2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

Cognitive Active Vision for Human Identification

Yuzuko Utsumi, Eric Sommerlade, Nicola Bellotto and Ian Reid

Abstract— **We describe an integrated, real-time multi-camera surveillance system that is able to find and track individuals, acquire and archive facial image sequences, and perform face recognition. The system is based around an inference engine that can extract high-level information from an observed scene, and generate appropriate commands for a set of pan-tilt-zoom (PTZ) cameras. The incorporation of a reliable facial recognition into the high-level feedback is a main novelty of our work, showing how high-level understanding of a scene can be used to deploy PTZ sensing resources effectively. The system comprises a distributed camera system using SQL tables as virtual communication channels, Situation Graph Trees for knowledge representation, inference and high-level camera control, and a variety of visual processing algorithms including an on-line acquisition of facial images, and on-line recognition of faces by comparing image sets using subspace distance. We provide an extensive evaluation of this method using our system for both acquisition of training data, and later recognition. A set of experiments in a surveillance scenario show the effectiveness of our approach and its potential for real applications of cognitive vision.**

I. INTRODUCTION

Although low-level visual feedback for real-time tracking of visual targets using pan-tilt-zoom (PTZ) cameras is a much studied area, very little attention has been focused on the problem of control using high-level feedback. By this we mean control of the PTZ cameras in response not to some low-level cue such as target position, but in response to some higher level abstraction of the scene and its agents. In this paper we describe such a system that incorporates an inference engine to reason about the scene and agents in the scene, and a set of control actions that are triggered during the inference process in response to the current understanding of the scene. In particular, the goal of the system we describe is to acquire sequences of people using multiple cameras as they traverse an area under surveillance, and to maximise the views of the faces at suitable resolution for recognition, and to perform recognition.

We base our system on an architecture described in [1] that considered the issues of low-level data acquisition processes, and how these processes communicate. In the current paper we extend this architecture, showing how we can 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 [2], [3]. A significant contribution of our work is to extend this paradigm by employing it in an active system for PTZ camera control; camera commands which suit a relevant situation are specified in the action part of the description. 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 associated 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).

We root our exploration of this general approach in a particular application, in which the goal of the system is to acquire sequences of people using multiple cameras as they traverse an area under surveillance, to maximise the views of the facial views obtained at suitable resolution for recognition, and to perform recognition. To this end we have implemented a recent sequence-to-sequence matching scheme [4]. We provide an evaluation of this algorithm for the case of automatically acquired facial images and show that it is remarkably effective for a database of individuals which was acquired automatically using our “detect, track, archive” in spite of variations in lighting, facial orientation, etc., as well as being suitable for on-line recognition. A working example of collaboration between static and active cameras has been given recently by Krahnstoever et al.[5], where active cameras are steered according to the inputs from static ones. The data the active cameras provide is used for identification purposes, but not for active, on-line control. Soto et al.[6] address the task of tracking with multiple, active cameras in a game-theoretic fashion. The collaboration among the cameras is obtained from a distributed method that seeks to reach consensus among neighbouring nodes. The resulting control provides complete coverage of the acquired targets, while providing higher resolution imagery from a few targets. Contrary to our method, the whole target has to remain in the field of view of the camera, which puts an upper bound on the resolution attainable.

II. DISTRIBUTED CAMERA SYSTEM AND VISUAL PROCESSING

In this section we recap the architecture described in [1], a schematic of which is illustrated in Fig. 1. The system comprises a set of distributed static and PTZ cameras and

Y. Utsumi is with Graduate School of Engineering, Osaka Prefecture University, Japan yuzuko@cs.osakafu-u.ac.jp

E. Sommerlade and I. Reid are with Dept of Engineering Science, University of Oxford, UK {eric, ian}@robots.ox.ac.uk

N. Bellotto is with Centre for Vision and Robotics Research, University of Lincoln, UK nbellotto@lincoln.ac.uk

This work was supported by the EU FP6 grant number IST-027110 for the HERMES project – http://www.hermes-project.eu, Grant-in-Aid for JSPS Fellows No. 21·558, and Oxford Risk

978-1-4673-1404-6/12/$31.00 ©2012 IEEE

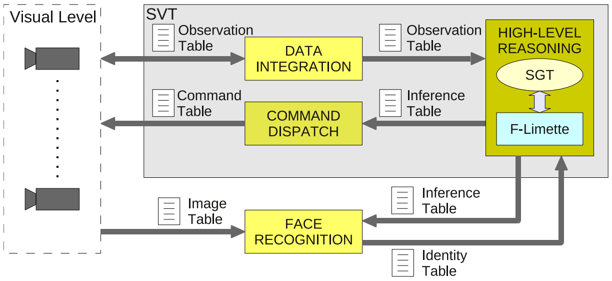


Fig. 1. Conceptual operation of the system in which tables are shown as one or two way communication paths between processes. All these tables are held in a central SQL database.

visual tracking algorithms, together with a central supervisor process.

