This document is published in:

Expert Systems with Applications (2011), 38 (6), pp. 7494–7510.

DOI: 10.1016/j.eswa.2010.12.118

Ontology-based context representation and reasoning for object tracking and scene interpretation in video

Juan Gómez-Romero *, Miguel A. Patricio, Jesús García, José M. Molina

Department of Computer Science, Applied Artificial Intelligence Group, University Carlos III of Madrid, Av. de la Universidad Carlos III, 22, 28270 Colmenarejo, Madrid, Spain

ARTICLE INFO

Keywords:
Object tracking
Information Fusion
Context aware systems
Ontologies
Rules

ABSTRACT

Computer vision research has been traditionally focused on the development of quantitative techniques to calculate the properties and relations of the entities appearing in a video sequence. Most object tracking methods are based on statistical methods, which often result inadequate to process complex scenarios. Recently, new techniques based on the exploitation of contextual information have been proposed to overcome the problems that these classical approaches do not solve. The present paper is a contribution in this direction: we propose a Computer Vision framework aimed at the construction of a symbolic model of the scene by integrating tracking data and contextual information. The scene model, represented with formal ontologies, supports the execution of reasoning procedures in order to: (i) obtain a high-level interpretation of the scenario; (ii) provide feedback to the low-level tracking procedure to improve its accuracy and performance. The paper describes the layered architecture of the framework and the structure of the knowledge model, which have been designed in compliance with the JDL model for Information Fusion. We also explain how deductive and abductive reasoning is performed within the model to accomplish scene interpretation and tracking improvement. To show the advantages of our approach, we develop an example of the use of the framework in a video-surveillance application.

1. Introduction

In Computer Vision, tracking is the problem of estimating the number of objects in a continuous scene, together with their instantaneous locations, kinematic states, and other characteristics. Tracking has been traditionally tackled by applying statistical prediction and inference methods. Unfortunately, basic numerical methods have proved to be insufficient when dealing with complex scenarios that present interactions between objects (e.g. occlusions, unions, or separations), modifications of the objects (e.g. deformations), and changes in the scene (e.g. illumination). These events are hard to manage and frequently result in tracking errors, such as track discontinuity, inconsistent track labeling, inconsistent track size, etc. (Yilmaz, Javed, & Shah, 2006).

In addition, other difficulties arise as a result of the imperfect functioning of video cameras, the wide range of possible scenes, and the necessity of combining data acquired by different sensors. This last matter is a subject of study of the Data and Information Fusion research area, which is specially concerned to Computer Vision and tracking. Fusion techniques combine data from multiple sensors and related information to achieve more specific inferences than could be achieved by using a single, independent sensor (Hall & Llinas, 2009). In most cases, tracking requires the combination of data sources, and analogously, visual inputs are pervasive in fusion applications. Fusion processes are classified according to the JDL (Joint Directors of Laboratories) model, the prevailing theory to describe fusion systems (Steinberg & Bowman, 2004). JDL establishes a common vocabulary to facilitate communication and understanding among the heterogeneous specialists interested in Information Fusion, and defines a functional model to structure fusion applications.

A solution to overcome tracking issues is to provide the image-processing algorithms with knowledge about the observed entities not directly obtained by the cameras. For instance, predefined background patterns (Boult, Micheals, Gao, & Eckmann, 2001), human-body models (Haritaoglu, Harwood, & David, 2000; Huang & Huang, 2002), 2D and 3D object models (Remagnino et al., 1997), color spaces (Orwell, Remagnino, & Jones, 1999), and Soft Computing techniques (Patricio et al., 2008) have been used to improve the performance of tracking systems. According to Dey and Abowd (2000), this information can be regarded as *context*, because it is (implicitly or explicitly) used to characterize the situation of the entities. In tracking, context can be considered as encompassing any external knowledge used to complete the quantitative data about the scene computed by straightforward image-analysis algorithms: (i) information about the scene environment (structures, static objects, illumination and behavioral characteristics, etc.);

* Corresponding author. Tel.: +34 91 8561327; fax: +34 91 8561220.

E-mail addresses: jgromero@inf.uc3m.es (J. Gómez-Romero), mpatrici@inf.uc3m.es (M.A. Patricio), jgherrer@inf.uc3m.es (J. García), molina@ia.uc3m.es (J.M. Molina).

(ii) information about the parameters of the recording (camera, image, and location features); (iii) information previously calculated by the vision system (past detected events); (iv) user-requested information (data provided by human users) (Bremond & Thonnat, 1996).

Most of the previous approaches in the Computer Vision literature, however, have only taken into account local context. In these cases, the context of an object is a measure computed from the pixel values of the surrounding objects that is interpreted according to a priori criteria (Yang, Wu, & Hua, 2009). Though effective, these techniques are hardly extensible to different application domains. In contrast, cognitive approaches propose building a symbolic model of the world, expressed in a logic-based language, which abstractly represents the scene objects and their relations (Vernon, 2008). Such model can be regarded as the mental representation of the scene gained by cognitive software agents. It may include both perceptions and more complex contextual information. Cognitive approaches are more robust and extensible, but they require the development of suitable interpretation and reasoning procedures, which is not assumable or even possible in all cases.

Recent researchers in Information Fusion have recognized the advantages of cognitive situation models, and have pointed out the importance of formal context knowledge to achieve scene understanding. The last revision of the JDL specification highlights the importance of context knowledge (Steinberg & Bowman, 2009), specially when visual inputs are to be interpreted (Steinberg & Rogova, 2008). Conceptual models to acquire, represent, and exploit formal knowledge in the fusion loop have been proposed, with a special interest in ontologies (Nowak, 2003). An ontology is a representation of the mereological aspects of a reality, created from a common perspective and expressed in a formal language (Gruber, 1993), such as the standard Web Ontology Language OWL (McGuiness & van Harmelen, 2004). Nevertheless, practical implementations of (visual) data fusion systems based on formal knowledge representations still scarce.

The present paper is a contribution in this direction. We propose a Computer Vision framework that, by relying on an ontology-based scene representation model, combines contextual and sensor data to accomplish two tasks:

1. Construction of a high-level symbolic interpretation of the situation (i.e. recognition of events).
2. Improvement of the performance of object tracking procedures (i.e. correction of the errors that appear in a classical tracking procedure).

The framework applies logical reasoning from elemental tracking data, obtained by a classical tracker, to construct an ontological model of the objects and the activities happening in the observed area. Once this interpretation of the scene is created, the framework carries out subsequent reasoning procedures with it to detect and predict tracking errors. When an error is discovered, feedback is sent to the tracker in order to attune the low-level image-processing algorithms.

The framework is designed as a processing layer on top of the tracking software. The inputs of the additional layer are the values calculated by the tracking algorithm (which are considered *perceptions*) and additional knowledge about the scene in the form of static object descriptions, reasoning rules, etc. (which are considered *context*). The outputs of the layer are the high-level representation of the scene (*interpretation*) and the recommended actions that should be executed by the tracker (*feedback*).

As a matter of example of the processing in the framework, let us suppose that a moving entity is detected by the tracking algorithm (perception). If this object is new, has a predetermined size, and is close to a door (context), it can be deduced that a new person has entered the room (interpretation). In view of that, the tracking algorithm can be recommended to create a new track associated to this object (feedback).

1.1. Contributions and structure of the paper

In this paper, we describe the epistemological, functional, and structural characteristics of our Computer Vision framework. The framework provides a theoretical and practical reference for the design of cognitive man–machine vision systems that exploit context knowledge. It can be applied in different vision applications by adapting the architecture and extending the general model with domain-specific knowledge.

The framework solves various challenges that have been highlighted as crucial in related proposals (Lambert, 2003): (i) to discern what knowledge should be represented; (ii) to determine which representation formalisms are appropriate; (iii) to elucidate how acquired and contextual inputs are transformed from numerical measures to symbolic descriptions. For that reason, in this paper we focus on the specification of: (i) the structure and the contents of the knowledge model; (ii) the ontologies to encode this knowledge; (iii) the inference procedures to interpret the scene in terms of symbolic descriptions; (iv) the inference procedures to provide feedback to the low-level image-processing algorithms. We illustrate the advantages of our approach with an example of its use in a video-surveillance application, which interestingly can be extended to other domains.

