Cognitive Visual Tracking and Camera Control

Nicola Bellotto[c], Ben Benfold[a], Hanno Harland[b], Hans-Hellmut Nagel[b], Nicola Pirlo[b], Ian Reid[a], Eric Sommerlade[a], Chuan Zhao[a]

[a]*Dept of Engineering Science, University of Oxford, UK*
[b]*Institut für Algorithmen und Kognitive Systeme, Karlsruhe Institute of Technology, Germany*
[c]*School of Computer Science, University of Lincoln, UK*

Abstract

Cognitive visual tracking is the process of observing and understanding the behaviour of a moving person. This paper presents an efficient solution to extract, in real-time, high-level information from an observed scene, and generate the most appropriate commands for a set of pan-tilt-zoom (PTZ) cameras in a surveillance scenario. Such a high-level feedback control loop, which is the main novelty of our work, will serve to reduce uncertainties in the observed scene and to maximize the amount of information extracted from it. It is implemented with a distributed camera system using SQL tables as virtual communication channels, and Situation Graph Trees for knowledge representation, inference and high-level camera control. A set of experiments in a surveillance scenario show the effectiveness of our approach and its potential for real applications of cognitive vision.

Key words: cognitive vision, human tracking, active cameras

1. Introduction

We are motivated by the desire to develop cognitive visual systems. In the context of surveillance, the theme of this paper, by cognitive we mean a system which can not only track targets, but identify them, and explain what is taking place, especially taking account of any possible causal relationships.

This paper describes a system which supports work towards this goal. In a previous paper [1] we have considered the issues of low-level data acquisition processes, and how these processes communicate. In the current paper we are more concerned with augmenting this architecture with contextual

information to enable top-down processing. In particular we aim not only to generate inference from low-level visual primitives, but to use this inference to influence decisions about *active* visual sensing in a dynamic environment.

A typical application of an intelligent surveillance system would be to keep track of all targets in a scene and acquire high-resolution images of their faces for identification. In [1], in common with many other similar implementations, the system attempted to satisfy the first part of this goal using two simple, hard-coded rules: if a new target was observed by an overhead static camera, a PTZ device was immediately dispatched to acquire and track it autonomously at a pre-specified zoom level. However our objective in the current work is to *couple high-level inference to sensing strategies*. Prior domain knowledge is captured via fuzzy metric-temporal logic rules, and these are unified with instantaneous knowledge acquired to generate scene situation descriptions using the inference engine developed by [4]. We use this current state of the world, as determined by the inference, to help decide what the next sensing action should be, thereby completing a high-level sensing-perception-action control loop. Thus, in contrast to most previous work, we are concerned in this paper with using inference to *generate effective camera actions* and, by means of these, to influence future inference outcomes. This is a major contribution to the current state-of-the-art in active vision and intelligent surveillance. To present and demonstrate a limited but fully operational multi-camera system with high-level active sensing is also an important contribution of our work.

The paper is organized as follows: Section 2 presents relevant work in cognitive computer vision and camera control. System architecture and algorithms for low-level visual processing are introduced in Section 3. Section 4 illustrates the solutions implemented for on-line inference and high-level camera control, while Section 5 describes their application to specific scenarios of visual surveillance. Several experiments are presented in Section 6. Conclusion and future work are finally discussed in Section 7.

2. Related Work

In view of the different nature of knowledge, a modular scheme to describe behaviour understanding framework was proposed by Kanade in [5]. Our own interpretation of this is an architecture to perform human behaviour interpretation, drawing inspiration heavily from Kanade and Nagel [6] (see Fig. 1). At the bottom is the Sensor Actuator Level (SAL), which provides

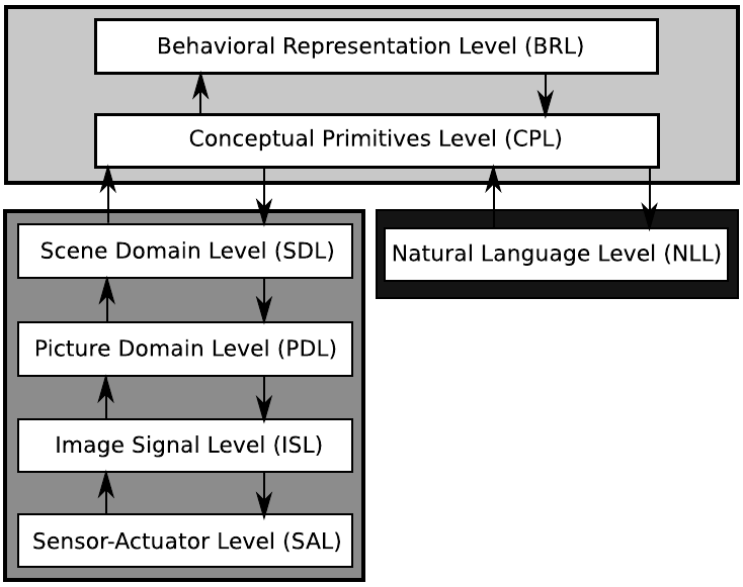

Figure 1: Modular scheme for behaviour interpretation [6].

image data and information about camera parameters. The above Image Signal Level (ISL) is where low-level image measurements are processed in order to extract features in each image. The following layer Picture Domain Level (PDL) is responsible for locating the moving blobs in 2-D representations. At the Scene Domain Level (SDL), models and algorithms are based on 3-D configurations of objects in the scene. The information obtained from the lower-levels is forwarded to the Conceptual Primitives Level (CPL) in order to determine tentative descriptions at each time step. At the Behavioural Representation Level (BRL), the object behaviour is modelled based on the tentative descriptions derived from CPL. Annotation of the behaviour in natural language is accomplished in the Natural Language Level (NLL).

Here we consider previous work on human activity analysis and behaviour understanding which fall within the top modules (conceptual and behavioural levels) of this architecture. We also review some of the previous camera control strategies.

2.1. Behaviour Interpretation

From the algorithmic point of view, most previous work has modelled human behaviour by considering one of two different approaches based on the direction of data flows: on the one hand, bottom-up approaches aggregate

low-level visual data into human behaviours using a set of motion patterns [7, 8, 9, 10]. These often incorporate aspects of machine learning, which makes them extendible, but usually only via the acquisition of a large corpus of training data. In contrast, top-down techniques, such as [11, 12, 6], predefine a set of motion patterns that represent the set of object behaviours to be instantiated. This kind of approach generally lacks a learning ability but does represent human behaviours in an intuitive and "human-readable" manner, which can be easily extended by expert knowledge. Additionally, they do not depend on the specific data from the training set.

Among the bottom-up approaches, Binford *et al.* [7] describe a solution using Bayesian belief networks to control inferences over a complex, multi-level representation system based on generalised cylinders. Intille and Bobick [8], instead, propose an automated annotation system to interpret sport scenes using belief networks. They exploit temporal structure descriptions to represent the logical and temporal relationships between agent goals. Belief networks are applied to represent and recognize individual agent goals from visual evidence. In [9], Remagnino *et al.* present an agent based surveillance system which supplies textual descriptions for the dynamic activity occurring in the 3D world to monitor scenes involving both pedestrians and vehicles. The descriptions are derived by means of dynamic and probabilistic inference based on geometric information provided by a tracking module. A behaviour agent is assigned to each tracking object, which uses a Bayesian network to infer the fundamental features of the objects' trajectory, and a situation agent is used to interpret pairwise interactions when two objects are close. Robertson and Reid developed a system in [10] for human behaviour recognition in video sequences by defining actions using feature vectors that comprise trajectory information and a set of local motion descriptors. Bayesian fusion of these feature vectors compose spatio-temporal actions, which were gathered to build up a database. Action recognition is achieved via probabilistic search in the database. HMMs which encode scene rules are used to smooth sequences of actions.

Turning to top-down solutions, Ayers *et al.* [11] describe a system which recognizes human actions in an environment with prior knowledge about the layout of the room. Based on low-level information obtained from three techniques, namely skin detection, tracking and scene change detection, action recognition is modelled by a state machine. Textual description of recognized actions is generated by reducing a video sequence into a smaller series of key frames. Kojima and Tamura address in [12] the generation of natural

language descriptions of human activities from video images. The proposed method generates textual description by extracting semantic features of human motions and associating them with concept hierarchy of actions. Nagel *et al.* [6] developed a system that makes use of so-called Situation Graph Trees (SGTs) as representation formalism for behavioural knowledge. They used it for on-line analysis and generation of natural language descriptions. In the present work we explore the use of this representation and inference mechanism as a means to control sensing actions.

2.2. Camera Control Strategies

Camera control can be related either to monocular or multi-camera configurations, and can be categorized in passive control and active control. Camera control is essential to perform navigation and monitoring tasks, as well as to obtain and keep the identities of tracked objects. The problem with general, unconstrained environments involving several objects is that information gained from fixed sensors is either too coarse or noisy to allow correct identification, and often too focused and narrow to keep track of all the objects, or to capture good identification features at the right time.

Here we categorize the strategies of camera control into three different models in terms of the attributes of the evidence the commands depends on: picture domain camera control (PDCC), scene domain camera control (SDCC) and conceptual level camera control (CLCC).

PDCC depends on control demands purely based on image information. It is the simplest approach to surveillance in which the goal is simply to track a hot-spot, i.e., selecting a localized interesting object (or group of objects) and keeping it centred in the field of view (FoV) of a PTZ camera, and/or preserve its size constant. This approach assumes that only one interesting object/group is in the FoV, whereas all the other blobs are either generated by noise or non-interesting objects. Examples for this kind of control are the works of Tordoff and Murray [13] and Denzler *et al.* [14]. Both address the choice of focal length for a single active camera when tracking a single target. The former tries to preserve the scale of the object, the latter chooses those observation parameters that yield a minimum uncertainty in a Kalman filter context. Another example is the person-following active camera system proposed by Prati *et al.* [2], where the camera control is implemented through a simple heuristic rule that re-centres the target in the FoV whenever it is close to the image's border.