In our implementation, there are two active cameras and a wide-angle overhead static camera. Each camera and its dedicated processing unit constitutes an independent "tracker" module. The Tracker Static Camera (TSC) and the Tracker Active Cameras (TACs) are connected and synchronised with a supervisor process via TCP/IP. In particular, the communication between all these modules is done via an SQL database: messages are "posted" into an SQL table, and read (via polling) by the recipient. For example, visual tracking data from the TSC are stored dynamically into a table of the database, a data association process running on a supervisor processor forms tracks and stores these into another table, while the supervisor (acting on actions triggered during inference) also stores demands for the PTZ cameras into an SQL table read by the TACs.

The static camera of our system (TSC) is used for real-time human detection on wide-angle image sequences. To detect people, we use a variant of the "LOTS" background subtraction method [7], [8].

A supervisor process (SVT) is responsible for the data fusion, reasoning and camera control strategy. The Data Integration module collects sensor observations from one or more cameras and generates trajectories of the current targets. This is implemented using Kalman filters and nearest-neighbour data association [9]. The High Level Reasoning, described in the next section, generates the most appropriate camera commands according to a semantic interpretation of the current observations. 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 arguments, like camera number or target ID).

A control command sent to a TAC client 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 velocity control to centre the projection of this area in the camera's view. Once the camera is sufficiently close, a standard face detection method [10] is run on the observed area. If a face is detected, the client tracks the target using a visual tracking method based on level sets [11]. The focal length for the tracking process is determined from the geometry of the scene to ensure a suitable compromise between facial resolution and to mitigate against target loss. 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.

In the next two sections we detail the inner workings of the processes that comprise the cognitive identification system. The first is the control method using high level inference, and the second an identification method that works with non-cooperative targets.

III. On-line Inference and Camera Control

The High-Level Reasoning module 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". Our step towards this goal is the use of an inference engine that combines situational *a priori* knowledge with visual predicates formed from the real-time data stream in SQL tables.

A. Situation Graph Trees

The *a priori* knowledge about the expected behaviour of an agent can be seen as a sequence of situated actions. We encode this using a co-called Situation Graph Tree (SGT) which is specified in a formal language, SIT++, based on a Fuzzy Metric-Temporal Horn Logic (FMTHL) [2], [3]. 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 of a tree 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.

The traversal of the SGT yields an instance of an agent's behaviour in this conceptual representation and for a particular time interval. This is the output of an inference engine for FHTML, called *F-Limette* [12], which is delivered in the form of time-indexed predicates obtained from the SGT.

The inference engine tries to find the most specific instantiation of behaviour given current knowledge, and traverses the tree in a depth-first search. It proceeds by unifying the fuzzy predicates generated by the low-level vision processes, whereas the SGT encodes the "rules of engagement".

B. High-Level Reasoning

The High-Level Reasoning module in Fig. 1 is responsible for knowledge representation, inference and (high-level) camera control. It is an independent process of the SVT that communicates with the other two modules, Data Integration and Command Dispatch, via SQL tables. Fig. 1 shows in the SVT and how the High-Level Reasoning module is integrated within the system. Virtual communication channels (shown in grey) are established via the database tables as indicated. Quantitative data, generated by the cameras and the Data Integration, is stored in the Observation Table and then converted into a list of (qualitative) predicates with F-Limette. An inference thread is invoked on the SGT when new visual predicates are available.

The results of the inference is written to an Inference Table. In case it contains high-level commands, these are processed by the Command Dispatch module and delivered to the TACs. The Inference Table is also used to control additional modules external to the SVT, like the Face Recognition discussed in Section IV. In this case, a "recognise target" command from the High-Level Reasoning starts the recognition process on images provided by the TAC(s) via an Image Table. The result of the recognition, which can be either a person's name or "unknown", is sent back to the High-Level Reasoning through an Identity Table.

IV. Face Recognition

As soon as an active camera acquires face images of a given target, it stores this data along with the current target's identification number (as specified by the system) into the database. A face recognition process polls the database for a commands to identify a target. This command specifies the identification number for this target, which is subsequently used to look for new face images in the database.