The remainder of the paper is organized as follows. In Section 2, we overview some related work pertaining to the use of formal context knowledge models in Information Fusion and Computer Vision, particularly in ontology-based and surveillance applications. In Section 3, we provide an introduction to knowledge representation and reasoning with ontologies. Next, in Section 4, w e describe the layered architecture of the framework. In Section 5, we depict the composition of the interrelated ontologies encompassed by the knowledge model of the contextual layer. In Section 6, we clarify the details of the reasoning processes performed with the contextual model. In Section 7, we exemplify the functioning of the system with a practical case on surveillance. Finally, the paper concludes with a discussion on the results and plans for future research work.

2. Related work

Several knowledge representations to create scene models have been studied in Cognitive Vision. Most of them stress the importance of bridging the gap between raw images and symbolic interpretation, which is known as the grounding problem (Pinz et al., 2008). Nevertheless, logic-based languages have received modest interest, in spite of their notable representation and reasoning advantages. Moreover, in this case most approximations have used ad hoc first order logic representation formalisms (Brdiczka, Yuen, Zaidenberg, Reignier, & Crowley, 2006), which have certain drawbacks: they are hardly extensible and reusable, and reasoning with unrestricted first order logic models is semi-decidable.

Ontologies, in turn, overcome these limitations. In the last years, various works proposing models and methods to exploit context knowledge represented with ontologies have been pub-lished. These researches can be classified according to the four lev-els defined by the JDL model for Information Fusion (Steinberg & Bowman, 2004; Llinas et al., 2004). The canonical JDL model establishes four operational levels in the transformation of input signals to decision-ready knowledge, namely: signal feature assessment (L0); entity assessment (L1); situation assessment (L2); impact assessment (L3); process assessment (L4). Some authors have

discussed the convenience of considering the intermediate information between L1 and L2, namely the L1–L$\frac{1}{2}$ layer (Das, 2008).

At image-data level (i.e. JDL level 0), one of the most important contributions is COMM (Core Ontology for MultiMedia), an OWL ontology to encode MPEG-7 data (Arndt, Troncy, Staab, Hardman, & Vacura, 2008). COMM does not represent high-level entities of the scene, such as people or events. Instead, it identifies the components of a MPEG-7 video sequence in order to link them to (Semantic) Web resources. Similarly, the Media Annotations Working Group of the W3C is working in an OWL-based language for adding metadata to Web images and videos (Lee, Bürger, & Sasaki, 2009).

Other proposals are targeted at modeling video content at object level (i.e. JDL L1). For example, François, Nevatia, Hobbs, Bolles, and Smith (2005) have created a framework for video event representation and annotation. In this framework, VERL (Video Event Representation Language) defines the concepts to describe processes (entities, events, time, composition operations, etc.), and VEML (Video Event Markup Language) is a XML-based vocabulary to markup video sequences (scenes, samples, streams, etc.). VEML 2.0 has been partially expressed in OWL. The limitation in the number of entities represented in VEML 2.0 reduces its usefulness, as it is discussed by Westermann and Jain (2007), who present a framework that supports representation of uncertain knowledge. An approach that stands halfway between data and object level is due to Kokar and Wang (2002). In this research work, the data managed by the tracking algorithm is represented symboli-cally, in a similar fashion as we do; however, our approach addi-tionally considers higher level knowledge, inferred by abductive reasoning.

Regarding high-level Information Fusion (i.e. JDL L2 and L3 levels), Little and Rogova (2009), study the development of ontologies for situation recognition, and propose a methodology to create domain-specific ontologies for Information Fusion based on the upper-level ontology BFO (Basic Formal Ontology) and its sub-ontologies SNAP and SPAN (for *endurant* entities and *perdurant* processes, respectively). Other contribution is STO (Situation Theory Ontology), which encodes Barwise's situation semantics (Kokar, Matheus, & Baclawski, 2009). With a special focus on Computer Vision, Neumann and Möller (2008), study scene interpretation based on Description Logics and reasoning with an inference engine. These authors also distinguish between lower level representations (GSD, Geometrical Scene Description) and higher level interpretations, as we do. A practical application in surveillance is shown by Snidaro, Belluz, and Foresti (2007), who have developed an OWL ontology enhanced with rules to represent and reason with objects and actors.

All these research works focus on contextual scene recognition, but as mentioned in Section 1, it is also interesting to apply this knowledge to refine image-processing algorithms (which corresponds to JDL L4), and particularly trackers. A preliminary approach to this topic has been presented in Sánchez, Patricio, García, and Molina (2009). The present paper enhances this work by offering an integral framework for object tracking, instead of a particular solution, and by implementing extensible and modular knowledge models represented with standard ontologies, which can be reused in different domains.

3. Knowledge representation and reasoning with ontologies

In this section, we provide a brief introduction to knowledge representation with Description Logic ontologies focused on the OWL language. We will assume the reader is familiar with these topics. For a comprehensive explanation, see Baader, Calvanese, McGuinness, Nardi, and Patel-Schneider (2003).

3.1. Representation

Ontologies are highly-expressive, logic-based knowledge models aimed to describe a domain with an automatically processable language (Studer, Benjamins, & Fensel, 1998). Description Logics (DLs) are a family of logics to represent structured knowledge (Baader, Horrocks, & Sattler, 2008) that have proved to be suitable ontology languages (Baader, Horrocks, & Sattler, 2005). Ontology languages are usually (equivalent to) a decidable DL, as it occurs in the case of the previously mentioned OWL (Horrocks & Patel-Schneider, 2004), and its successor OWL 2 (Hitzler, Krötzsch, Parsia, Patel-Schneider, & Rudolph, 2008). Depending on the expres-sivity of the used language, ontological representations range from simple taxonomies defining term hierarchies to sophisticated concept networks with complex restrictions in the relations.

DLs are structured in levels, each named with a string of capital letters that denote the allowed expressions in it. In general, having more constructors in a logic means that it is more expressive, and consequently, the computational complexity (of reasoning procedures) is greater. A commonly-used DL is $\mathcal{SHOIN}(\mathrm{D})$, which is in practice equivalent to OWL DL, the most expressive subset of OWL that is decidable. OWL 2 is compatible with $\mathcal{SROIQ}(\mathrm{D})$, which extends $\mathcal{SHOIN}(\mathrm{D})$ with various supplementary constructors. In the remaining sections of this paper, we use $\mathcal{SHOIQ}(\mathrm{D})$, an intermediate logic between OWL and OWL 2, and assume OWL 2 as ontology language.

A DL ontology is developed from the following elements: concept or classes (C, D), which represent the basic ideas of the domain; instances or individuals (a, b), which are concrete occurrences of concepts; relations, roles, or properties (R, T), which represent binary connections between individuals or individuals and typed values (t, u). Complex concept and relation expressions can be derived inductively from atomic primitives by applying the constructors defined by the logic. The valid constructors in $\mathcal{SHOIQ}(\mathrm{D})$ are presented in Table 1.

Table 1
Syntax of complex concepts and roles in $\mathcal{SHOIQ}(\mathrm{D})$.

