High-level Hierarchical Semantic Processing Framework for Smart Sensor Networks

Dietmar Bruckner †, *Member, IEEE,* Jamal Kasbi, ‡
Rosemarie Velik †, *Member, IEEE,* and Wolfgang Herzner ‡, *Member, IEEE*
† Vienna University of Technology, Vienna, Austria
{bruckner , velik}@ict.tuwien.ac.at
‡ Austrian Research Centers, Vienna, Austria
{jamal.kasbi, wolfgang.herzner}@arcs.ac.at

Abstract — **This paper presents the framework of a novel approach to combine multi-modal sensor information from audio and video modalities to gain valuable supplementary information compared to traditional video-based observation systems or even just CCTV systems. A hierarchical, multi-modal sensor processing architecture for observation and surveillance systems is proposed. It recognizes a set of pre-defined behavior and learns about usual behavior. Deviations from "normality" are reported in a way understandable even for staff without special training. The processing architecture including the physical sensor nodes is called SENSE (smart embedded network of sensing entities) [1, 4].**

Keywords — **sensor networks, sensor fusion, semantic symbols, data mining, hierarchical model**

I. Introduction

In current times, observation systems for public spaces become more widespread. They are a visible reaction for the public on threats like terrorism and crime.

This paper describes the concepts of the semantic processing layers in a network of SENSE nodes [1], [4]. The goal of these layers is to learn the "normality" in the environment of a SENSE network, in order to detect unusual behavior, situations, or events and to inform the customer in such cases [5]. SENSE consists of a network of communicating sensor nodes, each equipped with a camera and a microphone array. Those sensor modalities observe their environment and deliver a stream of so-called low-level symbols (LLS, e.g. moving objects, or sounds). In the reasoning unit the LLSs are processed in order to inform the person in charge in case of above mentioned events occur. The first application area of SENSE will be an airport, therefore all alarms and other considerations are taking into account the needs of the airport staff.

The goal will be achieved in several steps. First, the information received from the sensor layer, i.e. the uni-modal (audio and video) symbols, is pre-processed, which includes deletion of spurious objects and trying to track symbols by correlating the received sensor messages (snapshots at distinct points in time) over time.

Second, the pre-processed uni-modal symbols are fused, which results in integrated high-level symbols (HLS) characterized by the combined uni-modal properties. These symbols are input to the semantic symbol learning process, which derives the models for typical behavior and properties of the different objects categories – persons, luggage, etc. These models describe normal behavior with distributions of properties like speed and direction with respect of location. As the paths which symbols take through the (visual) sensing area of a SENSE node are an important aspect of behavior, trajectories are derived from the semantic symbol models.

All of the methods used in the particular layers are widely used in e.g. observation systems and many other applications, but to our knowledge no other system uses a combination thereof in order to let the messages of the system really look smart and meaningful to the user.

This paper is structured as follows: the next chapter outlines the system architecture, while chapter 3 describes the individual layers in more detail. Chapter 4 discusses tracking of low-level symbols, and chapter 5 contains conclusion and outlook.

II. Architecture Overview

An 8-tier data processing architecture is used, in which the lower levels will be responsible for a stable and comprehensive world representation to be evaluated in the higher levels (Fig. 1). At first, the modality symbols will be checked regarding their plausibility – the audio symbols with respect to position, the video symbols with respect to position, and size. In the second processing layer (in this paper, we will use *tier* and *layer* as synonyms), symbol tracking will take place. Here, symbols which pass the first (spatial) check will be checked regarding their temporal behavior using a Markov Chain Monte Carlo based particle filtering approach. The output of this tier is a stable and comprehensive world representation including uni-modal symbols. Tier 3 is the sensor fusion tier, in which the uni-modal symbols are fused to form multi-modal symbols.