Scene domain control is a more complex approach than PDCC. The surveillance goals are typically similar, but control demands are determined via 3D-scene information. The advantage of SDCC is that the 3D representation permits more flexible and reliable handling of complex appearance, disappearance, trajectory modelling and other target's interactions with the environment. Some examples are Tsuruoka *et al.* [15], who present a system that tracks a lecturer in a classroom using foreground segmentation on a fixed camera's image and a fuzzy control scheme to steer an active camera for close-up views. Another example is the "Face Cataloger" application presented by Hampapur *et al.* [16], which uses multiple active and passive cameras. The active ones are directed by simple geometric reasoning and static rules to acquire frontal face images. Recent work by Del Bimbo *et al.* [17] addresses the difficulty in positioning active cameras according to seemingly simple rules. Sommerlade and Reid [18] extend the approach of [14] to the scene domain, facilitating camera hand-off. In both these works however, the resulting control uses only the information about the position of targets and sensors.

Although much work remains to be done in terms of validation, control at the conceptual level (CLCC) is a high-level approach with the potential to meet the requirements of increasing complexity in surveillance tasks. A spatio-temporal description of the interesting object, with respect to other objects or to what is considered background, becomes necessary (e.g. a description of what kind of behaviour demands attention). An example is the home supervision project in [19], where the detected fall of a person results in the ringing of an alarm. Rules are encoded using conditions on spatio-temporal observations by human experts [20] and currently do not address any uncertainty in the observations.

Our work is in the same vein as the former in the sense of encoding expert knowledge, but, contrary to previous work, we also feed back the resulting inference into the data acquisition process: by appropriately controlling the active cameras, we are closing the perception-action-cycle using high-level reasoning.

3. Distributed Camera System and Visual Processing

In this section we introduce, and partly extend, the architecture described in [1], a schematic of which is illustrated in Fig. 2. The system comprises a set of distributed static and PTZ cameras with visual tracking algorithms,

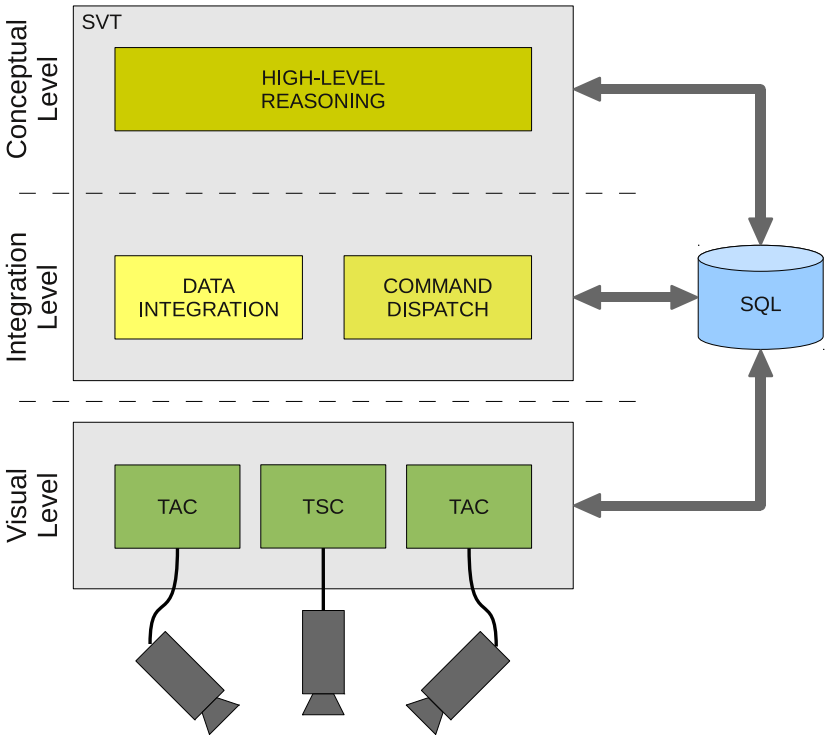

Figure 2: System architecture.

together with a central supervisor unit. The different modules in the figure are organized according to a hierarchical structure with a Visual Level at the bottom, a Conceptual Level at the top, and an Integration Level interfacing the two. These three levels are our own interpretation and simplification of the schematic in Fig. 1. The Visual Level contains all the hardware and software from picture domain level downward. The Conceptual Level, instead, covers the conceptual primitives and above. Finally, the Integration Level is our implementation of the scene domain, with additional modules for handling high-level commands. This is discussed further below and in Section 4.

In our current system, there are two active cameras (Sony DFW-VL500 with Directed Perception DTU-D46 pan-tilt unit) and an Axis 212 used as static camera. Each camera has a dedicated computer unit for visual processing and control, so that camera and unit, together, constitute an independent "tracker" module. The Tracker Static Camera (TSC) and the Tracker Active Cameras (TACs) are connected and synchronized with the Supervisor

Tracker (SVT) via TCP/IP.

The SVT includes three parallel processes – Data Integration, Command Dispatch and High-Level Reasoning, which all reside on a separate server machine. A key component of the system is the use of a central SQL database, which provides a reference application for the asynchronous communication between processes, as well as an efficient way to store large amounts of visual information and possible inference results.

Additional processes can similarly read from and write to the database, and they can be physically located on the SVT server or remote from it. We have implemented three such additional processes: a face recognition [21], a simple action classification [22] and a gaze direction detection [23]. Currently, only the face recognition, discussed in Section 3.5, is fully integrated into our system.

The remaing part of this section illustrates the SQL communication mechanism and the execution steps from target's detection with the static camera to actual tracking with the active camera. Further details on the system architecture can also be found in [1].

3.1. SQL Communication

Asynchronous inter-process communications and archiving of data are achieved in a simple and effective way via a central repository, implemented using an SQL database. Visual tracking data from static views are stored dynamically into tables of the database via client calls to the SQL server. The SVT determines if active zoom cameras should be dispatched to observe a particular target, and this message is effected via writing demands into another database table.

The SQL server includes the following tables:

- Calibration Table to store calibration information for all the trackers;
- Observation Table for observations/estimates generated by trackers and SVT;
- Command Table for high-level commands sent by the SVT;
- Image Table for stabilized images produced by the TACs.

Additional tables for inter-process communication within the SVT are introduced and discussed later in Section 4.

The basic scheme of communication among trackers, database and supervisor is illustrated in Fig. 3:

1. first, in an offline and one-off procedure, the cameras retrieve the calibration data stored in a common repository (i.e. Calibration Table), consequently used to convert pixel locations to world coordinates;
2. a background subtraction algorithm, described in Section 3.2, is used to detect potential human targets with the TSC in form of regions of interest (ROI). Using the previous calibration data, the absolute 2D position of these targets is computed and stored in the Observation Table;
3. the SVT reads the Observation Table, resolves the data association and computes a 3D estimate for every target's trajectory. These estimates are processed to generate camera commands sent via the Command Table, as described in Section 3.3.
4. The TAC receives commands from the Command Table and computes an approximate configuration of pan-tilt-zoom to predict the position of the target and initialize the tracking (see Section 3.4). Stabilized face images of the target are finally stored in the Image Table for archiving or identification purposes.

3.2. Detection with Static Camera

The static camera of our system (TSC) is used for real-time human detection on wide-angle image sequences. To detect people, we use an implementation of the Lehigh Omnidirectional Tracking System (LOTS) algorithm [24]. This is based on a background subtraction technique that uses two grey-scale images of the background and two per-pixel thresholds. The latter treat each pixel differently, allowing the detector to be robust to localized noise in low-size image regions, and evolve according to a pixel label provided by a light version of the traditional connected component analysis. Small adjacent regions detected by LOTS are clustered together, and their centroid calculated to give the target position. The background subtraction well suits our current installation of the static camera because the targets, being observed from the top, do not overlap in this case (see Fig. 4). Details of the original LOTS algorithm can be found in [25].

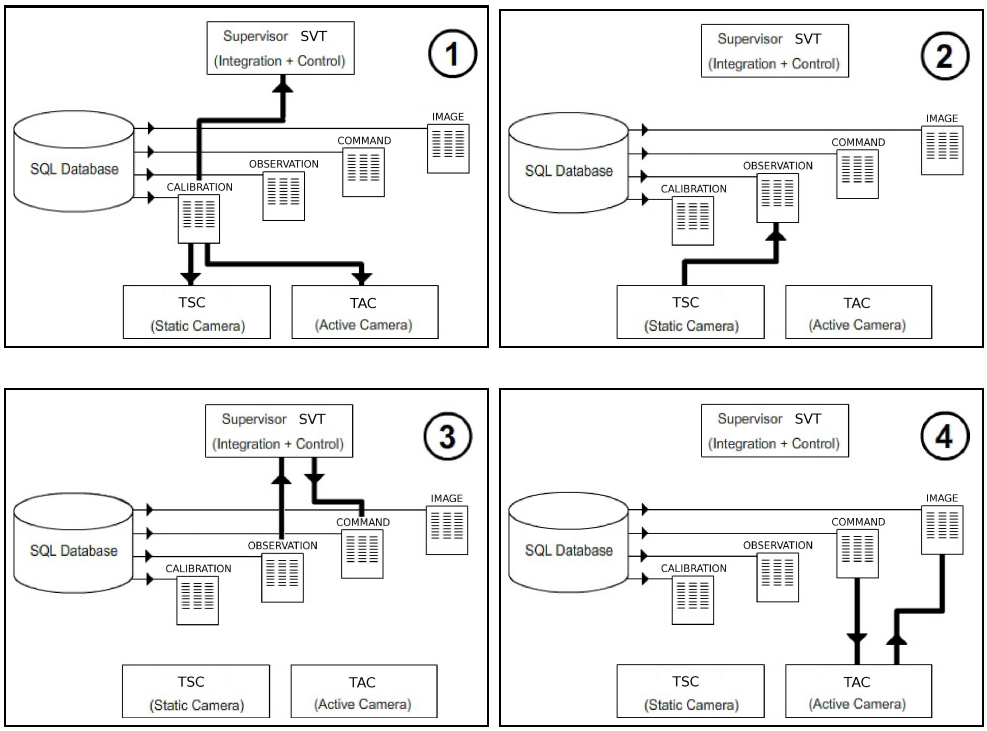

Figure 3: Communication steps between TSC, SVT and TAC to get stabilized images of a target.