Constructor	DL syntax	OWL syntax
Role constructors		
Atomic role	R_A	`owl:ObjectProperty`
Inverse role	R^-	`owl:inverseOf`
Concrete role	T	`owl:DatatypeProperty`
Universal role	U	`owl:TopObjectProperty` `owl:topDataProperty`
Concept constructors		
Top concept	$\top$	`owl:Thing`
Bottom concept	$\bot$	`owl:Nothing`
Atomic concept	A	`owl:Class`
Concept negation	$\neg C$	`owl:complementOf`
Concept intersection	$C \sqcap D$	`owl:intersectionOf`
Concept union	$C \sqcup D$	`owl:unionOf`
Universal quantification	$\forall R.C$	`owl:allValuesFrom`
Existential quantification	$\exists R.C$	`owl:someValuesFrom`
Nominals	$\{a, b, \ldots\}$	`owl:oneOf`
Qualified number restriction	$\geqslant n\ S.C$	`owl:maxQualifiedCardinality`[a]
	$\leqslant m\ S.C$	`owl:minQualifiedCardinality`[a]
	$=q\ S.C$	`owl:qualifiedCardinality`[a]
Universal quantification	$\forall T.u$	`owl:allValuesFrom`
Existential quantification	$\exists T.u$	`owl:someValuesFrom`
Nominals	$\{t_1, t_2, \ldots\}$	`owl:oneOf`
Qualified number restriction	$\geqslant n\ T.u$	`owl:maxQualifiedCardinality`[a]
	$\leqslant m\ T.u$	`owl:minQualifiedCardinality`[a]
	$=q\ T.u$	`owl:qualifiedCardinality`[a]

[a] $n, m, q \in \{0, 1, 2, \ldots\}$

Domain knowledge is represented by asserting axioms (O), which establish restrictions over concepts, instances, and relations, describing their attributes by delimiting their possible realization. A DL ontology is a triple $\mathcal{K} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, where $\mathcal{T}$ (the TBox) and $\mathcal{R}$ (the RBox) contain terminological axioms (respectively, axioms about concepts and roles), and $\mathcal{A}$ (the ABox) contains extensional axioms (axioms about individuals). In $\mathcal{SHOIQ}(\mathrm{D})$, valid axioms are (not exhaustively):

- $C \sqsubseteq D$ denotes that C is included in D (or D subsumes C) (`rdfs:subClassOf`)
- $C \equiv D$ denotes that C and D are equivalent (`owl:equivalentClass`)
- $R_1 \sqsubseteq R_2$ denotes that R_1 is a subset of R_2 (`rdfs:subPropertyOf`)
- $R_1 \equiv R_2$ denotes that R_1 and R_2 are equivalent (`owl:equivalentProperty`)
- (a:C) denotes that a is an instance of C (`C rdf:ID=a`)
- ((a,b):R) denotes that a and b are related through R (`R rdf:resource=b`)
- ((a,t):T) denotes that a has value t through data-valued property T (`T rdf:datatype=<type> t`)

3.2. Reasoning

Reasoning with ontologies is an automatic procedure that infers new axioms which have not been explicitly included in the knowledge base but are logical consequences of the represented axioms. In general, an axiom O is said to be *entailed* by the ontology $\mathcal{K}$ if every possible realization of the ontology satisfies the axiom. Entailment is denoted as $\mathcal{K} \models O$. Reasoning in DLs can be carried out with concepts in the TBox, individuals in the ABox, or TBox concepts and ABox individuals together.

The basic reasoning task regarding ontology concepts is concept satisfiability. Intuitively, a concept is *satisfiable* if it is not contradictory of the rest of the knowledge in the ontology. Another important task is concept *subsumption*, which infers if a concept is more general than another concept. The basic inference task with ontology individuals is to test if an axiom of the ABox is not contradictory of the other axioms in the ontology (i.e. it is *satisfiable* or *consistent*), or particularly in the ABox. If the assert is a membership axiom (a:C), this test is called *instance checking*.

These tasks can be transparently executed with DL inference engines, which allow loading, querying, and reasoning with ontologies expressed in OWL and/or OWL 2 (and particularly, with $\mathcal{SHOIQ}(\mathrm{D})$ ontologies). Pellet (Sirin, Parsia, Cuenca Grau, Kalyanpur, & Katz, 2007) and RACER (Häarslev & Möller, 2001) are two examples of DL reasoners. The computational complexity of the reasoning procedures depends on the expressivity of the considered language. For general $\mathcal{SHOIQ}(\mathrm{D})$ ontologies, this complexity is NEXPTIME-complete, which is quite high. Fortunately, worst-case inferences are infrequent, and reasoners have been highly optimized to offer good execution times in the typical cases.

It should be underlined that, when reasoning with DL ontologies, the open world assumption stands. The open world assumption supposes that the set of axioms in a knowledge base is not complete, and consequently, new knowledge cannot be inferred inductively. This feature makes ontologies a monotonic formalism (Bossu & Siegel, 1985), which means that it is not possible to update, retract, or remove previous knowledge while reasoning. This is an important restriction in our case, since we will rely on abductive reasoning, which implies knowledge update, to interpret the perceived scenarios. Nevertheless, simplified versions of this assumption are implemented by reasoning engines, and partial abduction with ontologies is therefore possible.

Likewise, rule-based reasoning is not directly supported by OWL, but several extensions have been proposed. One of the most interesting is SWRL (Semantic Web Rule Language) (Horrocks & Patel-Schneider, 2004), which allows deductive inference within OWL ontologies and is widely supported. Rule-based formalisms can be used with limitations, since reasoning with models combin-ing rules and OWL is decidable only under certain safety restric-tions (Motik, Sattler, & Studer, 2005). These restrictions are, however, relaxed in certain specific rule languages implemented by reasoning engines, as it occurs in RACER, which is used in this paper.

4. Framework architecture

The architecture of our Computer Vision framework is depicted in Fig. 1. The schema shows the tracking system (the GTL, *general tracking layer*) and, built upon it, the context-based extension (the CL, *context layer*). The GTL and the CL communicate through the GTL/CL interface. In this section, we describe the functioning of the GTL, the CL, and the GTL/CL interface.

The GTL is a classical tracker; we will assume the tracker described in Patricio et al. (2008), but the framework can be used in combination with any other tracking software adapted to work

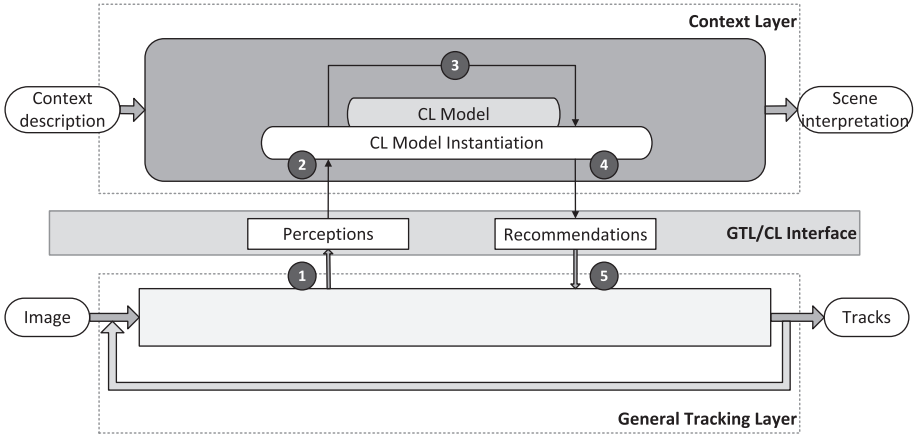


Fig. 1. Functional architecture of the framework.

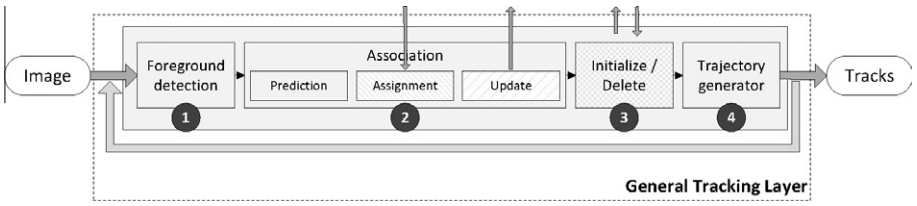


Fig. 2. Functional architecture of the GTL.

with the interface. The CL manages the CL model, i.e. the ontological representation of the scene (including context and perceived knowledge), and carries out procedures to update and reason with it. The CL model is implemented as a set of interrelated ontologies. The terminological axioms of the CL ontologies establish a controlled vocabulary to describe scenes. The current state of the scene is represented with instances of these concepts and relations. The GTL/CL interface, which guarantees interoperability and independence between both layers, includes methods to update the CL model according to the GTL tracking information, and to consult recommendations calculated by the CL.