Layer 4 is the parameter inference machine, in which probabilistic model(s) for symbols' parameters and events are optimized. The results of this tier are models of high-level symbols and features that describe behavior. In tier 5, the system learns about trajectories of symbols. Typical

1-4244-1543-8/08/$25.00 ©2008 IEEE

paths trough the view of the sensor node are stored. The 6th layer has the task of managing the communication to other nodes and establishing a global world view. The trajectories are also used in this layer to find out if a trajectory of one node can be prolonged over a neighboring node. In tier 7, the recognition of unusual behavior and events occurs with two approaches. One part compares current symbols with the learned models and trajectories. Therefore, this sub-layer calculates probabilities for the existence of symbols with respect to their position, velocity, direction, and probabilities for trajectories of symbols. It also calculates probabilities for the duration of stay of symbols in areas, probabilities for the movement along trajectories, also across nodes (global map, scenario recognition). Symbols with such probabilities below defined thresholds raise "unusual behavior" alarms. The second part of tier 7 is concerned with the recognition of pre-defined scenarios and the creation of alarms in case pre-defined conditions are met.

Finally, layer 8 is responsible for communication to the user. It generates alarm or status messages and filters them if particular conditions would be announced too often or the same event is recognized by both methods in layer 7.

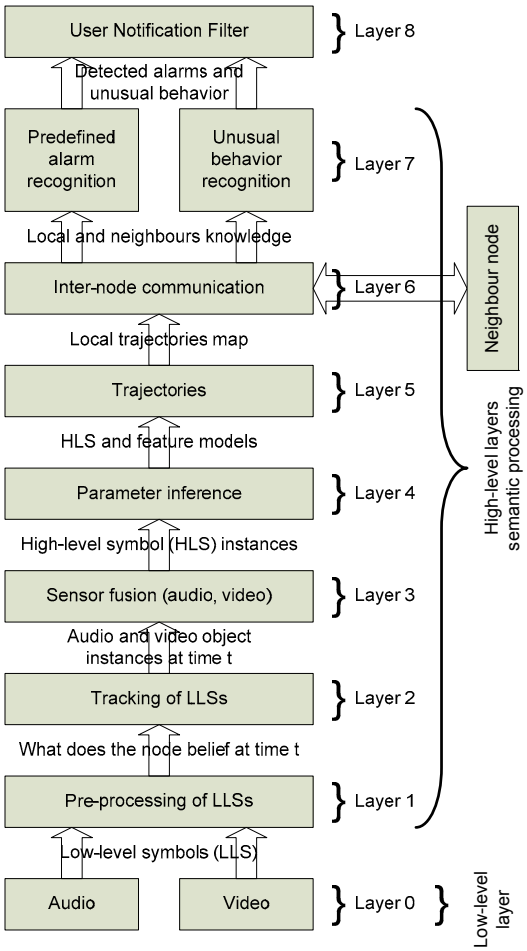

Fig. 1. Semantic Processing Layer Software Architecture

The visual feature extraction (layer 0) processes frame by frame from the camera in 2D camera coordinates. First results from the test videos show that visual detection can deliver significantly changing data from one frame to the next. In case of unfortunate conditions for the camera (many persons, they exchange positions in the crowd, bad light conditions, etc), detected symbols can change their label from person to object to person-group and back (for the same physical object). The size of detected symbols can change from small elements like bags to large groups of persons covering tens of square meters and including previously detected single persons and other objects. Consequently, the higher levels have to be prepared to work with imperfectly detected symbols.

III. DESCRIPTION OF TIERS

A. Low-level feature extraction

Description

This layer is the processing unit providing the semantic processing layer with uni-modal data streams describing what is observed by the sensors of a SENSE node at time t (i.e. in the environment where the node is embedded). The audio and video low-level symbols (LLS) represent defined primitives (Video: Person, Person Flow, Luggage, etc. Audio: Steps, Gun shooting, etc.).

Functionality

The particular functionality of this layer is not within the scope of this paper. We just want to point out that the audio and video data must be provided synchronized, i.e. time-stamped with reference to the same clock.

Interfaces

Consisting of two components, one for the audio and the other for the video modality, this layer provides two interfaces:

Through the Audio Interface, the audio symbols are sent to layer 1. An audio LLS consists of a label, describing the type of the detected audio symbols, the direction of arrival, the loudness, etc.

Through the Video Interface, the video symbols are propagated. A video LLS consists of a label, describing the type of the detected video symbols, the position in pixel, the size, the velocity, etc.