3.3. Integration and Control with the Supervisor Tracker

The SVT is responsible for the data fusion, reasoning and camera control strategy. As anticipated in Fig. 2, this can be thought as a set of semi-independent processes using bi-directional communication, via SQL queries, for information exchange.

The main purpose of the Data Integration module is to collect sensor observations from one or more cameras and generate proper trajectories of the current targets, which can be used for reasoning and active tracking. This module uses an efficient multi-target tracker based on Kalman filters with a constant-velocity model and nearest-neighbour data association [26]. It reads the targets' coordinates from the Observation Table, which are provided by the static camera, and computes the 3D estimates (position and velocity) of the targets' heads, assuming they all have the same height. These estimates, then, are written back to the same table. In this way, it is possible to identify the trajectory of a target, as a sequence of estimates with a unique ID, from

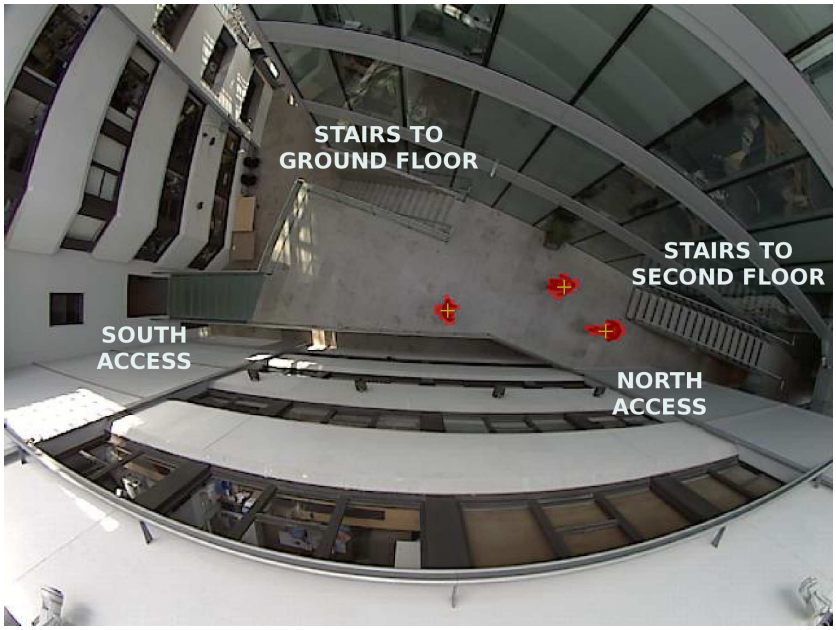

Figure 4: People detection with the TSC using LOTS background subtraction. The centroid of each red cluster, corresponding to a human target, is marked with a cross.

the anonymous and noisy detections provided by the static camera.

The High-Level Reasoning of the SVT contains a representation of the knowledge upon which on-line inference is performed, thus to provide a conceptual interpretation of the visual scene and generate opportune high-level commands for the cameras, such as "track target" or "acquire face image". The Command Dispatch module is responsible for sorting and delivering these commands, which are sent to the destination cameras through the Command Table of the database together with any requisite argument, like camera number or target ID. This part is discussed further in Section 4.

3.4. Tracking with Active Camera

The control command sent to a client with an active camera (TAC) comprises a target identification number together with a position and uncertainty on the ground plane common to the system. Combined with the expected height of targets, this is turned into a bounding box in three dimensions. The active camera is then steered using positional control to centre the projection of this area in the camera's view. Once the camera is sufficiently close, a standard face detection method [27] is run on the observed area. If a face

is detected, the client switches to velocity control mode and tracks the target using a visual tracking method based on level sets [28]. The focal length for the observation of the target is directly obtained from the perspective projection equation [3] and the requirement that the projection of the target should cover a fixed fraction of the image. This yields the appropriate zoom setting, as the cameras are intrinsically calibrated with a cubic mapping of zoom setting to focal length. The client transmits the stabilised face images from the visual tracking algorithm to the database, and stops tracking the target upon receipt of a tracking command where the target ID differs from the current one.

3.5. Face Recognition

Face images obtained by the TAC are used to determine the target's identity. For this purpose, we use the original face recognition algorithm proposed by Everingham *et al.* [29] and extended by Apostoloff *et al.* in [21]. A pictorial model is used to compute the locations of facial features, which form a descriptor of the person's face. When sufficient samples are collected, identification is performed using a random-ferns classifier by marginalising over the facial features. This confers robustness to localisation errors and occlusions, while enabling a real-time search of the face database.

Unfortunately, many of the face images sent by TACs are unsuitable for recognition purposes because they are not frontal views, or because they are affected by motion blur caused by pan-tilt movements, zoom changes and human motion. A mechanism for quality assessment is therefore necessary in order to choose only the best face images in a sequence. In our implementation, a pre-filter is integrated into the face recognizer to select the best images before performing any further analysis. This pre-filter is designed to be computationally inexpensive, still keeping high reliability, and is constituted by two parts: a simple face detection and a blur detection. The first one is based on the OpenCV implementation of the Viola-Jones algorithm [27], discarding face images which are too small (less than 20 x 20 pixel) or not sufficiently aligned. The second one makes use of the solution proposed in [30] to measure the sharpness or blurriness of an image and detect the best face among a pool of snapshots.

4. High-Level Reasoning and Camera Control

In intelligent surveillance, cameras must be appropriately tasked to collect and communicate the information relevant to the problem at hand. The level of intelligence of the system depends on the camera control strategy to some extent. Apart from hardware mechanical problems, the greatest challenge in setting up an automatic control system is a non-trivial trade-off between conflicting sub-goals, such as cost of commissioning a surveillance camera, flexibility in which object should be supervised, maximizing the information gain and reducing the risk of losing a supervised object.

We apply conceptual level camera control (CLCC) as outlined in Section 2.2. Compared to the approaches using information from picture and scene domain only, CLCC uses behaviour information as evidence, which is built based upon some human-like understanding and can be statistically more robust and stable than quantitative data such as object positions and pose. This formulation also allows us to formalize the background information in a consistent way and import it as logic facts. This *a priori* knowledge is easily integrated into CLCC utilizing the same logic formalization, and can be taken into account as auxiliary information.

A significant step towards this goal, which is also an important distinction between our previous work [1] and the current paper, is the addition of an inference engine for High-Level Reasoning that combines situational *a priori* knowledge with visual predicates. Besides knowledge representation and inference, the High-Level Reasoning is also responsible for CLCC. As shown in Fig. 5, it resides on the Conceptual Level of the SVT and communicates with the other modules at Integration Level via SQL tables.

In particular, quantitative data, generated at Visual Level and fused by the Data Integration module, is stored in the Observation Table and then converted into a list of (qualitative) predicates within the High-Level Reasoning. An inference thread is invoked on the latter whenever new visual predicates are available. The results of the inference are written to an Inference Table, including possible high-level commands, which are processed by the Command Dispatch and delivered to TACs. The Inference Table is also used to control additional modules external to the SVT, like the Face Recognition, the use of which will be discussed later in Section 5.4.

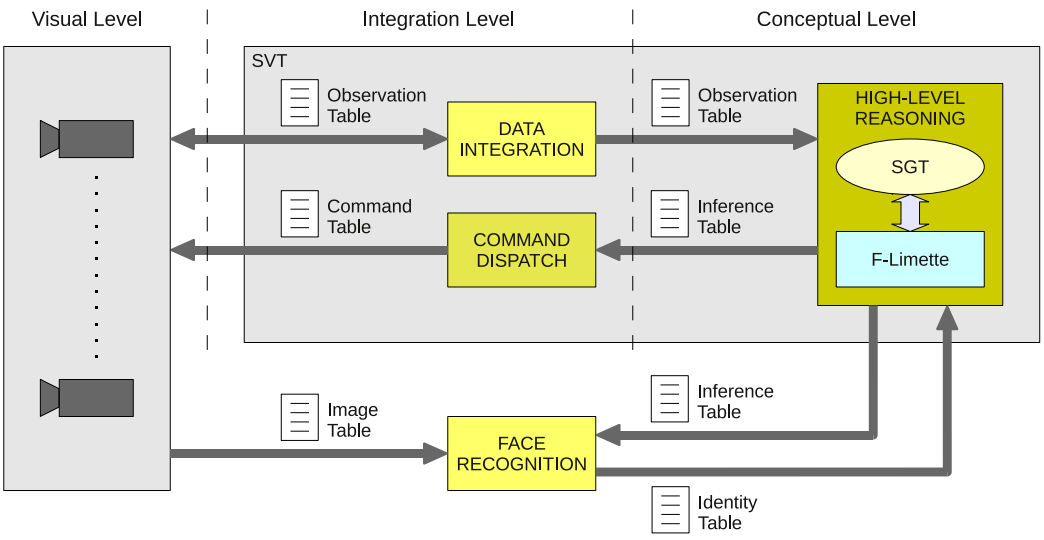

Figure 5: System architecture as in Fig. 2, but here showing details of the virtual communication channels between modules, implemented via SQL tables.