The framework works as follows:

1. The GTL calls the GTL/CL interface methods when tracks are created, deleted, or updated.
2. The GTL/CL interface transforms GTL track data into CL ontological instances, and insert them into the CL model.
3. Updated track information triggers reasoning processes in the CL that, supported by context knowledge, update the whole interpretation of the scene.
4. Recommendations are created by reasoning with the current scene model and a priori knowledge.
5. The GTL calls the GTL/CL interface methods into consult recommendations from the CL, and converts them to concrete actions.

The interaction between the GTL and the CL can be either synchronous or asynchronous. In synchronous mode, the steps 1–5 are sequentially executed in each iteration of the GTL. The GTL calls the interface methods to update the CL model and, when the processing in the CL is finished, it invokes the recommendation query methods. Recommendations are processed by the GTL as they are produced by the CL. In asynchronous mode, calls to the interface query methods in step 5 can be done at any moment, not necessarily after the sequence 1–4. To implement asynchrony, a queue of recommendations has been implemented in the GTL/CL interface. This way, low-level tracking and high-level reasoning are decoupled. Pending recommendations, marked with a time stamp, are placed in the queue when reasoning in the CL finishes. The GTL retrieves recommendations from the queue, which can be accepted or disregarded according to a priority ordering policy.

4.1. General tracking layer

The GTL is a software program that sequentially executes various image-processing algorithms with a video sequence in order to track all the targets within the local field of view. The GTL aims at segmenting video frames, distinguishing moving objects from background and labeling them consistently. GTLs are usually arranged in a pipelined structure of several modules, as shown in Fig. 2, which correspond to the successive stages of the tracking process. These steps are carried out for each frame of the input video sequence.

The main modules of our GTL implementation are dedicated to: (1) movement detection (background and foreground detection); (2) blob-track[1] association, which includes prediction, assignment, and update; (3) track creation and deletion; (4) trajectory generation. The modules have been developed in such a way that implementations of different methods for detection, association, etc. can be plugged in the pipeline.

The detector of moving objects (1) gives as a result a list of the blobs that are found in a frame. For each blob, position and size values are calculated and stored in a data structure. The association process (2) solves the problem of blob-to-track multi-assignment: once blobs have been identified, the association process finds a correspondence between detected blobs and existing tracks. A blob can be assigned to none, one, or more tracks in an iteration of the tracking procedure. Unassigned blobs may be removed by the algorithm. Association errors (e.g. blobs are mistaken for the background, close or overlapping blobs are not correctly assigned), which are frequent in complex scenarios, must be minimized. The association of a blob to a track may mean changes in track values, which are processed next. Association does not create or delete tracks; instead, these tasks are performed in the initialize/delete stage (3), if needed. A simple criterion to create a new track is when a blob in an image region has not been associated to any track in the last frames, a situation that can be interpreted as the apparition of a new object in the scene. In the last step (4), track trajectories are recorded and analyzed.

In our framework, during the tracking process, the GTL invokes methods of the GTL/CL interface when track information changes, i.e. in the update and initialize/delete stages. These calls request the interface to renew the symbolic CL model according to the recent perceptions. Since new track data is passed to the interface as numerical data types, the interface must transform them to ontology instances and then modify the CL model. The updates of the track information in the CL model trigger additional reasoning processes, as explained in the next section. Reasoning ultimately results in the generation of a scene interpretation and recommendations. Recommendations must be expressed in such a way that the GTL is able to convert them in concrete operations. Calls to the retrieval interface methods have been included in the assignment and initialization/delete stages of the GTL.

4.2. Context layer

The CL receives from the GTL tracking data, processes it, and provides as a result a set of recommendations or actions that should be performed by the GTL. Since the CL uses context knowledge to interpret the scene, it additionally has context information as input. Numerical inputs provided by the GTL to the GTL/CL interface are called hard inputs, whereas contextual and human entries are named soft inputs. Respectively, readable scene interpretations are called soft outputs, and recommendations, interpretable by the GTL, are named hard outputs.

[1] A blob is a set of pixels that form a connected region. A track is a low-level representation of a moving entity. It is represented as a single blob or as a set of related blobs with properties: size, color, velocity, etc.

A schema of the functional architecture of the CL is shown in Fig. 3. The figure depicts the key novelty of our approach: informa-tion in the CL, both contextual and perceived, is represented with a formal knowledge model implemented as set of OWL ontologies.

On the left side of Fig. 3, the schema shows the structure of the ontology-based representation model, organized in various levels from less abstract (track data) to more abstract (activity description and evaluation). This organization has been designed in accordance with the JDL model. JDL, resulting from the consensus of the main experts in Information Fusion, has proved its usefulness as conceptual schema to structure fusion systems. The rational systematization of JDL provides a very convenient support to make the CL model understandable and modular, in such a way that it can be easier extended and reused.

The different types of data, information, and knowledge of the CL are structured in the following layers:

- Camera data (L0). To be precise, this data is not managed in the CL, but in the GTL. Actually, the live or recorded sequence provided by the cameras (in some processable video format) is the input which will be analyzed by the GTL.
- Tracking data (L1). Entity data in our framework corresponds to the output of the tracking algorithm represented with ontological terms after the abstraction step. Instances of tracks and track properties (color, position, velocity, etc.), frames, etc. are created in this initial representation of the moving objects of the scene.
- Scene objects (L1–L$\frac{1}{2}$). Scene objects are the result of making a correspondence between existing tracks and possible scene objects. For example, a track can be inferred to correspond to a person (possibly by applying context knowledge). Scene objects also include static elements, which may be defined a priori. Properties and relations of these objects are also considered, which makes this knowledge stand between L1 and L2.
- Activities (L2). Activities or events are the behaviors of the objects of the scene. Activities describe relations between objects which last in time, for example, grouping, approaching, or picking/leaving an object. Activity recognition is achieved by interpreting object properties in context.
- Impacts and threats (L3). Activities may be associated with a cost or threat value after evaluation. This measurement is quite interesting in applications such as surveillance, because it consistently relates activities and evaluations of the risk.
- Feedback and process improvement (L4). JDL groups in L4 operations targeted at enhancing fusion processes at different levels. This is the case of the modules of our framework that provide suggestions to the tracking procedure. L4 knowledge consists of the representation of these suggestions.

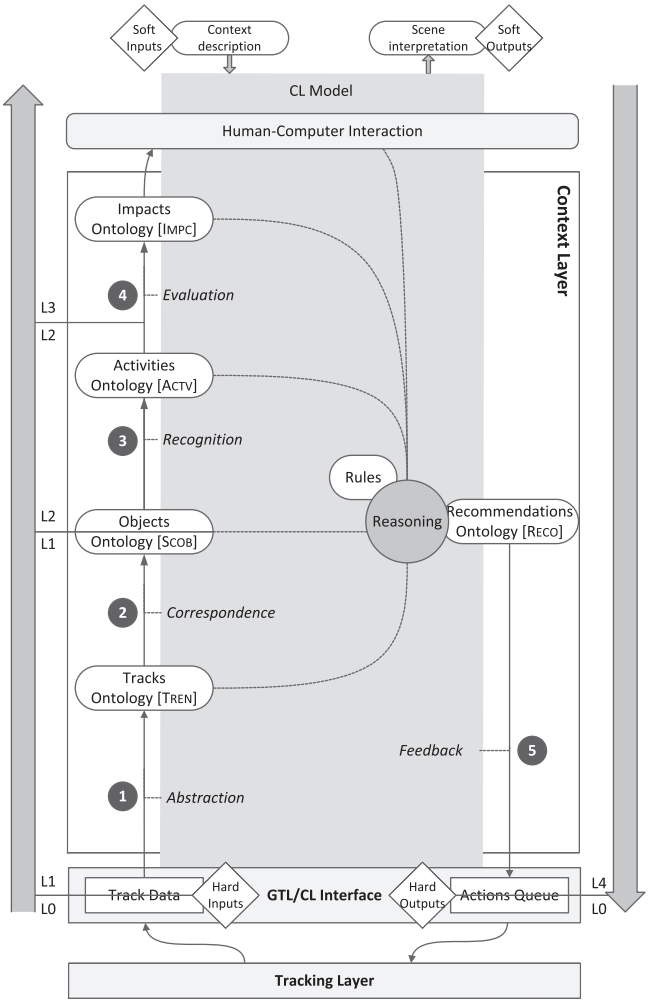


Fig. 3. Functional architecture of the CL.