Despite recent advances in face recognition [13], most identification methods require collaboration of the user, and controlled lighting. Both these conditions are not given in a surveillance setting. Our solution to this problem is based on the face recognition algorithm proposed by Cevikalp and Triggs[4], which attempts sequence-to-sequence classification using a conceptually simple idea, but which in our experience is very effective. Instead of identifying single images, a set of images is used both in the training, as well as the testing stage. Each input datum (i.e. image) of a class (i.e. person) is represented by a point in a high dimensional vector space. A set of input data (i.e. a sequence of face images) is then approximated by the linear subspace it spans (see figure 2). The distance of two sets of images is then the distance between the closest points in each class' subspace. This approximates the variation in the input data, and yields robust classification performance. Mathematically, this is expressed as follows. The affine hull is a linear combination of each input vector $\mathbf{x}_{c,k}$ (out of K_c) of a class c, where all coefficient must sum to one:

$$H_c^{\text{aff}} = \left\{ \sum_k \alpha_k \mathbf{x}_{c,k} | \sum_k \alpha_k = 1 \right\}. \tag{1}$$

I.e. each $\mathbf{x}_c$ has a unique combination of coefficients α_k. The constraint $\sum_k \alpha_k = 1$ can be rewritten in form of differences of the input data, which span a basis of rank $M \leq K_c - 1$. This can be expressed in a matrix $\mathbf{U} \in \mathbb{R}^{K_c \times M}$:

$$H_c^{\text{aff}} \quad = \quad \{\boldsymbol{\mu}_c + \mathbf{U}_c \mathbf{v}\}. \tag{2}$$

The distance of a point to a class is thus the minimum distance to its subspace:

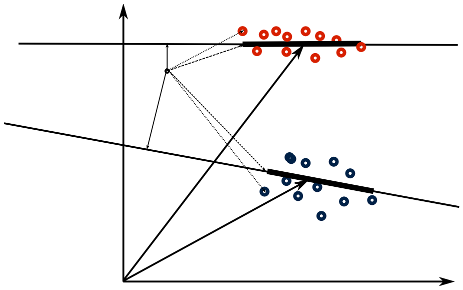


Fig. 2. Different distances of a test sample (empty circle) to either the affine hull (lines through points), the convex hull (thick lines), or the nearest neighbour of each class. Note that affine hulls encompass all points of a class. Here, we assume that there is noise on each training data point and only the subspace dimensions with sufficient variance are kept.

$$d^*(\mathbf{x}, H_c^{\text{aff}}) \quad = \quad \arg \min_{\mathbf{v} \in \mathbb{R}^M} \|\mathbf{x} - \boldsymbol{\mu}_c - \mathbf{U}_c \mathbf{v}\| \tag{3}$$

This results in an over-determined linear equation system which can be solved using Normal equations. Given two image sets (from training and testing) and their affine hulls $H_{1,2}^{\text{aff}}$, we look for the closest point on both subspaces:

$$d^*(H_1^{\text{aff}}, H_2^{\text{aff}}) \quad = \quad \arg \min_{\mathbf{x}_1 \in H_1^{\text{aff}}, \mathbf{x}_2 \in H_2^{\text{aff}}} \|\mathbf{x}_1 - \mathbf{x}_2\| \tag{4}$$

When rewriting $[\mathbf{U}_1 \ \ \mathbf{U}_2] = \mathbf{U}$ and $[\mathbf{v}_1^T \ \ \mathbf{v}_2^T]^T = \mathbf{v}$, this yields the following linear equation system:

$$\mathbf{v} = (\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1). \tag{5}$$

V. Application to Cognitive Active Vision

In this section we develop an SGT for the case of an agent moving across a large 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 IV as a visual processing module.

As explained before, we couple high-level inference to sensing strategies that control a set of active cameras for human identification. 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 design of an SGT for human behaviour interpretation and intelligent camera control.