B. Pre-processing including plausibility checks

Description

During visual feature extraction, a set of templates is matched with the current frame. The templates are scaled in order to find objects of various sizes. In order to filter unrealistic primitives from the data stream, at first we intend to learn about average size of primitive symbols depending on their type and position in camera coordinates. The second plausibility check is done on bounding boxes. The bounding boxes of symbols are taken to determine whether some person or object is blend into a larger object. In this case the count of smaller and larger symbols decides which kind of symbol is most probable and will be used for further processing. Similar to the size of symbols, also their average speed will be learned by the sensor. This information will be used for symbol tracking, too. We assume the existence of points in times, where no persons are in the sensitive area of the

node. All internal scenarios and symbols will be reset at these moments. When new LLSs appear, they will be tracked over time with respect to their position, size, and speed.

Functionality

For the average size of symbols, a Gaussian or mixture of Gaussians model will be utilized. One average size model will be used for pixel clusters, so that the 640x480 camera pixels translate to 40x30 pixel clusters, each 16x16 pixels large. Each pixel cluster has models for each type of symbol. The models may need different parameters dependent on the number of persons, i. e. it may turn out that people behave differently in groups than they do alone or in pairs. Due to the fact that symbols can change their type from frame to frame it would not be a good solution to delete all improbable symbols, therefore they are just marked. Next, we will use fuzzy logic to find out symbols that do not change much from frame to frame and sort them out as being able to track. Third, all other symbols that may appear, disappear, change their size, etc.: to find out if they overlap in the frame – and are not recognized as before because of the overlap – we take the bounding boxes and test if a smaller object blended into a larger or if a larger object split into smaller ones. If so, these symbols may be correct detections, but we cannot assign speed, and we do not know if the detection as small or big symbol is dominant over time. So, with the time, the timely symbol count for large or small symbol will determine how this symbol is handed over to the next layer. Finally, we will compute the speed of symbols.

In this process, it is necessary to evaluate more than the immediate past frame. Restrictions on the computational power will show the possibilities in this respect.

After all plausibility checks, a voter will decide if a symbol is handed over to the next layer.

C. Tracking

Description

This layer uses particle filter techniques to track the pre-processed symbols. The basic assumptions for the algorithm are presented in detail in the next chapter.

Traditionally, multiple objects (in the area of particle filters, “objects” are tracked, not “symbols”, therefore this term is used here) are tracked by multiple single-object-tracking filters. While using independent filters is computationally tractable, the result is prone to frequent failures. Each particle filter samples in a small space, and the resulting “joint” filter’s complexity is linear in the number of targets n. However, in cases where targets do interact, as in many of our scenarios, single particle filters are susceptible to failures exactly when interactions occur. In a typical failure mode, several trackers will start tracking the single target with the highest likelihood score.

Functionality

A particle filter specifically designed for tracking interacting objects [2] is used to track the pre-processed symbols. The approach for addressing tracker failures resulting from interactions is to introduce a motion model based on Markov random fields (MRFs) [11].

D. Sensor fusion

Description

This tier gets as input the stable uni-modal symbols. Its task is to fuse audio and video symbols. One possibility is to use factor analysis [10, 12] to determine the correlation between audio and video LLSs. The output of this tier will be a symbolic representation of the real world in form of a collection of multi-modal symbols [13].

Functionality

Fusion of the audio and the video data is a task that can be done using the correlation between the provided data streams. Based on the time correlation of LLS, features that can be taken into account for this purpose are: loudness, direction of arrivals, power spectrum, size of the video symbol in pixels, and position.

E. Parameter Inference

Description

This layer will process the incoming symbols of fused LLSs and adapt the parameters of the used probabilistic model(s) to fit the data. The data are defined as the set of all the instances of semantic symbols.

Functionality

Getting any symbol from the sensor fusion layer, the task of this layer is to infer the parameters of the underlying probabilistic model (Mixture of Gaussians, histograms, or mixture of factor analyzers [3]). Using an online version of the Expectation Maximization (EM) algorithm to find the parameters of the model(s), we can focus on the variant where the system learns the behavior only of the recent past (time window), by using online average moving. We also can assume that the data fits a static model and therefore we can use the "gradient descent variant".