4.1. Inference on Situation Graph Trees

The High-Level Reasoning provides *a priori* knowledge about the expected behaviour of agents in the field of view of the recording camera(s). "Behaviour", in this context, is to be understood as a sequence of situated actions of an agent. This schematic *a priori* knowledge is structured into a Situation Graph Tree (SGT) which is specified in a formal language, SIT++, based on a Fuzzy Metric-Temporal Horn Logic (FMTHL) [4].

An SGT comprises a hierarchy of temporal fuzzy-rules to describe a situation in terms that are as specific as possible for any given time. Each node represents a possible temporal decomposition of events and rules that is more specific than the parent. Since there may be more than one specialisation, the hierarchy of rules naturally forms a tree, where each node comprises a set of fuzzy rules specialising the parent. Examples are shown in Fig. 7 and Fig. 8, where the inheritance relation is denoted with bold-face arrows. The rules have temporal ordering as indicated by thin lines/arrows. Each temporal situation has situation identifier, a set of predicates to be satisfied, and optionally, an action to be performed when the predicates are satisfied.

In [6], a conceptual representation for the instantaneous behaviour of a particular agent, in a recorded video, is given by the instantiation of one of the

schematic behaviours encoded by an SGT. Such an instantiation is obtained by unifying the rules encoded in the tree with visual predicates provided by the Computer Vision Subsystem. In other words, the quantitative geometric results obtained by the latter provide the "individuals" (in the sense of a formal logic model) required to instantiate one of the schematic behaviours encoded in an SGT. The instantiation of a conceptual representation for an agent's behaviour, during a particular time interval, is the result of a process termed "SGT-traversal" (or simply traversal). This is given in the form of a sequence of time-indexed predicates, generated using an inference engine for FMTHL, called *F-Limette* [31]. Generally speaking, the more specific the instantiation, the better the description, and so the traversal proceeds by depth-first search. Less formally, inference proceeds by unifying fuzzy predicates produced by the low-level vision processes, with the "rules of engagement" encoded in the SGT.

4.2. Conceptual Level Camera Control

The previous applications of SGT-based inference have all been based on recorded videos with constant parameters, and were used to create off-line descriptions of observed agent in the scene [32, 33]. Our key contribution is to extend this paradigm by employing it in an active system for PTZ camera control. The strategic camera control, or CLCC, is embedded in the cognitive reasoning; camera commands which suit a relevant situation are specified in the action part of the situation scheme of the SGT. Instead of using traditional camera-centric commands such as "pan sensor A to direction X", we seek to issue high-level task commands for camera action, such as "track current target with best camera", when the situation associating with it is instantiated. Therefore, the control decisions are made based on the reasoning conclusions of agent behaviours and situations. The high-level commands must be then decomposed into a sequence of low-level demands issued to the appropriate sensors at the correct times (e.g. 30Hz velocity-control demands for closed-loop tracking).

Some kind of *a priori* knowledge should be incorporated in order to correct incompleteness of the available geometric data from tracking. The *a priori* knowledge sources are usually embedded in scene-dependent models. As a result, it is possible to free generic motion analysis methods from the specific scene where they are applied. Such knowledge is assumed to be available to the system, including:

1. camera models with internal and external calibration settings to determine the best acquisition configuration;
2. scene models to help with generation of agent estimates and prediction of occlusions due to static components;
3. agent motion models to predict movements of agents (despite occlusions).

In our system, the knowledge of camera models is kept and used locally by the low-level control units associated with the PTZ cameras (i.e. TACs in Fig. 2). These must "compile" requests generated from the Conceptual Level (i.e. High-Level Reasoning) into a sequence of sensor specifications and commands. The compilation mainly involves commanding the PTZ camera to look at the appropriate scene locations at the right times. Sensor-specific pan, tilt and zoom demands, associated with a particular object or ROI, are computed by the TAC using the known calibration parameters of the sensor and the known geometry of the object and the scene. The scene and agent motion models are established using conceptual predicates, which will be addressed in Section 5, and are exploited for behaviour reasoning and tracking.

5. Application to Cognitive Visual Tracking

As explained before, our current goal is to couple high-level inference to sensing strategies, i.e. to control a set of active cameras for surveillance in an intelligent fashion. This is done in practice using the actions emitted during the traversal of an SGT, which has been specifically designed for the particular scenario. The following sections illustrate the conceptual model of the area under surveillance and the incremental design of an SGT for human behaviour interpretation and intelligent camera control.

5.1. Conceptual Model of the Scene

Behaviour analysis requires an explicit reference to the spatial context, i.e. a conceptual model of the scene. The area under surveillance has been therefore divided into semantically distinct regions, as shown in Fig. 6. Thanks to this model it is possible to infer the relationship of an agent with respect to (predefined) static objects in the scene, and to associate facts to specific locations within it.

The scene illustrated in Fig. 6 is divided into polygonally bounded segments, describing the possible positions in which an agent can be found.

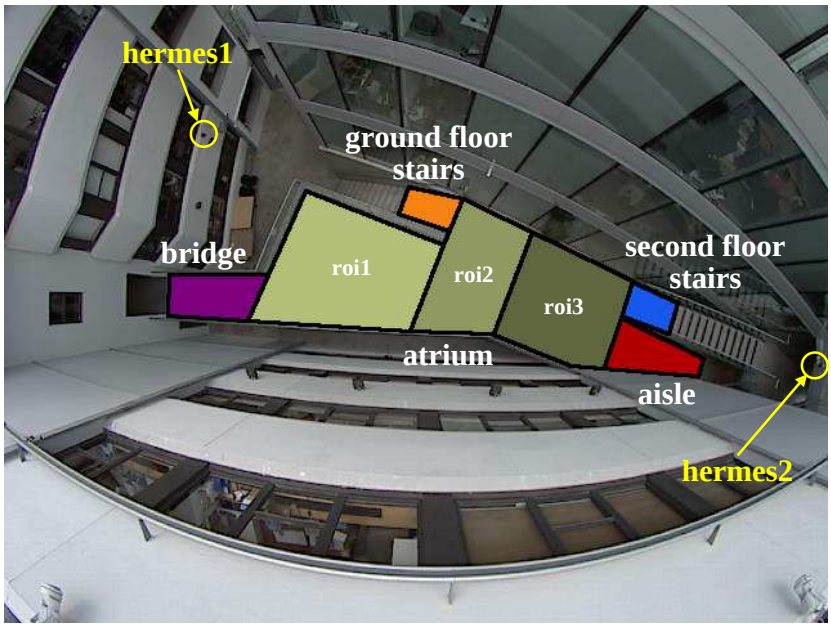

Figure 6: Atrium and floor-plan of the surveilled area with semantically distinct regions used for inference and action selection. *hermes1* and *hermes2* are the two active cameras (TACs).

Each segment has a label to determine the conceptual description associated with it. We distinguish two different types of segments, namely:

- **exit** segment, which includes:
  - *bridge*: connects the atrium to another building;
  - *ground floor stairs*: connects the atrium, which is on the first floor, to the the ground floor;
  - *second floor stairs*: connects the atrium to the second floor;
  - *aisle*: leading to other rooms;
- **atrium** segment, which includes:
  - *roi-i*: the *i*-th ROI placed on the main activity area of the atrium.

Note that, in the current implementation, the only purpose of the three ROIs is to simplify the atrium's representation with the inference engine *F-Limette*. We can now build predicates which relate to the spatial position of the agent with respect to the above segments. This is described in the next section.

5.2. Human Behaviour Interpretation

In general, the term "human behaviour" refers to the combination of one or more human actions with the knowledge about its environment, thus to derive a semantic description about the observed agent. In this paper we consider just a small subset of possible behaviours, which are mainly built around the "walking" action of individuals within the surveilled area. We are interested in particular on the interpretation of motion behaviours, such as entering, crossing or leaving the area by a single agent.

Fig. 7 depicts the SGT designed for the initial scenario. The root graph, on the specialization Layer 1, comprises only one situation scheme, in which the predicate `active(Agent)` states that an agent is present in the scene. In the language of logic programming, this unifies or binds the variable `Agent` (variables always start with a capital letter) with a particular target ID from the Observation Table.

The initial scheme is specialized by another situation graph in Layer 2, comprising two schemes both of which can be either the start or the end of the current situation. They indicate that the agent may be either on the first floor of the atrium (predicate `on(Agent, first_floor)`) or somewhere else, and may move between the two locations (as illustrated by the thin double arrow). However, it may not be simultaneously on the first floor and on another location. Note that the second scheme (predicate `not_on(Agent, first_floor)`) is instantiated when the agent is detected by the TSC outside the first floor. Indeed, we assume the latter is the ground plane and ignore any activity outside it.

The first scheme is particularized further to describe the behaviours of the agent on the first floor with two situation graphs in Layer 3, depending on where the agent is detected:

- `on_atrium(Agent)` is satisfied as the position of the agent, projected on the ground plane, is inside the atrium's area. The following specialization in Layer 4 (blue extension of the SGT in Fig. 7) is used for camera control and is discussed later in Section 5.3.