Each one of these levels corresponds to an ontology in the CL knowledge representation model, namely: TREN, SCOB, ACTV, and IMPC, and a set of reasoning rules managed by the reasoner. The development of these ontologies is explained in detail in Section 5, whereas rules are presented in Section 6.

Interpretation of acquired data with contextual knowledge in the CL is performed in four corresponding steps: (1) abstraction of acquired tracking data; (2) correspondence between tracks and objects; (3) recognition of activities from object descriptions; (4) evaluation of threats. In the first stage (1), tracking data provided by the tracking system is transformed into ontology instances by accessing the intermediate GTL/CL interface, which updates the CL abstract scene model. Subsequent symbolic representations are created as a result of reasoning with the input perceptions (steps 2–4), and eventually, activities are recognized and evaluated. This process, which transforms numerical to symbolic data by applying rule-based inference, is depicted in Section 6 and exemplified in Section 7.

The calculation of the recommendations by the CL (5) is performed in parallel to scene interpretation (right side of Fig. 3). This procedure is also detailed in Section 6. Recommendations range from non-concrete suggestions (e.g. take into account that new objects appear frequently in a region of the image) to precise indications (e.g. change covariance matrix of Kalman filter). It is a choice of the application developer to manage recommendations at his convenience, as we show in the example of Section 7.

5. Knowledge representation: CL model contents

As introduced in the previous section, the CL model encompasses various ontologies that represent tracking data, scene objects, activities, impacts, and GTL recommended actions, namely: TREN, SCOB, ACTV, IMPC, and RECO. A simplified schema of the classes and relations of these ontologies is depicted in Fig. 4. The main concepts of the ontologies, which link representations at different levels, are highlighted in gray. For example, SceneObject, which is a L1 concept, is imported by the L2 ontology, which describe activi-ties from object interactions.

We have distinguished in the CL model between the general knowledge (i.e. knowledge that is common to any vision application) and the specific knowledge (i.e. domain-specific knowledge).

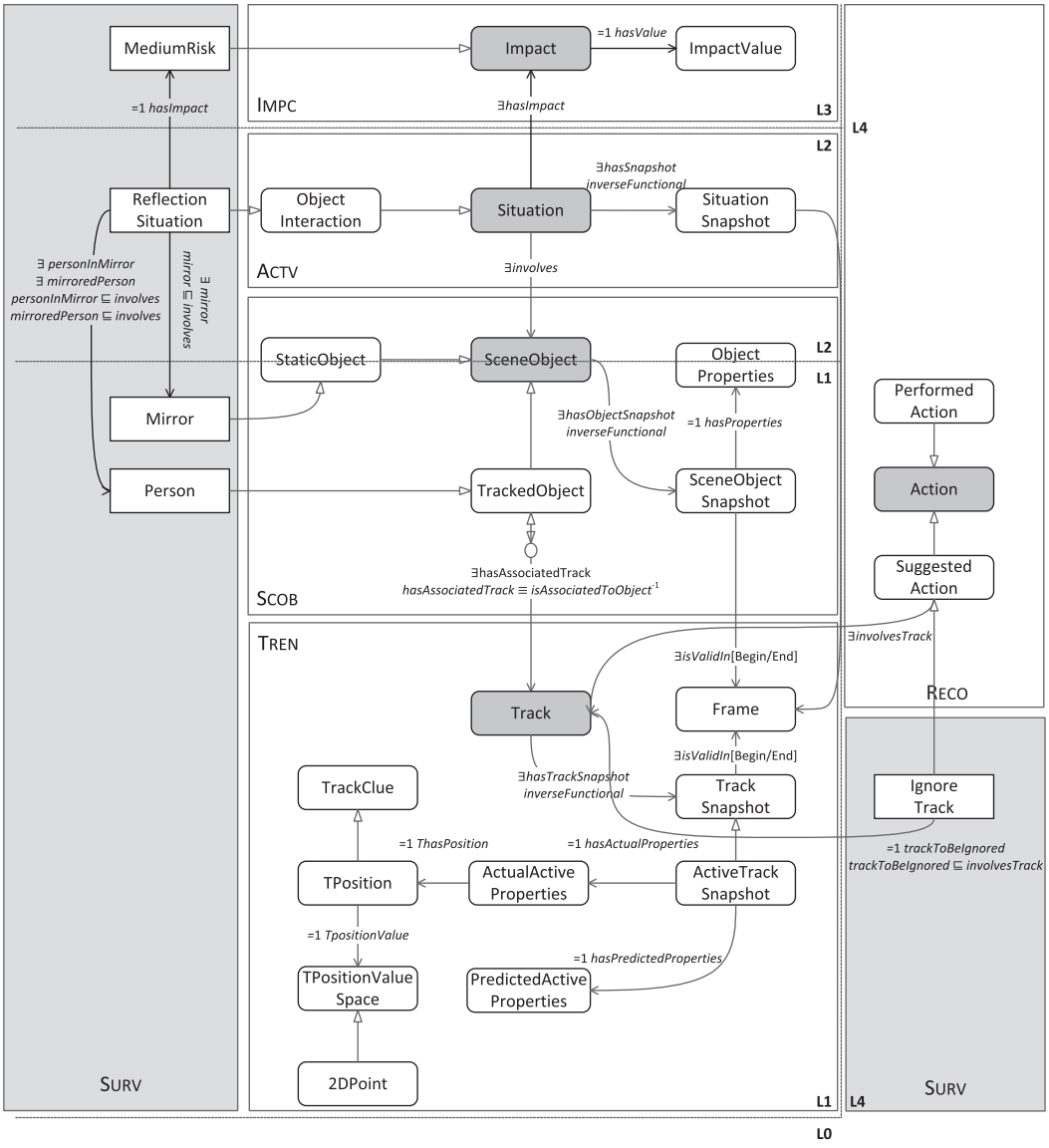


Fig. 4. Excerpt of the CL ontological model.

Accordingly, we provide upper ontologies that contain very general terminological axioms,[2] including reference basic concepts and relations, to be specialized in each application. Fig. 4 shows this distinction: the general ontologies include more abstract knowledge, whereas the specific ontology SURV refines then with concepts, relations, and axioms for an indoor-surveillance application. In this manner, the SCOB ontology defines a generic entrance object concept. In the SURV ontology, a Mirror concept has been created by specialization of the StaticObject concept. This distinction between general and specific knowledge is essential to guarantee the applicability of our framework in different application areas. The physical separation between levels in files can be maintained by the specializing ontology or it can be either flattened for the sake of simplicity.

Next, we explain the structure of the terminological part of the general CL ontologies provided with the framework, i.e. the concept and relation definitions. The creation of the instances of these ontologies as a result of reasoning processes is described in Section 6. The specialization of the general ontologies and the application of the framework to a surveillance scenario is depicted in Section 7.

5.1. Object assessment knowledge: tracking data and scene objects

Object knowledge is represented in the CL model with the ontologies TREN (TRacking Entities) and SCOB (SCene OBjects). These ontologies are used to describe track data and tracked entities information, respectively. TREN is an ontological encoding of the GTL data; it does not add further information to the track values provided by the GTL. SCOB is used to represent scene objects from a cognitive perspective; instead of unspecific tracks, it concretes the type of moving objects, which is necessary to comprehend the scene.

5.1.1. Tracking data

The core concepts in TREN are Frame and Track. A frame is identified by a numerical ID and can be marked with a time stamp using an OWL-Time DateTimeDescription (Hobbs & Pan, 2006).