The use of non-parametric methods like histograms (in particular, k nearest neighbors) can be taken into account. Generally, we are using online clustering methods like described in [8], [9], [10], [11], [12].

F. Trajectories

Description

Trajectories in a node will be derived through the use of a learned transition matrix consisting of transitions between model clusters. This could be done by using a local search for the most likely sequence. Each HLS (model) must therefore keep a list of all local trajectories to which the symbol is belonging and at the time t, in which an instance is being observed and belonging to that symbol, the suitable trajectory (which should be active for the instance) must be selected. The node must activate all the local trajectories which are possible at time t for the observed instance. Additionally, all the neighboring nodes, to which the node has some correlations, must also activate the suitable trajectories. Due to the fact that one HLS could belong to more than one trajectory (after learning), the possible trajectory will not be necessarily unique. But adding the real time information, i.e. the instance at time t, which is belonging to just one HLS, and which is in turn belonging to one or more trajectories, the

local and global trajectory is unambiguously identified.

Functionality

The dynamics of the detected objects in the environment of the node are described by building the local trajectories map of a node. The map building process makes use of the attributes of the HLS; especially, it uses the velocity attributes mean and variance in the correspondent position.

Each of the involved nodes has to learn the global possible trajectories (including the probability of having the trajectory active, the density - high, middle, low - the direction, and the velocity). Then each node keeps a lookup table or a matrix where all the possible trajectories are registered. A simple switch from one local or global path to another one must cause an alarm (because it is unusual that people take this path). This matrix of all the possible trajectories will be built using the local transition matrices and the weight matrices (correlations between the HLS in the different nodes).

G. *Inter-Node Communication*

Description

Inter-node communication is based on the Loopy-Belief Propagation (LBP) algorithm [6, 7]. LBP will be used to form collections of neighboring nodes and to map the positions within one node's view into another one's view. This information can finally be used to e.g. store the trajectory of persons over multiple nodes, or to find out if somebody tries to escape observation.

Functionality

This module will be at the heart of the information exchange within the SENSE system. This layer sends and receives messages from and to other nodes in the SENSE system. Those nodes must be reachable by the nodes (i.e. "neighbors" with sensing areas which overlap with that of the respective node). The information of neighborhood is stored in a matrix and is therefore a dynamic parameter that can change over time.

Another task of this layer will be to update the compatibilities and therefore the weights between HLSs.

Interfaces

Layer 6 (Inter-Node communication) → Layer 7 (Alarm Generator): We assume that the sub layer inter-sensor communication provides the sub layer alarm generator with the global correlations that concerns the map (trajectories), the high-level symbols, the global tracking of persons, and the correspondent attributes. This affects the node-to-node interface.

Layer 6 (Inter-Node communication) ↔ Neighbors: For learning the global structure through the high-level symbols, only two variables will be exchanged: the belief of the node given the evidence at time t and also the belief of the node without the evidence at time t. (The evidence is the sensor reading). The exchange of the HLS including its parameters (attributes, detected instances at time t, local trajectories, and global trajectories) should also be done. It will be investigated whether other information can also be exchanged.

H. *Alarm Generator*

Description

This tier serves to detect predefined alarms and further unusual behavior. It generates alarms in case of very unlikely symbols (as a result of the above mentioned tracking and probability estimating tasks). A second – partly independent – part will be the detection of predefined scenarios: The information of persons moving, meeting, having luggage, etc. will be compared with template scenarios. If possible, these scenarios (e.g. one person coming with luggage, dropping luggage, leaving) will be used to monitor the behavior of persons and for alarming.

The major difference between predefined alarms and recognized unusual behavior is that the first can be associated with a predefined, human-readable text, like "scream noise" or "person dropped luggage". In contrast, with unusual behavior only the involved symbols(s) can be identified (in the user interface), without further explanatory text associated (besides potential naming of the features detected as being out of normal).