- `on_exit(Agent, Exit)` describes the situation when the agent is located in one of the exit segments (*bridge*, *ground floor stairs*, *aisle* or *second floor stairs*). It is followed by two specialized situation graphs in Layer 4, each containing a single scheme:

Figure 7: SGT for human behaviour interpretation, including an extension (Layer 4, left-hand side) for active camera selection.

- `from(Agent, Exit)`, in the first situation, represents the agent *entering* the atrium from an exit segment;
- `via(Agent, Exit)`, in the second situation, describes instead the case of an agent *leaving* the atrium through an exit.

As noted before, whenever a node in the SGT is satisfied during the traversal, it can optionally emit an action. In previous work [32, 33], the typical application of this ability was the use of the `note(...)` action, which prints a particular state to monitor or log the inference process, or for natural language text-generation. Within our SGT, the `note(...)` action is extended to form a new one, called `sts(...)`, that records the current *status* of the

traversal in the Inference Table of the SQL database. For example, in case of `agent_12` leaving the atrium through the bridge, the SGT-traversal would result in an action `sts(leaving_via(agent_12, bridge))` written on the Inference Table.

5.3. Multi-Camera Selection

In the previous section we have described a simple SGT which is used to make inference about where an agent is in terms of qualitative labels associated with the semantically different regions in the scene. In this part we extend that to include the direction of motion, and we use it to invoke a camera-select action, so that closed-loop tracking is effected in the camera most likely to obtain frontal views of the agent. The SGT is the one illustrated in Fig. 7, including the additional specialization in Layer 4 (blue extension on the left-hand side).

Remember that the first situation in Layer 3 refers to the case when the agent is walking in the atrium. The specialization that follows in Layer 4 seeks to determine its direction (predicate `towards(Agent, Exit)`) and the camera it is facing to (predicate `side_of(Exit, Camera)`). Here the system tries to unify the variables `Camera` and `Exit` for a consistent interpretation with the current binding of `Agent`, resulting in selection of the appropriate camera in the node's actions (i.e. a camera the moving agent is facing to). The specialisation of the second situation on Layer 3, instead, is concerned with the case that the person is either just leaving or entering the first floor of the atrium, in which case no TAC is dispatched.

Besides writing the current status, the SGT-traversal can now generate a special `cmd(...)` action that writes a *command* string in the Inference Table. In this case, `cmd(follow(Camera, Agent))` tells one of the TACs (*hermes1* or *hermes2* in Fig. 6) to follow the agent. This string is interpreted by the Command Dispatch module of the SVT and sent to the relative camera via the Command Table. The `follow` command, in particular, uses position-based open-loop control of the active camera based on 3D estimates from the Data Integration and continuous demands from the TSC. During its execution, the zoom is minimum (i.e. maximum FoV).

While this may appear to be overkill for a relatively simple action rule, we argue that the general formalism available through such knowledge conceptualization and inference will in the future enable much more complex inference and corresponding actions. Furthermore, unlike this simple example in which the action is the end in itself, we can instead use it as a means

to an end, namely to effect further actions to explore aspects of the scene actively. The next example below deals with this in regard to face acquisition and recognition.

5.4. Automatic Zoom for Human Identification

In the example above the action emitted is an end in itself. That is, the "follow" behaviour of the TAC, resulting from the traversal of the previous SGT, does not influence future inference outcomes. Much more interesting is the case where the result of an action has direct consequences on the next traversal; the aim of such an action will be to acquire new information to enable a deeper instantiation in the SGT. Such a situation provides a connection between high-level inference and sensing actions, in which the high-level knowledge has contributed to the low-level sensing, and vice-versa. This has been a key goal of our current research.

In this section we develop an SGT for the case of an agent moving through the atrium in which the goal is to obtain a close-up view of the face and to perform face recognition. To this end, we have integrated the real-time face recognition system in Section 3.5 as a visual processing module. Like all the other processes, the Face Recognition simply reads and writes data from/to the database via the standard SQL protocol, meaning the physical location of the process is immaterial. Its operation within the system, and in particular the communication with the inference engine, is illustrated in Fig. 5.

The SGT for this scenario, which is shown in Fig. 8, is a further extension of the previous solution (differences highlighted in green on the figure). This new SGT performs both camera selection and "follow" behaviour; additionally, it can instruct a camera to "track" a target's face and "recognize" it. While the execution of actual `follow` command uses position-based open-loop control of the active camera based on continuous demands from the TSC, the `track` command results in closed-loop visual tracking independent of the TSC. In particular, it tells the TAC to locate and zoom on the target's face, sending a continuous stream of high-resolution images to the Image Table. A `recognize` command would then activate the Face Recognition module, reading images from the Image Table and sending the result to the High-Level Reasoning through the Identity Table. Once identification is complete, the camera is instructed, via the SGT, to zoom-out and follow the target in open-loop mode.

More specifically, we refer now to the left-hand branch of the tree in which a new agent has entered the scene. Upon reaching Layer 4 with the instan-

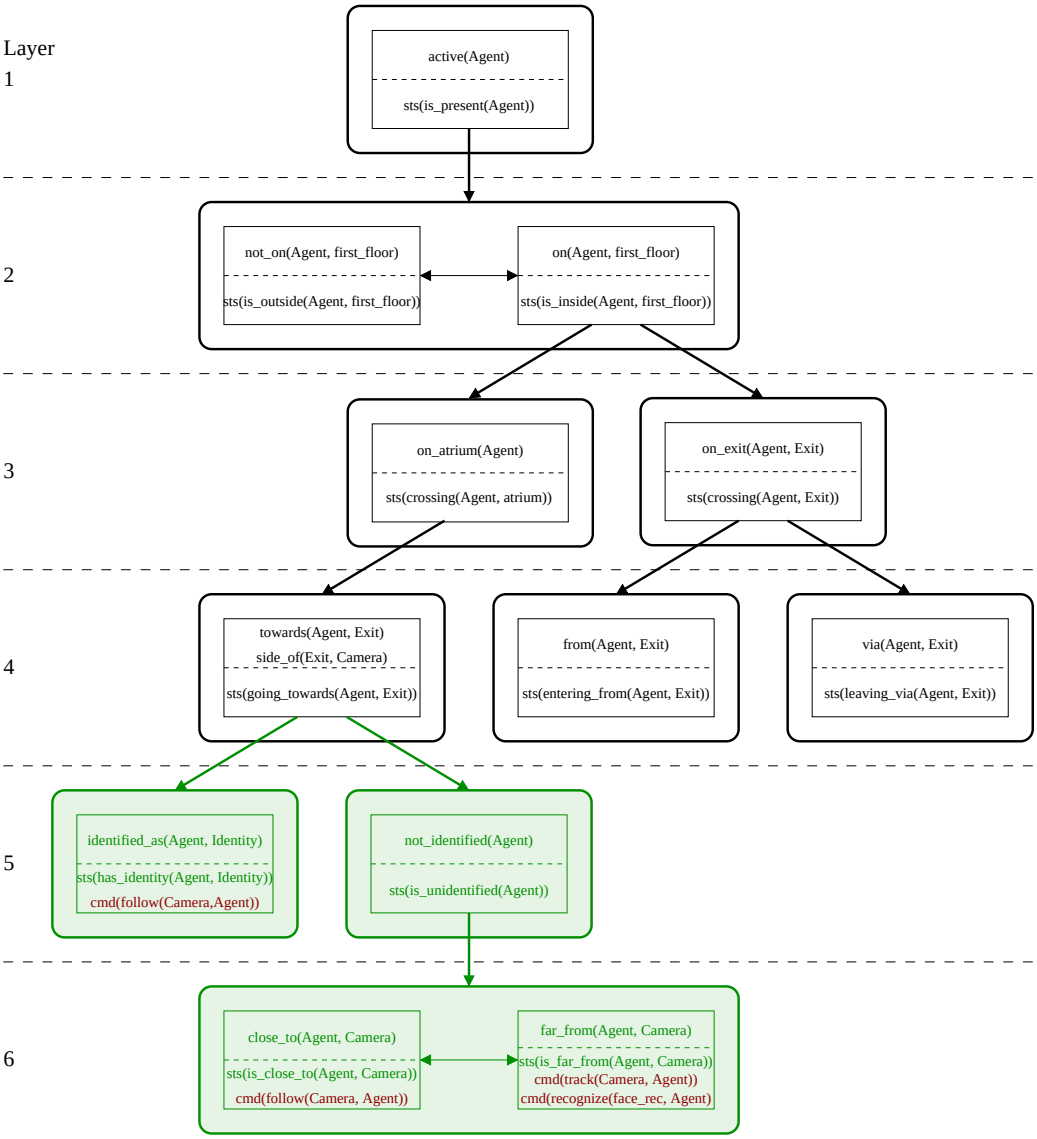

Figure 8: SGT for active camera selection and face recognition.

tiation of the `Agent` and `Camera` variables, the previous `follow` command is replaced by a new specialization inserted in Layer 5. This considers whether the target has been identified or not, with the instantiation of the following predicates:

- `not_identified(Agent)`, means the agents has not been identified yet,

so the traversal proceeds further to Layer 6. Here the situation is represented by the following two schemes:

  - `close_to(Agent, Camera)`, in which the target is on a region of the atrium too close to the current camera. In this case, the camera control cannot react quickly enough to zoom on the moving target and acquire high-resolution images of the face. A simple `follow` command is therefore dispatched.
  - `far_from(Agent, Camera)`, where the target is at optimal distance for closed-loop face tracking and recognition. This is accomplished by the current camera and the face recognition via the respective `track` and `recognize` commands. In particular, `cmd(track(Camera, Agent))` generates the first command for the selected camera, which zooms on the face and tracks it in closed-loop control; `cmd(recognize(face_rec, Agent)`, instead, invokes the Face Recognition module to identify the target.