[2] http://www.giaa.inf.uc3m.es/miembros/jgomez/ontologies/ontologies.html.

With respect to tracks, it is necessary to represent their temporal evolution and not only its state in a given instant. We want to keep all the information related to a track during the complete sequence (activity, occlusions, position, size, velocity, etc.), which changes between frames, and not only its lastly updated values. Therefore, we must associate to each track various sets of property values that are valid only during some frames. To solve this issue, we have followed an ontology design pattern proposed by the W3C Semantic Web Best Practices and Deployment Working Group to define ternary relations in OWL ontologies (Noy & Rector, 2006). We have associated a set of TrackSnapshots to each Track. Each TrackSnapshot, representing track feature values, is asserted to be valid in various frames.

Additionally, track properties must be defined as general as possible, in such a way that they can be easily extended. To solve this issue, we have followed the *qualia* approach, used in the upper ontology DOLCE (Gangemi, Guarino, Masolo, Oltramari, & Schneider, 2002). This modeling pattern distinguishes between properties themselves and the space in which they take values. This way, we have associated properties to ActiveTrackSnapshots, such as TPosition or TSize. TPosition is related with the property TpositionValue to a single value of the TPositionValueSpace. A 2DPoint is a possible TPositionValueSpace. The definition of geometrical entities has been developed according to the proposal by Maillot, Thonnat, and Boucher (2004), which defines primitive concepts such as Point, PointSet, Curve (a subclass of PointSet), or Polygon (a kind of Curve). Additional axioms or rules to calculate complex properties of tracks (e.g. distances), as well as spatial relationships (inclusion, adjacency, etc.), could be considered and created in TREN.

5.1.2. Scene objects

Scene objects are real-world entities that have a visual appearance. The SCOB ontology offers a general terminology to represent scene objects and properties. In contrast to TREN, SCOB is expected to be extended in particular applications. For example, in a surveillance application, concepts such as *person*, *car*, *door*, or *column* will be created to particularize the more general SCOB concepts. SCOB mainly contains L1 knowledge, that is, knowledge about single object without considering interactions between them. However, it may be interesting to represent some relations between objects not strictly pertaining to activities. For this reason, the tracked entities knowledge and the associated context knowledge is in the L1–L$2\frac{1}{2}$ JDL level.

The main concept in SCOB is SceneObject. SceneObject is a concept that includes all the interesting objects in the scene, either dynamic or contextual. Most contextual object instances belong to the StaticObject concept, since their properties usually do not change. Dynamic object instances belong to the TrackedObject concept, which encompasses all the scene objects that have associated a Track (of the TREN ontology, which is imported). Tracked objects are associated to a unique track during their lifetime, though this track can have associated several snapshots. If the same track is recognized as two different objects, two instances of TrackedObject are created.

SceneObjects have properties, e.g. position, illumination, behavior, etc., which may vary in the sequence. Some of the values of these properties may be derived from the associated track, e.g. position values. In order to represent them properly, we have applied the combined snapshot/qualia approach explained for tracks. Thus, we have corresponding SceneObjectSnapshot and ObjectProperties concepts in SCOB. In this way, it is possible to describe the changing properties of an object during its whole life and to add new properties easily. In the current SCOB ontology, objects are assumed to exist in a 2D space, but the adaptation to a 3D space is easy thanks to this design.

5.2. Situation and impact assessment: activities and impacts

The terminological axioms of the ACTV (ACTiVities) ontology provide a vocabulary for describing events in the scene. Scene situations are defined in terms of relations between scene objects expressed in the SCOB ontology, which is imported. For convenience, these relations have been reified as concepts descending from a top Situation concept. Likewise, the IMPC (IMPaCts) ontology is built on top of ACTV and expresses the relation between situations (instances of the Situation concept) and impact evaluations (instances of the Impact concept).

Since the number of possible scenes is countless, only very general activities have been included in ACTV. Domain-specific activities must be created by refinement of the elements of the ACTV ontology. We offer a reference taxonomy based on the upper levels of the ontology for cognitive surveillance presented by Fernández and González (2007). We have also introduced some properties to establish the temporal duration of the situations that follows the same pattern based on snapshots described for the lower levels.

The IMPC ontology, in turn, contains a vocabulary to associate an evaluation value to ACTV Situation instances. This value can be a simple numerical assessment or, more probably, a complex expression suggesting or predicting future actions. To allow the representation of different impact evaluations, the qualia approach has been applied.

5.3. Process assessment: recommendations

The JDL process assessment level encompasses actions aimed at improving the quality of the acquired data and enhancing the performance of the Fusion procedure. In our case, process assessment knowledge includes certain meta-information about the functioning of the framework that can be used to improve it. More precisely, the RECO (RECOmmendations) L4 ontology includes concepts and relations that represent actions of the GTL, either performed by it or suggested to be carried in the near future.

The main concept of the RECO ontology is Action, which generically includes all the actions that the GTL can execute. Actions are classified in SuggestedActions, which are calculated by the CL, and PerformedActions, if they have been previously executed by the GTL. The GTL/CL interface creates instances of PerformedAction when the create and update methods are invoked, whereas SuggestedAction instances are created as a result of the processing of the CL. We have included some Actions in the general CL model ontologies that can be extended in different applications.

Actions, and in particular SuggestedAction instances can be generated at different abstraction levels, but they need to have as a result an operation that can be processed by the GTL (or, at least, that can be transformed by the GTL/CL interface to actual GTL methods). SuggestedActions are placed in the recommendations queue of the framework, as explained in Section 4. Actions have also associated a time stamp to support the development of selection techniques based on their creation time.

6. Reasoning: scene recognition and feedback to the tracker

As detailed in the previous section, the CL ontologies offer a formal vocabulary to symbolically describe the dynamics and the surroundings of the observed scene. The instances of the concepts and the relations of the CL ontology represent the evolution in time of the scene tracks, objects, situations, and impacts. Thus, scene interpretation is the process of creating instances of the CL ontologies accordingly and consistently to the perceived data. The instances of the CL model are created in successive reasoning steps,

corresponding to the correspondence, recognition, and evaluation procedures depicted in Fig. 3.

Standard ontology reasoning procedures can be performed within the CL ontologies to infer additional knowledge from the explicitly asserted facts (tracking data and a priori context). By using a DL inference engine, tasks such as classification or instance checking can be performed. It is as well possible to perform other extended inferences based on them, or to add SWRL rules. Nevertheless, as mentioned in Section 3.2, monotonicity of DLs forbids adding new knowledge to ontologies while reasoning. Rules, and specifically SWRL rules, also have this restriction as a consequence of the DL safety condition. This is a serious drawback to the use of ontologies in scene interpretation. As it has been pointed out by several authors, while standard DLs are valuable for scene representation, scene interpretation requires more expressive retrieval languages for querying the models (Neumann & Möller, 2008). Two reasons have been argued: scene interpretation cannot be modeled as classification, because it is more a model constructing task; and direct instance checking is not possible, since individuals do not exist for undiscovered objects.

In essence, the problem is that scene interpretation is a paradigmatic case of abductive reasoning, in contrast to the DL deductive reasoning. Abductive reasoning takes a set of facts as input and finds a suitable hypothesis that explains them – sometimes with an associated degree of confidence or probability. This is what is needed in our case: we want to figure out what is happening in the scene from the observed and the contextual facts. In terms of the architecture of the CL, scene interpretation can be seen as an abductive transformation from knowledge expressed in a lower level ontology to knowledge expressed in a higher level ontology. In contrast to monotonic DL reasoning, which can be considered an intra-ontology procedure, this process can be regarded as inter-ontology reasoning. Abductive reasoning is out of the scope of classical DLs (Elsenbroich, Kutz, & Sattler, 2006), and is not supported by ontologies. Fortunately, it can be simulated by using customized procedures or, preferably, by defining transformation rules in a suitable query language.