A. 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. 3. 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.

Each polygonal segment in Fig. 3 describes a possible position in which an agent can be found within the scene, and is annotated by a label which determines this area's conceptual description and type. In this work, we distinguish two different type of segments:

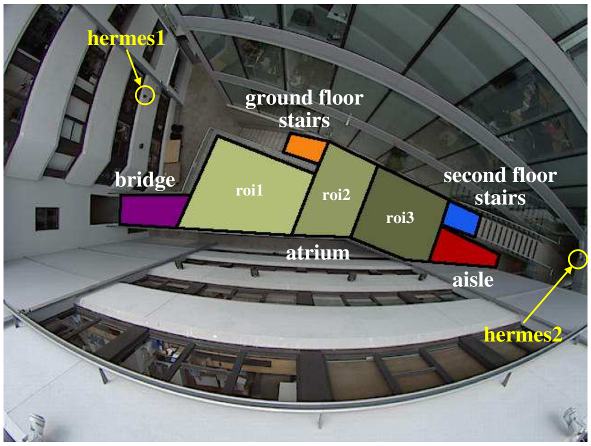


Fig. 3. Atrium and floor-plan showing semantically distinct regions used for inference. *hermes1* and *hermes2* are the two active cameras (TACs).

- an **exit** type, which includes the *bridge*, *stairs* and *aisle*. These areas lead to other, non-supervised regions.
- an **atrium** type, which includes the segments labelled *roi-i*. These form the main activity area of the atrium.

. In the next section we describe how to form predicates that relate the spatial position of an agent to the segments above.

B. SGT for Human Identification and Camera Control

Fig. 4 depicts the SGT used in the first scenario to describe simple behaviour of agents and trigger camera actions for identification and tracking of targets. The root node in the first layer (with the least specialisation), uses the predicate `active` to describe whether a target is present in the system. In the language used for logic programming, this binds the variable `Agent` (variables always start with capital letters) to a particular target ID from the Observation Table.

The initial scheme is specialised 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 arrows). 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 particularised 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 specialisation in Layer 4 is used to determine the direction of the target and select the most appropriate camera for identification, as discussed below.
- `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 specialised situation graphs in Layer 4, each containing a single scheme:
  - `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.

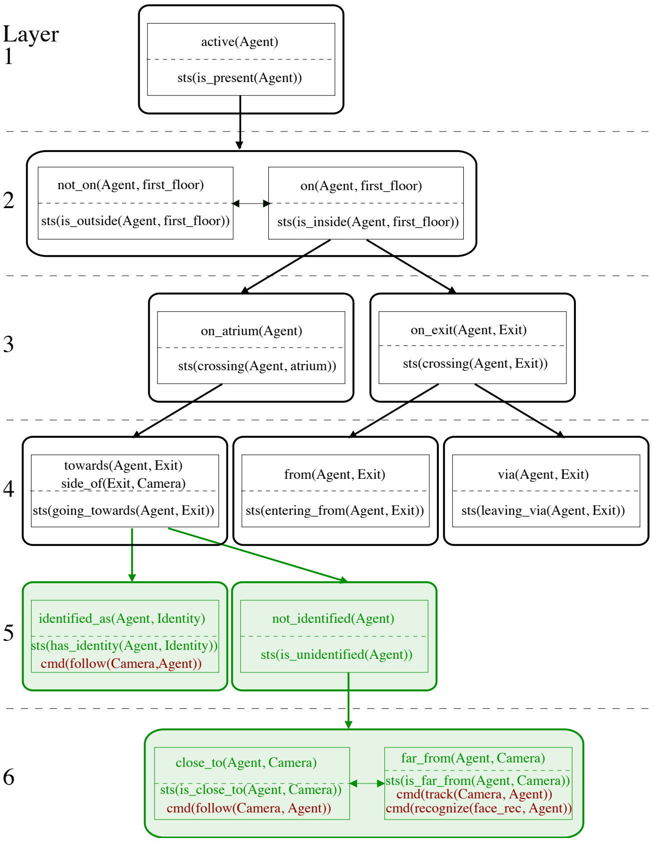

Layer
1
2
3
4
5
6
active(Agent)
sts(is_present(Agent))
not_on(Agent, first_floor)
sts(is_outside(Agent, first_floor))
on(Agent, first_floor)
sts(is_inside(Agent, first_floor))
on_atrium(Agent)
sts(crossing(Agent, atrium))
on_exit(Agent, Exit)
sts(crossing(Agent, Exit))
towards(Agent, Exit)
side_of(Exit, Camera)
sts(going_towards(Agent, Exit))
from(Agent, Exit)
sts(entering_from(Agent, Exit))
via(Agent, Exit)
sts(leaving_via(Agent, Exit))
identified_as(Agent, Identity)
sts(has_identity(Agent, Identity))
cmd(follow(Camera,Agent))
not_identified(Agent)
sts(is_unidentified(Agent))
close_to(Agent, Camera)
sts(is_close_to(Agent, Camera))
cmd(follow(Camera, Agent))
far_from(Agent, Camera)
sts(is_far_from(Agent, Camera))
cmd(track(Camera, Agent))
cmd(recognize(face_rec, Agent))

Fig. 4. SGT for active camera selection and face recognition.