Functionality 1: (Predefined) scenario recognition

This method takes the symbolic representation on the level of the modality related symbols as input. It uses a rule-base to combine these symbols in a hierarchical way to create symbols with higher (more abstract) semantic meaning out of symbols with lower semantic level. Predefined alarms are not flexible and thus cannot adapt to new situations. Whatever is recognized has to be built into the system before it becomes operative – as opposed to recognizing unusual behaviour, which learns during operation. The advantage of recognizing predefined alarms is the ability to provide a human user with a semantic description of the type of alarm that the system has detected (e.g. "Unattended Luggage" instead of "Unusual Behavior"), also for complex scenarios.

The predefined alarm scenarios that the system at the airport shall detect are:

- Unattended luggage
- Loitering person
- Car in parking area has exceeded maximum parking time
- Screaming person
- Gunfire
- Breaking glass

While the predefined alarms "Screaming person", "Gunfire" and "Breaking glass" merely rely on information available from low-level symbols of the audio modality and will be handed through to the User Notification Filter, the alarms "Unattended luggage" and "Loitering person" require a symbolic processing of different information sources as described in [10].

Unattended luggage

To detect unattended luggage, the system has to detect person objects and luggage objects. When these two objects can be reliably detected, the system can reason about the associations between person and luggage symbols. The first scenario is a luggage symbol, which cannot be assigned to a person reliably. If the system fails

to assign the luggage for a certain period, it assumes the luggage to be unattended. This scenario is a successful connection between a person and a piece of luggage.

In the second scenario, the system has to track both person and luggage and detect if the person has moved away from the luggage for a longer period.

It is expected that the first scenario yields more false alarms, since it relies only on the successful visual detection of luggage. The second scenario, however, is expected to be more reliable and furthermore possibly allows identification of the person who has left the piece of luggage.

Loitering person

Person symbols that can be successfully tracked for a longer period can raise an alarm for loitering if they remain in the same location for an extended period of time. The challenge here is to prove that loitering can be detected, even if the person shifts its position between the viewport of different cameras. Thus the system shall be able to detect a person that has been moving around the area for e.g. a whole day. Without inter-node communication, the system can be fooled by changing the position between different camera positions. By handing over identified persons between nodes, this should not be possible.

Functionality 2: Unusual behavior recognition

Probability of existence: This method calculates the probability of a symbol with respect to its position within the sensor's view, its direction of movement, and its speed. The result will be Probability Mountains over the mentioned parameters.

Probability of duration: In addition to the probability of the simple existence of symbols, this task measures the duration of existence of symbols within some range. The range can be composed of parts of the sensor's view up to the view across several sensors.

Probability of movement: The probability of the movement of a HLS within the node's view (e.g. is it usual that a person coming from one trajectory moves to another one?) and across nodes: Given an instance that is observed at time t, the probability that this instance will follow the same path (or takes the same trajectory) as other instances should be calculated. Also a switch from one trajectory to another should be handled as an alarm.

I. *User Notification Filter*

Description

This tier builds the interface of the high-level sensor processing. It delivers alarms to the user interface and can be asked about the status of a node or several nodes. It filters identical alarms, e.g. when the same lurking person is reported several times from the predefined scenario recognition, or a reported unusual behavior can be matched with a predefined alarm, only the latter will be delivered. Additionally, the user can apply filtering rules to omit or prolong alarms via the GUI.

Functionality

A basic filtering mechanism for avoiding sending the same alarm several times or sending a per-defined alarm and an unusual behavior for the same thing is applied. An additional rule-base with user preferences is also considered.

Note: although by means of LBP a global view will be established among the nodes and despite the filtering, it cannot be excluded that the GUI will receive identical alarms from different nodes.

IV. Detailed Description of Tracking of LLS

We are concerned with the problem of tracking multiple interacting targets. Our objective is to obtain a record of the trajectories of targets over time and to maintain correct, unique identification of each target.

Tracking multiple identical targets becomes challenging when the targets pass close to one another or merge as persons do in a crowd. In recent times, an approach that relies on the use of a motion model that is able to adequately describe target behavior throughout an interaction event was developed [2]. This approach has a motion model that reflects the additional complexity of the target behavior.