- `identified_as(Agent, Identity)`, means the identity of the target is known, therefore a simple `follow` command is generated, which results in a zoom-out and open-loop control of the TAC.

Being the traversal a depth-first search, the delay introduced adding more layers to the SGT is negligible, at least in comparison to the typical 30fps of a camera and to the time needed for face recognition. Different would be the case, of course, in which the traversal was a breadth-first exploration of the SGT. Note also that the case of multiple targets has not been specifically addressed in the current research, although the SGT here presented yielded good results, to some extent, when applied to scenarios with more than one person. This is discussed further in our experiments.

6. Experimental Results

We have conducted numerous experiments with the overall system in [1], showing that our architecture is suitable for inter-communication, control and data storage of a distributed multi-camera system. Here we concentrate on the presentation of results pertaining to the inference mechanism. In particular, we consider applications of the SGTs in Section 5 to follow a target with the most appropriate camera for frontal-view observation, and to acquire high-resolution images for on-line identification.

6.1. Target Following

The first example shows the application of the SGT in Fig. 7 for multi-camera selection. When a pedestrian walks across the atrium, one of the active cameras is selected to follow and acquire generic frontal-view images of the person. Before the TACs initialization, the Data Integration module of the SVT computes the 3D coordinates of the agent, which are based on the information from the background subtraction algorithm of the static camera (TSC). The module for High-Level Reasoning of the SVT relies on this geometric data for inference and generation of high-level commands.

The results in Fig. 9 and 10 are from two sequences showing a person individually followed by the most appropriate camera to get a frontal view. On the left and right columns are the images acquired from the two cameras, *hermes1* and *hermes2*, the location of which was highlighted in Fig. 6. The middle column shows the current path of the SGT-traversal as discussed in Section 5.3 (see details of the SGT in Fig. 7).

In the sequence of Fig. 9 the agent moves from the aisle to the bridge. The camera *hermes1* is chosen to follow the agent according to its direction, so that frontal-view images of the latter can be acquired. The camera *hermes2* is in idle state during the whole process. The sequence in Fig. 10 shows the opposite case, where the agent, walking from the bridge to the aisle, is followed by *hermes2*, while *hermes1* is idle.

As highlighted by the relative SGT-traversal, the active camera is selected based on the high-level command `follow` sent by the SVT, in which the parameter `Camera` is set to be either `hermes1` or `hermes2`. Note also that agents are always kept in the field of view of this camera, demonstrating that the use of high-level commands from the inference process of the SVT are sufficiently fast for autonomous surveillance.

This experiment shows in particular two important properties of our implementation:

- through an extensive but efficient use of databases for message passing, observations and commands generated by the system are processed in real-time, as already reported in a previous work [1];

- multiple cameras are selected according to a high-level interpretation of the current human motion, which is not only based on image sequences, but also on a semantic representation of the environment.

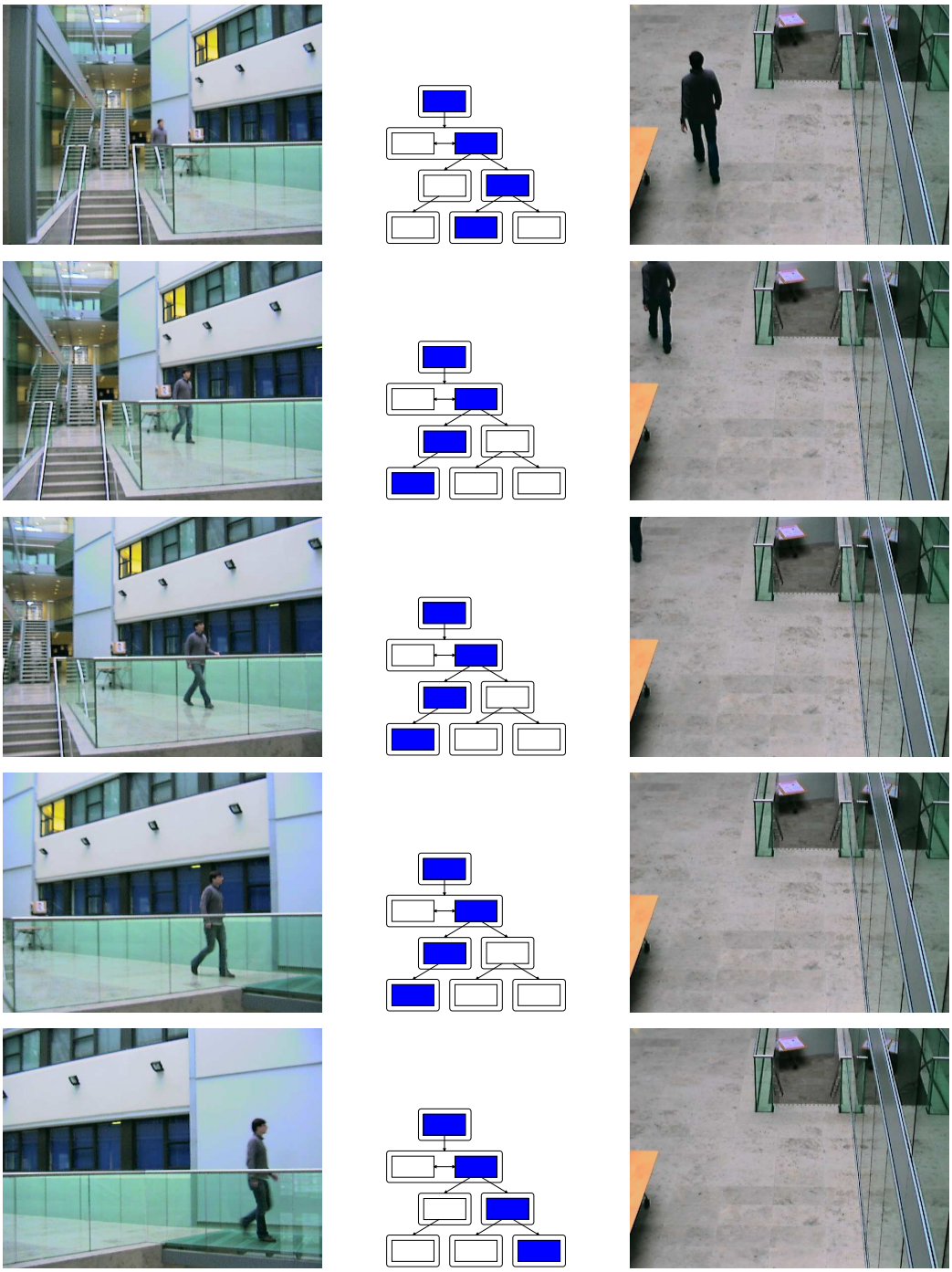

Figure 9: Selection of camera *hermes1* for target following.

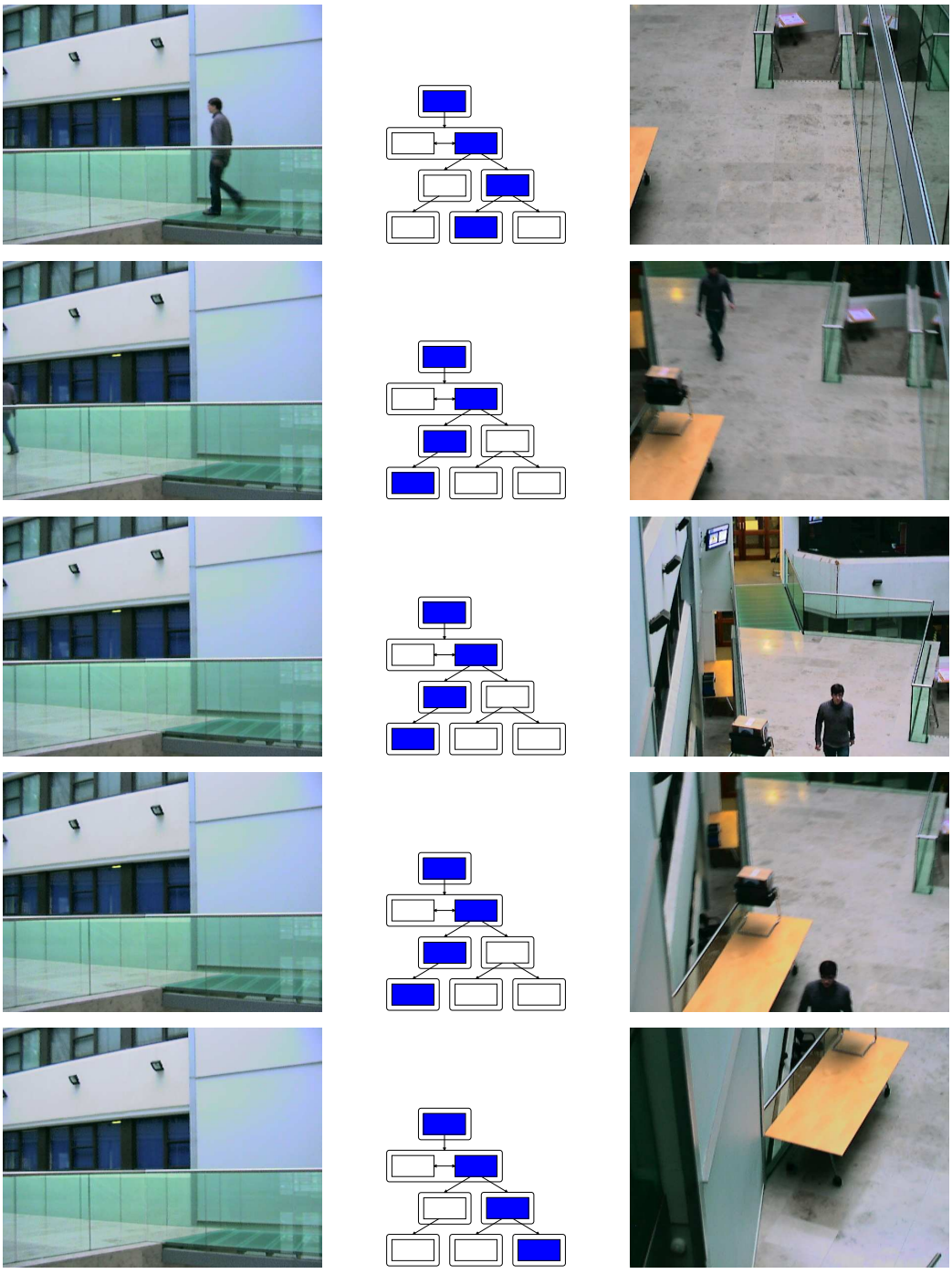

Figure 10: Selection of camera *hermes2* for target following.