As noted before, whenever a node in the SGT is satisfied during the traversal, it can optionally emit an action. In previous work [3], [14], 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.

Remember that the first situation in Layer 3 refers to the case when the agent is walking in the atrium. The specialisation that follows in Layer 4 seeks to determine its direction (predicate `towards(Agent, Exit)`) and the camera it is pointing 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 of the subsequent layers. 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.

Much more interesting are the cases illustrated in Layer 5 and 6, where the result of an action has direct consequences on the next traversal; the aim of this action will be to acquire new information to enable a deeper instantiation in the SGT. This part of the SGT performs both camera selection and "follow" behaviour; additionally, it can instruct a camera to "track" a target's face and "recognise" it.

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. 3) to follow the agent. This string is interpreted by the Command Dispatch module of the SVT and sent to the relevant camera via the Command Table. 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 recognise 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, refer to the left-hand branch of the SGT tree in which a new agent has entered the scene. Upon reaching Layer 4 with the instantiation of the Agent and Camera variables, a new specialisation in Layer 5 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 recognise 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(recognise(face_rec, Agent), instead, invokes the Face Recognition module to identify the target as already explained.
- 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.

VI. Experiments

We have conducted numerous experiments with the overall system in [1], showing that our SQL-based architecture is suitable for inter-communication, control and data storage of a distributed multi-camera system. Here we evaluate the performance of the classification method on real-world data, acquired by application of the SGT in section V without writing identification results into the database (i.e. the predicate not_identified(Agent) is never satisfied). The acquired data is used to learn classifiers for a set of individuals. A successful recognition process then yields a database entry and thus closes the perception-action cycle defined by previous SGT.

A. Evaluation of Face Classifiers

To train the affine hull classifiers, we use facial sequences which have been acquired intermittently by the surveillance system over a period of 20 days. This part of the database comprises about 200 sequences involving about 150 individuals, with sequence length varying between 50 and 600 frames (2–20 seconds).

The imagery from the database poses the typical obstacles that need to be overcome by surveillance system. Despite of an enclosed environment, the lighting conditions of the sequences vary considerably with the weather and time of day, as the ceiling of the atrium is semitransparent. Furthermore, the posture of the face varies from fully frontal to profile views. Some images are also severely affected by motion blur. Obviously, the acquisition process is also not devoid of error – whenever an initial detection yields a false positive (e.g. a person's reflexion in an office window), the level set tracker will yield a set of non-face images (yet accurately tracked). These misacquisitions and outliers comprise 11 percent of the data set.

Lastly, as the images are the output of a robust level-set tracker, they contain significant parts of background. For the purpose of recognition, the faces are thus centred by cropping to the output of the Viola-Jones face detector[10], converted to grey-scale, resampled to a resolution of 20×20, and histogram equalised.

Seven individuals in the database consented to the use of their images for evaluation of the recognition classifiers, constituting 17308 images. we are able to identify only the seven individuals correctly in the database. Because of

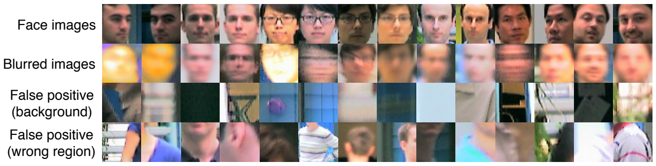


Fig. 5. Typical examples of the input data.

exact evaluations, we use the seven individuals. For training and evaluation purposes, we use 6421 images of other individuals, comprising an “unknown” class. Fig 5 shows some examples of the database images.

1) Selection of Training Data: The quantity and quality of training data affects the recognition results. Here we focus on the aspect of the training data that has the highest relevance to the system: the variation in the data.

Ideally, we would like to use a single sequence (i.e. continuous recording) of an individual as training data for a classifier, and the shortest sequence possible. The first makes acquisition of new individuals much faster, but does not capture the variation due to the environment. The same holds for the second: the more variation by the individual is required, the longer the system has to track a target for initial acquisition.

To begin we consider taking training data from a single sequence of an individual, and vary the number of images taken between 15 and 240 in increments of 15. Each batch of 15 images forms a subsequence (i.e. approx 0.5 seconds). We compare this with choosing the same number of images, but each subsequence selected randomly from all sequences of each individual in the database.