The reason for using this approach lies in the fact that the number of symbols in the observation model can change from sensor observation to sensor observation. E.g. if several persons are going through a corridor, the visual feature extraction algorithms might detect a satisfactory number of persons in one image and just a group of persons in the consecutive one. In case of unlucky conditions, the detection can change often within short periods of time for the same physical object.

The multiple target tracking problem can be expressed as a Bayesian filter. We recursively update the posterior distribution $P(X_t \mid Z^t)$ over the joint state of all n targets $\{X_{it} \mid i \in 1..n\}$ given all observations $Z^t = Z_1..Z_t$ up to and including time t, according to:

$$P(X_t \mid Z^t) = kP(Z_t \mid X_t)\int_{X_{t-1}} P(X_t \mid X_{t-1})P(X_{t-1} \mid Z^{t-1})$$

The likelihood $P(Z_t \mid X_t)$ expresses the *measurement model*, the probability we observed the measurement Z_t given the state X_t at time t, which is a model for the modality-related feature extraction algorithms. The *motion model* $P(X_t \mid X_{t-1})$ predicts the state X_t at time t given the previous state X_{t-1}. In all that follows we will assume that the likelihood $P(Z_t \mid X_t)$ factors across targets as

$$P(Z_t \mid X_t) = \prod_{i=1}^{n} P(Z_{it} \mid X_{it})$$

and that the appearances of targets are conditionally independent, which may not be completely true in case of people, who belong together, but will hold most of the time between all persons in the crowd.

If we assume the targets as being independent, or non-interacting, they can be tracked with single-target particle

filters. In other words, the motion model is factored in a product of motion models for individual targets

$$P(X_t \mid X_{t-1}) = \prod_i P(X_{it} \mid X_{i,t-1}).$$

The task is to approximate the posterior $P(X_{it} \mid Z^t)$ over each target's state X_{it}. In other words, the probability that the current observations are made, because the observed objects behave in a particular way.

One view on particle filters is to see them as importance filter for this posterior. Therefore, we assume the posterior of the previous time being approximated by a set of weighted particles,

$$P(X_{it} \mid Z^{t-1}) \approx \left\{X_{i,t-1}^{(r)}, \pi_{i,t-1}^{(r)}\right\}_{r=1}^{N}.$$

Then, for the current time step, we draw N samples $X_{it}^{(s)}$ from a proposal distribution

$$X_{it}^{(s)} \sim q(X_{it}) = \sum_r \pi_{i,t-1}^{(r)} P(X_{it} \mid X_{i,t-1}^{(r)})$$

which is a mixture of motion models $P(X_{it} \mid X_{i,t-1}^{(r)})$. Finally, we weight each sample by its likelihood. The resulting set $\{X_{it}^{(s)}, \pi_{it}^{(s)} = P(Z_{it} \mid X_{it}^{(s)})\}_{s=1}^{N}$ is a weighted approximation for the posterior over the target's state X_{it} at time t.

The MRF-based approach for the motion model uses pair wise MRFs, where the $\psi(X_{it}, X_{jt})$ are pair wise interaction potentials (along edges) and E the space of edges between objects:

$$P(X_t \mid X_{t-1}) \propto \prod_i P(X_{it} \mid X_{i(t-1)}) \prod_{i,j \in E} \psi(X_{it}, X_{jt})$$

Each two objects share a particular potential. This term can be incorporated into the Bayesian filter easily, but now we approximate the joint state of all targets, which would result in the necessity of drawing an incredible high number of particles to be able to find a good approximation.

Applying the Monte Carlo approximation to the original Bayesian posterior, we get

$$P(X_t \mid Z^t) \approx kP(Z_t \mid X_t) \sum_r \pi_{t-1}^{(r)} P(X_t \mid X_{t-1}^{(r)}),$$

and incorporating the MRF motion model, we obtain

$$P(X_t \mid Z^t) \approx kP(Z_t \mid X_t) \prod_{i,j \in E} \psi(X_{it}, X_{jt}) \sum_r \pi_{t-1}^{(r)} \prod_i P(X_{it} \mid X_{i,t-1}^{(r)})$$