However, one of the limitations of the current selection strategy is that, once the camera *hermes1* or *hermes2* has been enabled, it continues to follow the target until this leaves the scene, or at least until is lost. That is, the current SGT can correctly initialize the camera the target is facing to, but at the moment it lacks a mechanism to stop this camera when the target moves in the opposite direction. Such a situation is illustrated by the sequence in Fig. 11. Here the agent moves initially towards the bridge and the camera *hermes1* is successfully selected to follow, while *hermes2* is idle. When the agent, in the middle of the atrium, decides to turn back and move towards the aisle, *hermes2* is enabled too, and both the cameras keep on following the person until he leaves the scene. New rules will have to implemented to forcibly stop a camera when it is not needed anymore.

6.2. Target Identification

The next examples illustrate the effect of camera commands to identify people walking in the atrium. We show how inference is used to control the pan-tilt-zoom of active cameras and collect close-up images for further face recognition. The SGT, in this case, is the one illustrated in Fig. 8. Running the system over an extended period yields facial snapshots of every individual who traversed the area under surveillance. Fig. 12 shows a series of snapshots of people acquired by the system and stored in the database over a period of a few hours. These data were recorded during the evening, when few people were crossing the area; they are trivially recovered from the database, even when weeks old. In this case the face recognition process was turned off, so the system has no identifications.

Output from the process running with full recognition (using a small database of 5 individuals) is shown in Fig. 13, in which the camera is controlled to obtain good facial images for identification. The figure shows the images acquired using the TAC, along with relative SGT-traversals and the result of the inference. The sequence in the figure illustrates an agent walking from the aisle to the bridge. The agent is initially detected by the static camera (TSC) on the top of the atrium and processed by the High-Level Reasoning module of the SVT. In particular, the first row of the sequence shows an agent entering the atrium from the aisle. Its behaviour is described by the status messages generated by the current SGT-traversal, which are shown on the right-hand side of the figure, and written to the Inference Table.

The High-Level Reasoning then sends a `track` command to a specific TAC, based on the inference over the low-level information obtained from

Snapshots from camera *hermes1* at time $t = 0, 4, 8, 12, 16, 20, 24, 28$ $[s]$

Snapshots from camera *hermes2* at time $t = 0, 4, 8, 12, 16, 20, 24, 28$ $[s]$

Figure 11: Selection of both cameras for person-following.

the TSC. Once the agent is successfully tracked, face images (shown on the top-right corner of the second and third snapshot) are sent to the database. A `recognize` command activates the face recognizer, which retrieves these images from the database and tries to determine the agent's identity. In this case, the TAC tracks the agent on the second and third frame of the sequence until it is identified, as specified by the inference results relative to the fourth frame (i.e. status `has_identity`). The successful recognition causes a `follow` command to be sent to the same TAC, which therefore zooms-out and keeps simply the person within its field of view. The TAC still follows the agent when this leaves the atrium through the bridge.

In this experiments, it is important to note the path of the SGT-traversal

Figure 12: Face images acquired during an extended operation of the system. The images had been manually "anonymised".

in the middle column of the figure. While the change of the traversal's path between the first and the second row, or between the fourth and the fifth row, depends only on the particular behaviour of the agent (*entering* or *leaving* the atrium), the difference between the third and the fourth row is a direct consequence of the high-level commands generated by the system. In particular, the execution of `track` and `recognize` permits the change of the agent' state from `is_unidentified` to `has_identity`, and the consequent traversal of a different branch of the SGT.

Although the current SGT has been designed to deal with a single target, the case reported next describes a simple scenario in which multiple people are present. The new sequence illustrated in Fig. 14 shows five different individuals in the atrium area, one of whom is walking towards the bridge

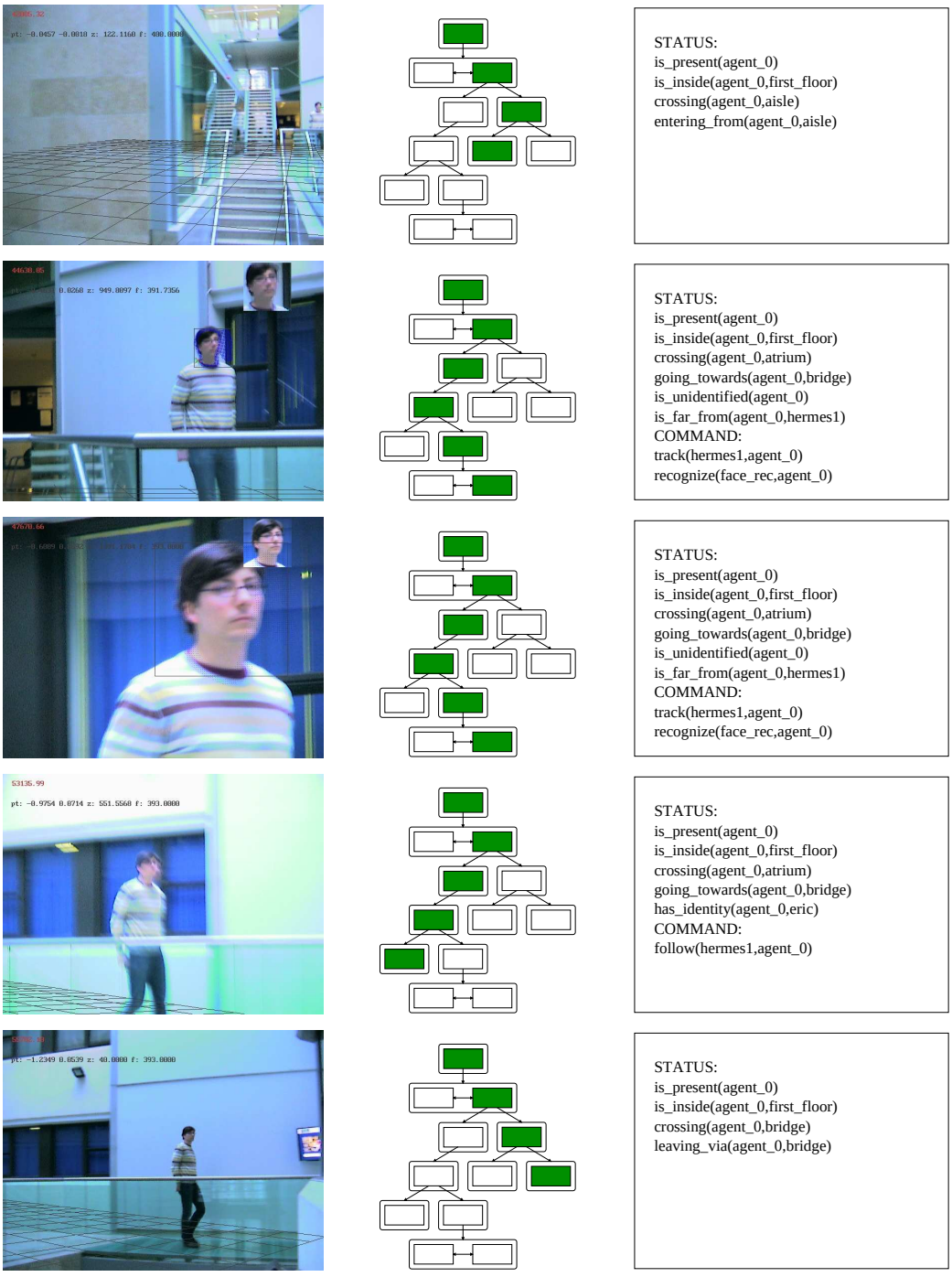

Figure 13: Target tracking and identification.