The affine hull classifier is evaluated using the rest of the database, and each test comprises recognition evaluation of a subsequence.

For off-line evaluation, we repeat the training and testing procedure 10 times, and report average recognition results using recall-precision curves shown in Fig. 6. Asterisks shows the recognition results when 15 images were used for learning, and circles shows the results when 30–240 images were used for learning.

Unsurprisingly, the recognition results improve when the number of training images is increased. Also unsurprisingly, when the training data are selected randomly from different sequences (of the same individual of course), rather than drawn from a single sequence, the recognition results are better. This is to be expected because the randomly drawn samples span a wider variation in possible input conditions.

2) Selection of norm: We employ principal component analysis to obtain the basis $\mathbf{U}$ that spans the subspace of the classifier (see equation 2). We evaluate the influence of the outliers on the recognition result by comparing the performance of the standard, linear PCA approach based on singular value decomposition with the L1-norm PCA [15]. Training images were chosen as before, but both training as well as testing data was polluted by outliers. We learned and evaluated the classifier 10 times and report the average in figure 6.

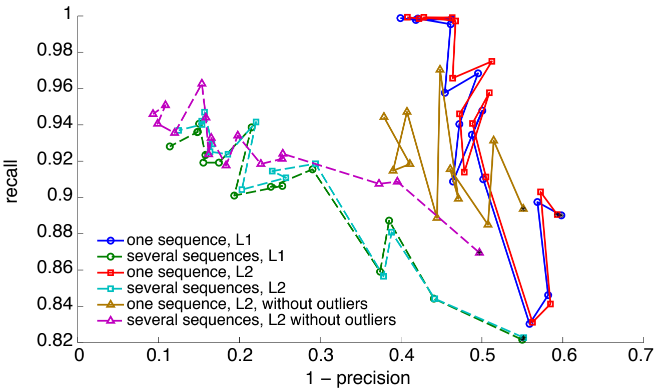


Fig. 6. ROC curve of classification results.

The recognition results of the L1-norm method show fairly similar results to the conventional PCA, which renders this approach ineffective for the type of outliers in our database. As the (computational) performance of the standard PCA is much higher than the L1-norm PCA, we choose the former (L2-norm) for face recognition.

B. Target Identification

The next example illustrates the effect of camera commands to identify unknown pedestrians walking in the atrium. It shows how inference is used to control the pan-tilt-zoom of active cameras and collect close-up images for further face recognition. The SGT used is the one illustrated in Fig. 4.

Running the system over an extended period yields facial snapshots of every individual to have traversed the area under surveillance. For the purposes of our subsequent experiments we used our “detect, track, archive” SGT-controller to automatically acquire 200 sequences (23729 frames) of 150 individuals over several days of operation. Snapshots are trivially recovered from the database, even when weeks old. During this acquisition process the face recognition process was not enabled, so the system has no identifications.

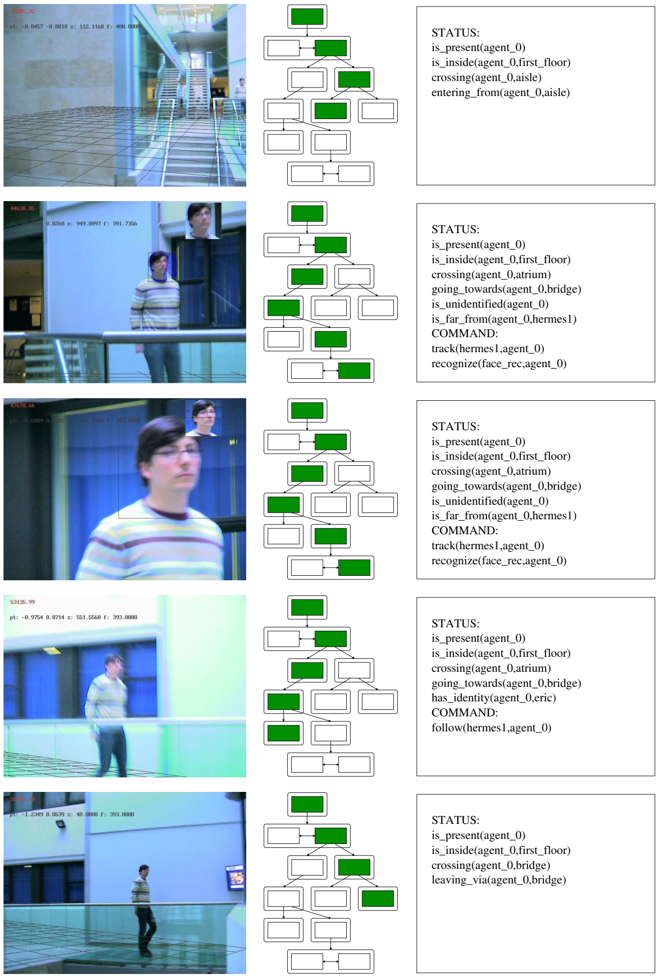
Face recognition is then enabled for seven of the individuals in the database. High level inference in the SGT controls the active cameras to acquire good facial images for identification, and “releases” them once recognition is successful.