Fortunately, we see that the interaction potential is independent on earlier states, so it can be treated as additional factor. Unfortunately, approximating this term approximates the joint position of all targets, which is not our scope. Therefore, we apply MCMC (Markov chain Monte Carlo) sampling, so that the stationary distribution of the chain is exactly the target distribution, and we change only the state of one target at a time by sampling directly from the motion model of that target

$$Q(X_t' \mid X_t) = \frac{1}{N} Q(X_t' \mid X_t, i) = \frac{1}{N} \sum_r P(X_{it}'' \mid X_{i,t-1}^{(r)}) \prod_{j \neq i} \delta(X_{jt}' = X_j)$$

The acceptance ratio of this sampling method is only

$$a_S = \min\left(1, \frac{P(Z_t \mid X_{it}') \prod_{i,j \in E} \psi(X_{it}', X_{jt}')}{P(Z_t \mid X_{it}) \prod_{i,j \in E} \psi(X_{it}, X_{jt})}\right).$$

This method minimizes the computational effort in comparison to joint particle filters for tracking of multiple objects and also minimizes the fault detection rate compared to a set of single-object-tracking particle filters.

V. Conclusion and Outlook

This paper presents a hierarchical processing architecture for smart sensor networks. The innovative aspect lies in the step-by-step processing, through which from the low-level symbols there emerges information being more and more meaningful to a human person in charge. Expected results are described for the deployment in an airport environment.

All layers of the hierarchical processing framework are described in order to understand the idea behind and the algorithm for tracking of multiple objects is described in more detail.

Iterative tests and interviews with the airport staff are planned to gain measures for evaluating the results.

References

[1] www.sense-ist.org (SENSE project website)

[2] Z. Khan, T. Balch, and F. Dellaert: "An MCMC-based Particle Filter for Tracking Multiple Interacting Targets", IEEE Transactions on Pattern Analysis and Machine Intelligence, 2006

[3] Bishop, C. M.: *Neural Networks for Pattern Recognition.* New York NY.: Oxford University Press Inc., 1995

[4] G. Zucker (ne Pratl), and L. Frangu: "Smart Nodes for Semantic Analysis of Visual and Aural Data", Proceedings of the IEEE INDIN, p. 1001-1006, 2007

[5] B. Sallans, D. Bruckner, and G. Russ: "Statistical Detection of Alarm Conditions in Building Automation Systems". In: Proceedings of 2006 IEEE INDIN, S. 6, 2006

[6] C. Crick and A. Pfeffer: "Loopy belief propagation as a basis for communication in sensor networks", In Proceedings of Uncertainty in Artificial Intelligence (UAI), 2003

[7] J.S. Yedidia, W.T. Freeman, and Y. Weiss: "Understanding Belief Propagation and Its Generalizations", IJCAI 2001

[8] D.-S. Lee: "Online adaptive gaussian mixture learning for video. applications," in ECCV 2004 Workshop on Statistical for Video Processing

[9] N. Vlassis, and A. Likas: "A Greedy EM Algorithm for Gaussian Mixture Learning", Neural Processing Letters, Volume 15, Number 1, 2002

[10] Z. Ghahramani, and M. J. Beal: "Variational Inference for Bayesian Mixtures of Factor Analysers", Advances in Neural Information Processing Systems. 2000, vol. 12, MIT Press

[11] N. Ueda, R. Nakano, Z. Ghahramani, and G. E. Hinton: "SMEM Algorithm for Mixture Models", Neural Computation archive Volume 12

[12] Z. Ghahramani, and G. E. Hinton: "The EM Algorithm for Mixture of Factor Analyzers", Technical Report CRG-TR-96-1, Department of Computer Science, University of Toronto, 1996.

[13] P. Lombardi: "A study on data fusion techniques for visual modules", Technical. report, University of Pavia, 2002

[14] W. Burgstaller: "Interpretation of Situations in Buildings", Dissertation thesis, Vienna University of Technology, 2007

[15] R. Kindermann, and J. L. Snell: "Markov Random Fields and Their Applications", AMS Books Online, ISBN: 0-8218-3381-2, www.ams.org/online_bks/conm1