure 4 shows some examples for these three types of events.

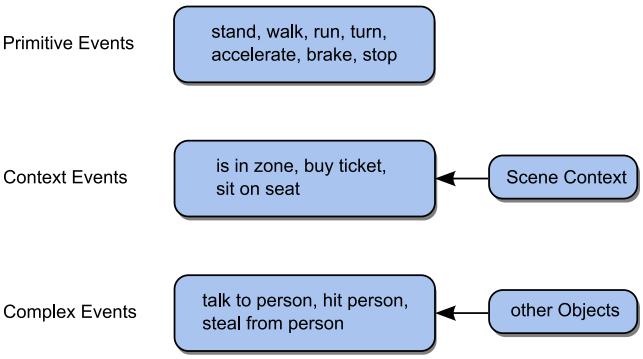


Figure 4: Event Generation (see [5]).

In the following the event generation in our system is described.

Primitive Events.

As stated before, the inputs of the analysis module are results of the tracking and the classification module. For each object present in the scene, there are six pieces of information, which have to be analyzed in the first step. The results of this first step are *primitive events* (see [5]). The word "primitive" refers to the fact that each object is processed independently. The following *primitive events* are generated:

- *stopped*: An object has stopped, if its velocity in x-direction and y-direction are both below one pixel per frame.
- *moving_normal*: An object moves, if one of its velocity components is below three pixels per frame.
- *moving_fast*: An object moves fast, if one of its velocity components is greater than three pixels per frame.
- *appears*: An object has just appeared in the scene, if a corresponding fact is not yet in the knowledge base.
- *disappears*: An object has just disappeared, if it is no longer present in the scene.
- *direction_angle*: The moving direction is measured in degrees.

Context Events.

In the second stage information about the scene context is taken into account and *context events* are generated. Knowledge about the scene context is provided to the system by

defining *zones of interest* (see section 4.2). By considering this context information together with the current position of an object, our system can conclude in which zone the object is located in. Since in our case no calibration data for the cameras is available, it is not possible to estimate an object's position in three dimensional coordinates. Thus, the zone a human is located in is estimated by considering the highest y-value of a blob corresponding to a person. The context events that are generated for each object are:

- *inside_zone*: All zones an object is located in.
- *completely_inside_zone*: If an object is located in just one zone, it is completely inside this zone.
- *moving_to_zone*: By considering the current position and the moving direction, the zone an object is moving to can be computed. This movement could be used as a hint. For example, if a person stands on the platform, but moves to the tracks, a proximity warning is more probable. Note that in the current implementation this hint is not used.

Complex Events.

If a surveillance system has to detect interactions between objects, like two persons talking, a third step in the analysis stage could be applied. After having analyzed an object independently (primitive events) and with respect to the context of the scene (context events), interactions between objects could be examined by considering all objects detected by the vision modules. In the case of this system no interactions between objects have to be detected, so this step is skipped and no complex events are generated.

4.4 Rules: Event Detection

In the following subsections the detection of some events is explained. For each event there exists a rule in the rule set of the expert system that invokes a function that just logs the alarm. This function could be extended to ask a human operator to evaluate the situation or to cause a train to stop.

Proximity warning.

A proximity warning for object o is raised if it is classified as a human and if it is completely inside the yellow zone (white line, see figure 3) or if it is inside the green and the red zone.

Dropping objects on tracks.

This alarm is raised for an object with label `?id` (in CLIPS `?id` denotes a variable), if it has the properties defined in listing 1. The angle of the moving direction (`?angle`) must lie between $-35°$ and $-145°$ where an angle of $0°$ corresponds to a moving direction to the right.

Launching objects across the platforms.

This event is similar to the previous event. In contrast to that, the angle of the moving direction (`?angle`) must be greater than $-35°$ or less than $-145°$.

Person trapped by the door of a moving train.

This event is recognized if the bounding box of an object classified as train overlaps the white line (see listing 2), as

```
(trackobject (label ?id)
    (classification unknown)
    (motion ~stopped)
    (completely-inside-zone tracks)
    (angle ?angle))
```

Listing 1: Properties of an object droppped on the tracks. The operator $\sim$ is a connective constraint. Such a constraint is satisfied, if the following constraint is not satisfied (see [2]). In this case the object must not have stopped. The angle of the moving direction must lie between $-35°$ and $-145°$.

the blobs of the train and the person merge into a single foreground region. Additionally, a proximity warning is raised, too.

```
(defrule alarm4
    ?fact <- (trackobject (label ?label)
                (bl ?minx ?maxy)
                (tr ?maxx ?miny)
                (inside-zone $?zones)
                (classification train))
    (test(member$ platform (create$ $?zones)))
     =>
     (alarm 4 ?label ?minx ?maxx ?miny ?maxy)
)
```

Listing 2: Complete rule for the event "Person trapped by the door of a moving train". bl stands for the bottom left corner of the bounding box and tl for its top right corner. If this rule is applicable, a function is called which logs this alarm together with the object's id and its bounding box.

Walking on rails.

A person is walking on the rails, if an object with label `?id` is detected, that has the properties defined in listing 3. The angle of the moving direction (`?angle`) must have a

```
(trackobject (label ?label)
    (completely-inside-zone tracks)
    (classification human)
    (angle ?angle))
```

Listing 3: Walking on rails.

value greater than $35°$ and less than $145°$ or less than $-35°$ and greater than $-145°$.

Fall on the tracks.

This alarm is raised, if a person touches the red zone. See listing 4 for the complete rule.