Output from an example is given in Fig. 7 which 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 `agent_0` 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 on 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 the TSC. Once the agent is successfully tracked, face images (shown on top-right corner of the second and third snapshot) are sent to the database. A `recognise` command activates the face recogniser, which retrieves these images from the database and tries to determine the agent's identity. In this case, *hermes1* tracks `agent_0` on the second and the third row of the sequence, until the agent is identified as `eric` in the fourth row. The successful recognition causes a `follow` command to be sent to *hermes1*, which zooms-out the camera and simply keeps `agent_0` within its field of view. The TAC still follows the agent when this leaves the atrium through the bridge.

In this experiment, it is important to note the path of the SGT-traversal in the middle column of the figure. While the change of the traversal between the first and the second row, or between the forth 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 forth row is a direct consequence of the high-level commands generated by the system. In particular, the execution of `track` and `recognise` permits the status change of `agent_0` from `is_unidentified` to `has_identity`, and the consequent traversal of a different branch of the SGT.

Figure 8 highlights an advantage of our approach. Within a populated environment, the target that can be most reliably identified is selected for identification. Two occluding targets are not chosen for identification as they are not moving towards an exit in the direction of the camera. The supervisor tracker provides the target position which – at the time the command is issued – is occluded by other targets. The camera zooms for an acquisition of the target, which happens right after the command has been issued. Initial face images are acquired once the occluding persons separate, and by virtue of the robustness of the level set tracker employed, tracking continues through occlusion. Once the target reappears, another face image is acquired and the recognition process started. However, due to the large proportion of outliers in this sequence, the identification was incorrect.



```
STATUS:
is_present(agent_0)
is_inside(agent_0,first_floor)
crossing(agent_0,aisle)
entering_from(agent_0,aisle)
```

```
STATUS:
is_present(agent_0)
is_inside(agent_0,first_floor)
crossing(agent_0,atrium)
going_towards(agent_0,bridge)
is_unidentified(agent_0)
is_far_from(agent_0,hermes1)
COMMAND:
track(hermes1,agent_0)
recognize(face_rec,agent_0)
```

```
STATUS:
is_present(agent_0)
is_inside(agent_0,first_floor)
crossing(agent_0,atrium)
going_towards(agent_0,bridge)
is_unidentified(agent_0)
is_far_from(agent_0,hermes1)
COMMAND:
track(hermes1,agent_0)
recognize(face_rec,agent_0)
```

```
STATUS:
is_present(agent_0)
is_inside(agent_0,first_floor)
crossing(agent_0,atrium)
going_towards(agent_0,bridge)
has_identity(agent_0,eric)
COMMAND:
follow(hermes1,agent_0)
```

```
STATUS:
is_present(agent_0)
is_inside(agent_0,first_floor)
crossing(agent_0,bridge)
leaving_via(agent_0,bridge)
```

Fig. 7. Target tracking and identification.

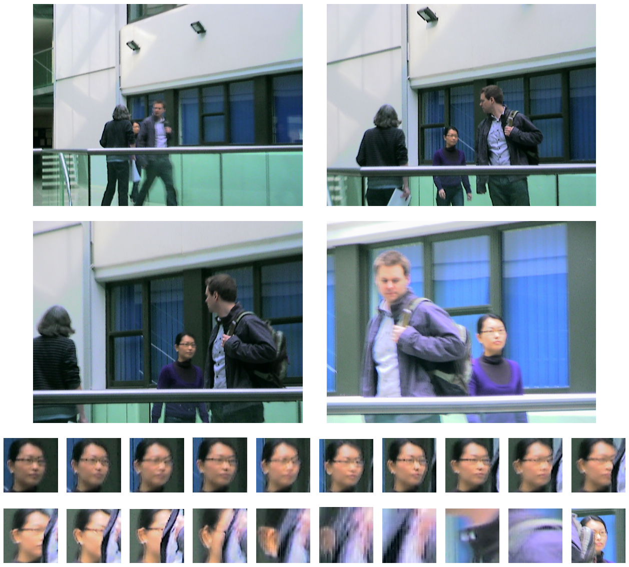


Fig. 8. Target tracking and identification through occlusion works, but yields mediocre results due to brittle PCA.