Interestingly enough, RACER reasoner allows abductive reasoning. In our framework, we use RACER features beyond the usual DL support to accomplish scene interpretation. RACER transparently supports standard (deductive) and non-standard (abductive) ontology-based reasoning with nRQL (new RACER Query Language) (Wessel & Möller, 2005). nRQL is a modification and query language that allows instance querying and rule definition. In this section, we explain some nRQL features used in the implementation of the framework. We describe various examples of nRQL rules based on the terms of the CL model that show how heuristic and common-sense knowledge (i.e. context knowledge) can be formally represented and applied to accomplish high-level scene interpretation and low-level tracking refinement.

6.1. Model querying

The basic queries in nRQL are instance queries. nRQL instance queries, expressed in a Lisp-like syntax, consists of a head and a body. The body is a set of conditions defined with concept and role expressions involving variables (noted with '?'). Variables are bound to the named instances of the ontology (both asserted and inferred). The head is a list that includes various of the variables in the body. The result of the query is the set of bindings of the variables in the head that satisfy the logical condition of the body.

In our framework, instance queries are used to get information from the model. For example, the next query finds all the instances of SceneObject that are being observed in the current scene. For the sake of simplicity, we will abbreviate the complete namespaces of the CL ontologies from this point on.

```
;;; Simple nRQL query
(retrieve
 (?x)
  (and
   (?x   scob:SceneObject)
   (?x ?s scob:hasObjectSnapshot)
   (?s   scob:SceneObjectSnapshot)
   (?s   tren:unknown_frame tren:isValidInEnd)
  )
)
```

More precisely, the query retrieves the SceneObject instances that have associated a snapshot (with property hasObjectSnapshot) which is currently valid (i.e. it has an unknown end of validity frame – property isValidInEnd). Matching between ontology instances and variables in this query is shown in Fig. 5. We assume that mirror1 has been asserted to be an instance of the Mirror concept, and o is a SceneObject. a and b are SceneObjectSnapshot instances respectively related to o and mirror1 with hasObjectSnapshot property. Instances a and b are respectively related to f and unknown_frame with isValidEnd property. Since a is not related to unknown_frame, it does not match the query pattern, and the result of the consult is the binding (?x, mirror1).

It is also possible to create defined queries, which are stored queries that can be nested inside another `retrieve` commands. The following command defines a query, named `current-position-of-object`, that retrieves the current position of an object (instance of OPosition concept). Defined queries can be used in subsequent consults. Analogous stored queries have been created for convenience to perform other common consults (current position of a track, current area, etc.).

```
;;; Defined query
(defquery current-position-of-object (?o ?op)
 (and
  (?o                    scob:SceneObject)
  (?o   ?osn             scob:hasObjectSnapshot)
  (?osn                  tren:unknown_frame
                         tren:isValidInEnd)
  (?osn ?opr             scob:hasObjectProperties)
  (?opr ?op              scob:OhasPosition)
  (?op                   scob:OPosition)
 )
)
```

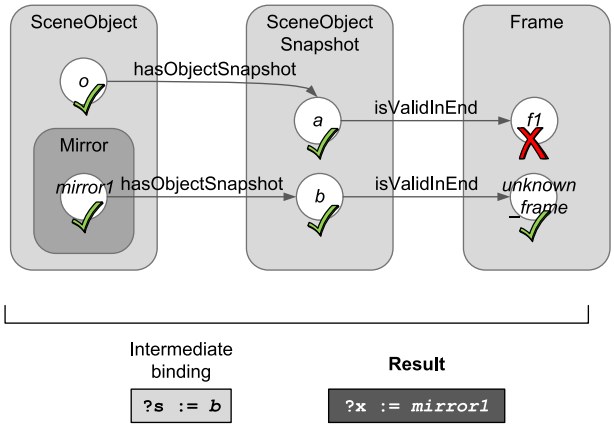


Fig. 5. Matching between variables and instances in a model query.

6.2. Arithmetical calculus

More complex queries have been created to compute arithmetical functions on the values of the CL instances. nRQL admits the use of lambda calculus expressions to denote computations on the retrieved values, both in the head and the conditions of the queries. The query below calculates the Euclidean distance between any pair of 2D points of the scene. The query can be easily adapted to obtain other object and track numerical properties:

```
;;; Query with arithmetical operations
(retrieve
 (?pl ?p2
  (
   (lambda
    (xl x2 yl y2)
    (expt
     (+
      (expt
       (- (first xl) (first x2)) 2)
      (expt
       (- (first yl) (first y2)) 2)
  )
  1/2)
 )
  (datatype-fillers (tren:x ?pl))
  (datatype-fillers (tren:x ?p2))
  (datatype-fillers (tren:y ?pl))
  (datatype-fillers (tren:y ?p2))
  ))
  (and
   (?pl  tren:2DPoint)
   (?p2  tren:2DPoint)
  )
)
```

6.3. Spatial and temporal reasoning

Spatial reasoning is essential in Computer Vision systems. Topological reasoning has been implemented in our framework by relying on the native support of RACER for RCC-based predicates. RCC (Region Connection Calculus) is a logic-based formalism to symbolically represent and reason with topological relations between objects (Randell, Cui, & Cohn, 1992). RCC semantics cannot be directly represented with OWL (Katz & Cuenca Grau, 2005; Grüntter, Scharrenbach, & Bauer-Messmer, 2008), but RACER offers ad hoc support for them. Specifically, RACER supports the RCC-8 subset through the definition of an additional layer (substratum) and the use of special query symbols.

The use of RCC predicates in our framework is shown in the following query. First, the RCC substrate is activated. Next, the property insideOf, defined in the SCOB ontology, is stated as equivalent to the RCC predicates `ntpp` (inclusion of the first object in the second one without connection) and `tpp` (inclusion of the first object in the second one with tangential connection). Finally, all the instances of Person located inside a Building are retrieved. With the RCC substrate activated, if a person p is asserted to be inside a room r, and the room is inside of a building b, the query retrieves p, according to the semantics of `ntpp` and `tpp`.

```
;;; RCC-based topological query
(enable-rcc-substrate-mirroring)
(rcc-synonym scob:insideOf (:ntpp:tpp))
(retrieve
 (?*x ?*y)
 (and
  (?*x ?*y scob:insideOf)
  (?*x    surv:Person)
  (?*y    surv:Building)
 )
)
```

RCC semantics can also be used with temporal properties, in such a way that a time interval can be asserted to be included in another one, consecutive, partially coincident, etc.

6.4. Deductive rules

More interestingly, the framework uses RACER capability for reasoning with rules. nRQL rules have a structure very similar to instance queries. The antecedent of the rule is a pattern equivalent to the body of a query. The consequent includes logical conditions that must be satisfied by the ontology, or expressions to create new instances. Deductive rules only include conditions of the first type, which means that they do not contain in the consequent any individual not mentioned in the antecedent.

Deductive rules are used to maintain the consistency of the ontology and to explicitly assert axioms affecting existing instances. For example, the following command prepares a deductive rule that states that the position value of a TrackedObject must be equal to the position value of its associated Track. If both positions have been introduced and the (x, y) values are not equal, the ontology is reported as inconsistent. If the (x, y) values of the position of the Track have been introduced, while the (x, y) values of the TrackObject have not been, then these values of the second position are made equal to the values of the first one.

```
;;; Rule 1. Deductive rule
(prepare-rule
 (and
  (?o       scob:TrackedObject)
  (?t       tren:Track)
  (?o  ?t   scob:hasAssociatedTrack)
  (?t  ?tp  current-position-of-track)
  (?o  ?op  current-position-of-object)
  (?tp  ?tpt  tren:TpositionValue)
  (?op ?opt  scob:OpositionValue))
 (
  (instance ?opt
   (some tren:x
    (= racer-internal%has-real-value
     ( (lambda (x) (float (first x)) )
     (datatype-fillers (tren:x ?tpt))))))
  (instance ?opt
   (some tren:y
    (= racer-internal%has-real-value
     ( (lambda (y) (float (first y)) )
      (datatype-fillers (tren:y ?tpt))))))
 )
)
```

6.5. Abductive rules

Abductive rules include new individuals in the consequent, which are created as new instances of the ontology. As explained, abductive rules are used in the framework to achieve its two main objectives: scene interpretation and creation of feedback for the GTL.