Crossing the rails.

This event is similar to the event "Walking on rails". The only difference is the constraint on the angle of the moving direction: Its value has to be between $-35°$ and $35°$ or it has to be greater than $145°$ or less than $-145°$.

```
(defrule alarm6
    ?fact <- (trackobject (label ?label)
                  (bl ?minx ?maxy)
                  (tr ?maxx ?miny)
                  (inside-zone $?zones)
                  (classification human)
                  (motion ~stopped))
    (test(member$ tracks (create$ $?zones)))
     (or (test(member$ line (create$ $?zones)))
     (test(member$ platform (create$ $?zones)))
     )
    =>
    (alarm 6 ?label ?minx ?maxx ?miny ?maxy)
)
```

Listing 4: Complete rule for the event "Fall on the track".

5. RESULTS

The detected events of our system have been compared to the ground truth annotation of the video sequences. Most of the predefined events could be detected with a sufficient detection rate. For example, the event "Walking on Rails" is detected with a detection rate of 83%, whereas "Person trapped by the door of a moving train" is always detected.
As the event detection module analyzes the tracking and classification results, it highly relies on the accuracy of these results. For example, the detection rate of proximity warnings is 67% as the low-level vision modules yield poor results. In particular our motion detection module has problems to detect shadows and thus the exact position of a person is not estimated correctly.
Our system has a sufficient performance with about 12 fps when executed in parallel on four processors. The event detection module is very fast due to the inference engine of CLIPS and therefore just about 5% of the processing time is spent on the event detection.
In summary, it is important to note that the most important criterion for the quality of an automatic surveillance system is the accuracy of the low-level modules. If the event detection module has to analyze inexact or erroneous data, its detection rate will be low.

6. CONCLUSION

The described event detection module uses an approach which distinguishes it from most other works in the field of surveillance systems. It provides a simple syntax for rules, so that even domain experts are capable of describing events that should be detected. Furthermore, the expert system CLIPS offers a fast inference engine. To improve results it is especially important to increase the accuracy of the low-level modules such as motion detection and tracking.

7. REFERENCES

[1] Call for real-time event detection solutions (creds) for enhanced security and safety in public transportation, 2005.

[2] *CLIPS: Reference Manual, Volume 1 Basic Programming Guide*, 2006.

[3] F. Bremond and G. Medioni. Scenario recognition in airborne video imagery, 1998.

[4] F. Cupillard, Alberto.Avanzi, F. Bremond, and M. Thonnat. Video understanding for metro surveillance. In *IEEE International Conference on Networking, Sensing and Control*, volume 1, pages 186–191, 2004.

[5] C. Fernandez, P. Baiget, F. X. Roca, and J. Gonzalez. Semantic annotation of complex human scenes for multimedia surveillance. In *10th International Conference on Advances in Artificial Intelligence*, 2007.

[6] N. Ghanem, D. DeMenthon, D. Doermann, and L. Davis. Representation and recognition of events in surveillance video using petri nets. In *Conference on Computer Vision and Pattern Recognition Workshop*, volume 7, page 112, 2004.

[7] S. Hongeng, F. Bremond, and R. Nevatia. Bayesian framework for video surveillance application. In *International Conference on Pattern Recognition*, volume 01, page 1164, 2000.

[8] 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*, 34(3):334–352, August 2004.

[9] Y. A. Ivanov and A. F. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):852–872, 2000.

[10] P. Jackson. *Introduction to Expert Systems (3rd Edition)*. Addison Wesley, 1998.

[11] A. Kazarov and Y. Ryabov. Clips expert system tool: a candidate for the diagnostic system engine, 1998.

[12] B. Krausz. Detection of events in video surveillance. Master's thesis, FH Bonn-Rhein-Sieg, 2007.

[13] J. Muncaster and Y. Ma. Hierarchical model-based activity recognition with automatic low-level state discovery. *Journal of Multimedia*, 2:66–76, 2007.

[14] R. Nevatia, T. Zhao, and S. Hongeng. Hierarchical language-based representation of events in video streams. In *Workshop Event Mining*, page 39, 2003.

[15] S. Rao and P. Sastry. Abnormal activity detection in video sequences using learnt probability densities. *Conference on Convergent Technologies for Asia-Pacific Region*, 1:369– 372, 2003.

[16] V. D. Shet, D. Harwood, and L. S. Davis. Vidmap: Video monitoring of activity with prolog. In *IEEE Conference on Advanced Video and Signal Based Surveillance*, 2005.

[17] V. D. Shet, D. Harwood, and L. S. Davis. Multivalued default logic for identity maintenance in visual surveillance. In *Computer Vision - ECCV*, pages 119–132, 2006.

[18] T. Starner and A. Pentland. Real-time american sign language recognition from video using hidden markov models. *International Symposium on Computer Vision*, page 265, 1995.

[19] V. T. Vu, F. Bremond, and M. Thonnat. Human behaviour visualisation and simulation for automatic video understanding. In *10th International Conference on Computer Graphics, Visualization and Computer Vision*, pages 485–492, 2002.

[20] V.-T. Vu, F. Bremond, and M. Thonnat. Automatic video interpretation: A novel algorithm for temporal scenario recognition. In *18th International Joint Conference on Artificial Intelligence*, pages 1295–1302, 2003.

[21] A. D. Wilson and A. F. Bobick. Recognition and interpretation of parametric gesture. In *Sixth International Conference on Computer Vision*, pages 329–336, 1998.