VII. Conclusions

We have described a system architecture which facilitates linking of high-level inference to sensing actions. In particular we have illustrated this system in the context of surveillance, showing how high level inference can link low-level processes such as target detection and visual tracking with a process for recognition of uncooperative subjects.

In future work, we plan to exploit the capabilities of the fuzzy logic the reasoning engine rests upon. For this we map probabilities (e.g. as returned from the visual tracking algorithm) to fuzzy degrees of belief. Unfortunately, these are simply thresholded by the current inference mechanism to traverse the SGT in a depth-first fashion. We are thus investigating a breadth-first traversal to enable selection from multiple, competing hypotheses on the same specialisation layer.

Regarding the face recognition results, we have shown a real-time, real-world application of the affine hull method proposed by Cevikalp and Triggs [4]. We have furthermore investigated the lack of improvement due to the robust PCA. A relatively recent development provides a principled approach to handling gross outliers in the data [16], and promising initial performance on our data. We are furthermore interested in providing simpler means to acquisition of previously unseen targets, which leads our future work towards incremental clustering methods.

REFERENCES

[1] N. Bellotto, E. Sommerlade, B. Benfold, C. Bibby, I. Reid, D. Roth, L. V. Gool, C. Fernández, and J. Gonzàlez, "A distributed camera system for multi-resolution surveillance," in *Third ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC 2009)*, 2009.

[2] K. H. Schäfer, "Unscharfe zeitlogische Modellierung von Situationen und Handlungen in der Bildfolgenauswertung und Robotik," Ph.D. dissertation, Fakultät für Informatik der Universität Karlsruhe, 1996, (in German).

[3] 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, 2000, pp. 1–6.

[4] H. Cevikalp and B. Triggs, "Face recognition based on image sets," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[5] N. Krahnstoever, T. Yu, S.-N. Lima, and K. Patwardhana, "Collaborative control of active cameras in large-scale surveillance," in *Multi-Camera Networks: Principles and Applications*, H. Aghajan and A. Cavallaro, Eds. Elsevier Science Inc., 2009, pp. 165–188.

[6] C. Soto, B. Song, and A. K. R. Chowdhury, "Distributed multi-target tracking in a self-configuring camera network." in *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2009, pp. 1486–1493.

[7] T. E. Boult, R. J. Micheals, X. Gao, and M. Eckmann, "Into the woods: Visual surveillance of noncooperative and camouflaged targets in complex outdoor settings." in *Proc. of the IEEE, 89(10)*, October 2001, pp. 1382–1402.

[8] D. Hall, J. Nascimento, P. Ribeiro, E. Andrade, P. Moreno, S. Pesnel, T. List, R. Emonet, R. B. Fisher, S. J. Victor, and J. L. Crowley, "Comparison of target detection algorithms using adaptive background models," in *International workshop on Performance evaluation of Tracking and Surveillance*, 2005.

[9] N. Bellotto and H. Hu, "Computationally efficient solutions for tracking people with a mobile robot: an experimental evaluation of Bayesian filters," *Autonomous Robots*, vol. 28, pp. 425–438, 2010.

[10] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[11] C. Bibby and I. Reid, "Robust real-time visual tracking using pixel-wise posteriors," in *Proceedings of the 2008 European Conference on Computer Vision*, 2008.

[12] K. H. Schäfer, *Limette User Manual*, Universität Karlsruhe, 1997.

[13] P. Phillips, W. Scruggs, A. O'Toole, P. Flynn, K. Bowyer, C. Schott, and M. Sharpe, "FRVT 2006 and ICE 2006 large-scale experimental results," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 5, pp. 831 –846, May 2010.

[14] M. Haag and H.-H. Nagel, "Incremental recognition of traffic situations from video image sequences," *Image and Vision Computing*, vol. 18, no. 2, pp. 137–153, 2000.

[15] N. Kwak, "Principal component analysis based on l1-norm maximization," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 30, no. 9, pp. 1672–1680, 2008.

[16] E. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" Department of Statistics, Stanford University, Tech. Rep. 2009-13, 2009.