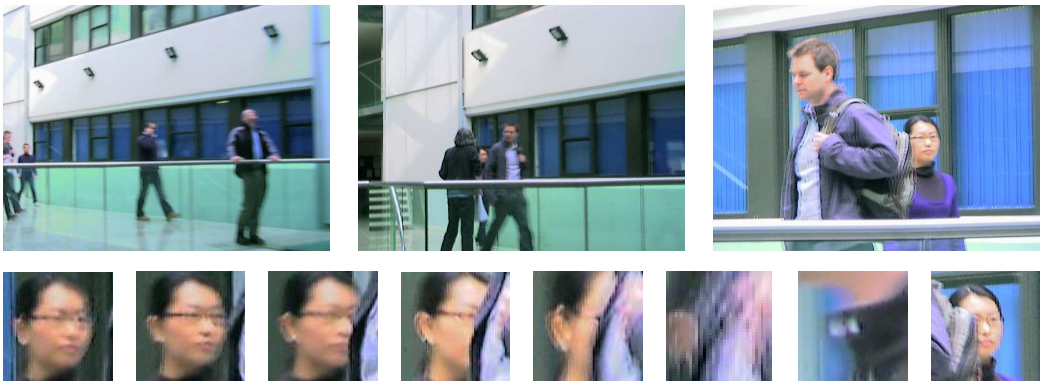

Figure 14: Target tracking with multiple people. Note that in this case the face recognition was unsuccessful because of occlusions.

and facing the *hermes1* camera. Even in this case, despite the presence of other people, the High-Level Reasoning enables the same process "select active camera, zoom on face and recognize" for the target of interest. This target is simply chosen according to the order of the evidence that is available at the current time step. In practice, the Observation Table of the SQL database functions as a queueing mechanism, in which the oldest target is considered first. As long as the target exists, it has precedence over all others, and the system's commands focus on it. Further investigation on how to handle multiple targets is beyond the scope of this paper and is left for future extensions of our work.

7. Conclusions

We have described a system architecture which facilitates linking of high-level inference to sensing actions. We have demonstrated this in some simple scenarios, but there remains much to do. Notably, we have yet to take proper advantage of the fuzzy capabilities of the reasoning system. To do so would require that we map probabilities (such as those returned by the visual tracking algorithm) to fuzzy degrees of belief. A weakness of the current inference mechanism is that it proceeds in a depth-first fashion via thresholds set on the fuzzy degrees of belief. An alternative which would have significant benefits would be *breadth-first* to enable selection between multiple competing hypotheses. In this instance we could then consider actions designed deliberately to reduce uncertainty.

In parallel with our work on situational rule-based camera control, we have also been exploring the use of information theoretic means to achieve emergent cooperative behaviour from a set of cameras, via choosing actions to minimise uncertainty about the scene [18]. We are currently interesting in applying this approach to maximise information about the current belief at the conceptual level by choosing the right camera settings. This would require the extension to breadth-first search mentioned before.

Furthermore, we would like to explore the possibility of this form of camera control as an intermediate layer between the low-level "reactive" processes, such as closed-loop tracking, and the high-level commands. In this way, the inference process could emit actions at a more abstract level (e.g. "monitor scene" or "track" without reference to a specific camera), allowing the intermediate layer to make appropriate choices for satisfying abstract goals.

Acknowledgments

This work was supported by the EU FP6 grant number IST-027110 for the HERMES project – http://www.hermes-project.eu

References

[1] N. Bellotto, E. Sommerlade, B. Benfold, C. Bibby, I. Reid, D. Roth, C. Fernández, L. V. Gool, J. Gonzàlez, A distributed camera system for multi-resolution surveillance, in: Proc. of the 3rd ACM/IEEE Int. Conf. on Distributed Smart Cameras (ICDSC), Como, Italy, 2009.

[2] A. Prati, F. Seghedoni, R. Cucchiara, Fast dynamic mosaicing and person following, in: Int. Conf. on Pattern Recognition (ICPR), 2006, pp. 920–923.

[3] R. I. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, 2nd Edition, Cambridge University Press, 2004.

[4] K. H. Schäfer, Unscharfe zeitlogische Modellierung von Situationen und Handlungen in der Bildfolgenauswertung und Robotik, Dissertation, Fakultät für Informatik der Universität Karlsruhe, ISBN 3-89601-135-9, (in German) (1996).

[5] T. Kanade, Region segmentation: Signal vs semantics, in: Proc. of 4th Int. Joint Conf. on Pattern Recognition, 1978, pp. 95–105.

[6] H.-H. Nagel, Image sequence evaluation: 30 years and still going strong, in: Proc. of the 15th Int.l Conf. on Pattern Recognition (ICPR), Vol. 1, IEEE Computer Society, Los Alamitos, CA, USA, 2000, pp. 1–6.

[7] T. O. Binford, Bayesian inference in model-based machine vision, International Journal of Approximate Reasoning 2 (3) (1988) 327–328.

[8] S. Intille, A. Bobick, Representation and visual recognition of complex, multi-agent actions using belief networks, Tech. Rep. 454, M.I.T. Media Laboratory Perceptual Computing Section (1998).

[9] P. Remagnino, T. Tan, K. Baker, Agent orientated annotation in model based visual surveillance, in: Proc. of the 6th Int. Conf. on Computer Vision (ICCV), 1998, p. 857.

[10] N. Robertson, I. Reid, A general method for human activity recognition in video, Computer Vision and Image Understanding 104 (2) (2006) 232–248.

[11] D. Ayers, M. Shah, Monitoring human behavior from video taken in an office environment, Image and Vision Computing 19 (12) (2001) 833–846.

[12] A. Kojima, T. Tamura, K. Fukunaga, Natural language description of human activities from video images based on concept hierarchy of actions, International Journal of Computer Vision 50 (2) (2002) 171–184.

[13] B. Tordoff, D. Murray, Reactive zoom control while tracking, in: Proc. of 12th British Machine Vision Conf. (BMVC), 2001, pp. 53–62.

[14] J. Denzler, M. Zobel, H. Niemann, Information theoretic focal length selection for real-time active 3-d object tracking., in: 9th IEEE Int. Conf. on Computer Vision (ICCV), 2003, pp. 400–407.

[15] S. Tsuruoka, T. Yamaguchi, K. Kato, T. Yoshikawa, T. Shinogi, A camera control based fuzzy behaviour recognition of lecturer for distance lecture, in: Proc. of 10th IEEE Int. Conf. on Fuzzy Systems, 2001, pp. 940–943.

[16] A. Hampapur, S. Pankanti, A. Senior, Y.-L.Tian, L. Brown, R. Bolle, Face cataloger: Multi-scale imaging for relating identity to location, in: Proc. of the IEEE Conf. on Advanced Video and Signal Based Surveillance, Washington, DC, USA, 2003, pp. 13–20.

[17] A. D. Bimbo, F. Dini, G. Lisanti, F. Pernici, Exploiting distinctive visual landmark maps in pan-tilt-zoom camera network, Computer Vision and Image Understanding 114 (6) (2010) 611–623.

[18] E. Sommerlade, I. Reid, Probabilistic surveillance with multiple active cameras, in: IEEE Int. Conf. on Robotics and Automation (ICRA), 2010, pp. 440–445.

[19] N. Zouba, F. Bremond, M. Thonnat, An activity monitoring system for real elderly at home: Validation study, in: 7th IEEE Int. Conf. on Advanced Video and Signal Based Surveillance (AVSS), 2010, pp. 278–285.

[20] F. Brémond, M. Thonnat, M. Zúñiga, Video-understanding framework for automatic behavior recognition, Behavior Research Methods 38 (3) (2006) 416–426.

[21] N. Apostoloff, A. Zisserman, Who are you? – real-time person identification, in: Proc. of the 18th British Machine Vision Conf. (BMVC), 2007.

[22] A. Patron, I. Reid, A Probabilistic Framework for Recognizing Similar Actions using Spatio-Temporal Features, in: Proc. of the 18th British Machine Vision Conf. (BMVC), 2007.

[23] B. Benfold, I. Reid, Guiding visual surveillance by tracking human attention, in: Proc.s of the 20th British Machine Vision Conf. (BMVC), 2009.

[24] D. Hall, J. Nascimento, P. Ribeiro, E. Andrade, P. Moreno, S. Pesnel, T. List, R. Emonet, R. B. Fisher, J. S. Victor, J. L. Crowley, Comparison of target detection algorithms using adaptive background models, in: ICCCN '05: Proc. of the 14th Int. Conf. on Computer Communications and Networks, IEEE Computer Society, Washington, DC, USA, 2005, pp. 113–120.

[25] T. Boult, R. Micheals, X. Gao, M. Eckmann, Into the woods: visual surveillance of noncooperative and camouflaged targets in complex outdoor settings, Proceedings of the IEEE 89 (10) (2001) 1382–1402. doi:10.1109/5.959337.

[26] N. Bellotto, H. Hu, Computationally efficient solutions for tracking people with a mobile robot: an experimental evaluation of Bayesian filters, Autonomous Robots 28 (4) (2010) 425–438.

[27] P. Viola, M. J. Jones, Robust real-time face detection, Int. Journal of Computer Vision 57 (2) (2004) 137–154.

[28] C. Bibby, I. Reid, Robust real-time visual tracking using pixel-wise posteriors, in: Proc. of European Conference on Computer Vision (ECCV), 2008.

[29] M. Everingham, J. Sivic, A. Zisserman, "Hello! My name is... Buffy" – Automatic Naming of Characters in TV Video, in: Proc. of the British Machine Vision Conf. (BMVC), 2006.

[30] E. Tsomko, H. J. Kim, J. Paik, I.-K. Yeo, Efficient method of detecting blurry images, Journal of Ubiquitous Convergence Technology 2 (1) (2008) 27–39.

[31] K. H. Schäfer, Limette User Manual, Universität Karlsruhe (1997).

[32] M. Haag, H.-H. Nagel, Incremental recognition of traffic situations from video image sequences, Image and Vision Computing 18 (2) (2000) 137–153.

[33] M. Arens, R. Gerber, H.-H. Nagel, Conceptual representations between video signals and natural language descriptions, Image and Vision Computing 26 (1) (2008) 53–66, Cognitive Vision - Special Issue.