6.5.1. Scene interpretation

Abductive rules are defined in the framework to interpret what is happening in the scene from the basic tracking data. In that

manner, symbolic scene descriptions are built from visual measures through various inference steps (correspondence, recognition, evaluation) that calculate instances of an upper ontology with rules operating on instances of a lower ontology.

An example of an abductive rule to calculate correspondences between tracks and objects is presented below. This rule creates a new Person instance when a track bigger than a predefined size (25×40) (and not associated to any object) is detected in the image. The created Person instance is associated with the identified track – it would be as well necessary to initialize the properties of the new object, even if they are not assigned concrete numerical values, but this is omitted in the example.

```
;;; Rule 2. Correspondence abduction rule
(prepare-rule
 (and
  (?t     scob:unknown_object
          scob:isAssociatedToObject)
  (?t ?ts  current-size-of-track)
  (?ts ?d   tren:TsizeValue)
  (?d     tren:2DDimension)
  (?d     (>= tren:w 25.0))
  (?d     (>= tren:h 40.0)))
(
  (instance
   (new-ind person-ins ?t) surv:Person)
  (forget-role-assertion
   ?t
   scob:unknown_object
   scob:isAssociatedToObject)
  (related
   (new-ind person-ins ?t)
   ?t
  scob:hasAssociatedTrack)
 )
)
```

Abduction rules for scene recognition can involve knowledge at different levels. Another example of abductive rule is the following one, which recognizes the situation when a person is reflected by a mirror. This rule is fired when a person is inside of the area of influence of a mirror and, at the same time, there is another detected person inside of the corresponding mirror. For the sake of clarity, in this example rule we suppose that the value of the insideOf property has been previously calculated. As a result, the rule creates a new instance of the ReflectionSituation concept with suitable property values (the person and the mirror involved in the activity). In this case, it would be also convenient to assign to the new ReflectionSituation instance a proper SituationSnapshot instance – this part of the rule consequence is omitted.

```
;;; Rule 3. Recognition abduction rule
(prepare-rule
 (and
  (?real_person              surv:Person)
  (?real_person   ?olp       current-position-of-
                             object)
  (?mirror_person            surv:Person)
  (?mirror_person ?o2p       current-position-of-
                             object)
  (?mirror                   surv:Mirror)
  (?mirror      ?ma          current-area-of-
                             object)
  (?mirrored_area            surv:MirroredArea)
  (?mirror ?mirrored_area    surv:mirrorsArea)
  (?mirrored_area ?maa       current-area-of-
                             object)
  (?olp         ?ma          scob:insideOf)
  (?o2p         ?maa         scob:insideOf))
 (
  (instance
   (new-ind ref-ins ?mirror_person)
   surv:ReflectionSituation)
  (related
   (new-ind ref-ins ?mirror_ person)
   ?mirror
   surv:mirror)
  (related
   (new-ind ref-ins ?mirror_person)
   ?mirror_person
   surv:personInMirror)
  (related
   (new-ind ref-ins ?mirror_ person)
   ?real_person
   surv:mirroredPerson)
 )
)
```

6.5.2. Feedback

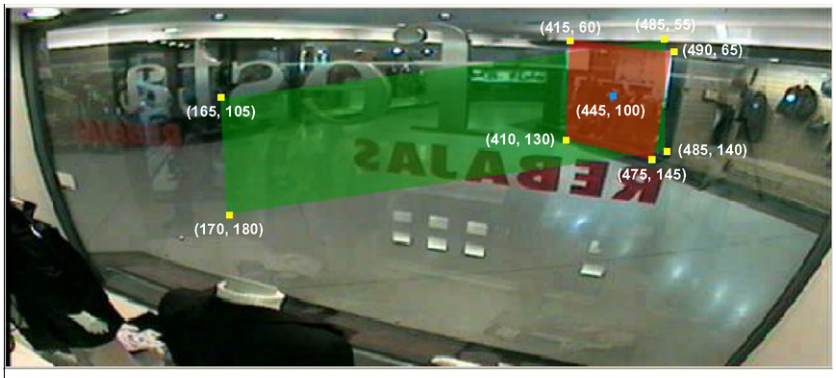
In essence, process refinement is very similar to scene interpretation, since in both cases abductive reasoning is carried out to draw new knowledge – in this case, instances of the RECO ontology. From the current interpretation of the scene, the historical data, and the predictions, feedback abductive rules are triggered to create instances of the SuggestedAction concept. As explained in Section 4, these actions are retrieved by the GTL through the GTL/CL interface and interpreted accordingly, resulting (if not discarded) in corrections of the tracking parameters, modifications of the data structures managed by the algorithm, etc.

Recommendations can be generated at different abstraction levels, i.e. they can involve Tracks, SceneObjects, Situations, Impacts, etc. or even combinations of them. In practice, that means that feedback rules have terms of the TREN, SCOB, ACTV, and IMPC ontologies in their antecedent; and terms of the RECO ontology in their consequent. The following rule obtains a recommendation that suggests ignoring the track associated to a reflection in a mirror.

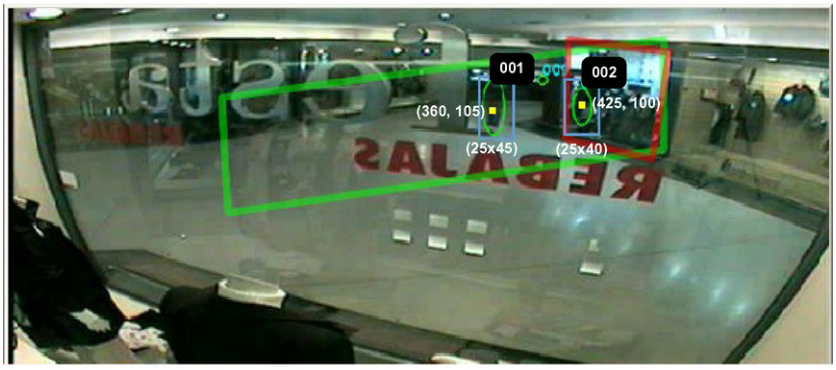
```
;;; Rule 4. Feedback abduction rule
(prepare-rule
 (and
  (?s                surv:ReflectionSituation)
  (?s ?personInMirror surv:personInMirror)
  (?personInMirror ?t scob:hasAssociatedTrack))
 (
  (instance
   (new-ind ign-instance ?s)
   surv:IgnoreTrack)
  (related
   (new-ind ign-instance ?s)
   ?t
   surv:trackToBeIgnored)
 )
)
```

7. Example: surveillance application

The queries and rules of the previous section give insights on the implementation of reasoning processes within our framework. In this section, we depict the functioning of the framework with an example of a video-surveillance application. Firstly, we



(a) Frame 1. Initial state of the scenario



(b) Frame 2. A person appears in the scenario and is reflected in the mirror

Fig. 6. Frames 1 and 2 of the video sequence.

describe how the CL model is created and contextual information and rules are introduced. Secondly, we depict the reasoning within the CL.

For the sake of simplicity, we use two frames of a video sequence of the PETS2002 benchmark.[3] In this recording, several people walk in front of a shop window. Fig. 6(a) shows the initial state of the scenario. We have marked a mirror and its associated mirrored area on the frame. Fig. 6(b), which is assumed to be the next available frame, shows two new tracks detected by the GTL.

7.1. Initialization of the CL model

We have developed the specific SURV ontology (introduced in Section 5) that specializes the generic CL model to this scenario. The SURV ontology imports the general CL ontologies (TREN, SCOB, ACTV, IMPC, RECO), and additionally includes new:

- Concepts:
  Person, Mirror, MirroredArea, ReflectionSituation, Building, Room, etc.
- Relations:
  isMirroredBy, mirrorsArea, etc.
- Axioms:
  Mirror $\sqsubseteq$ StaticObject,
  MirroredArea $\sqsubseteq$ StaticObject,
  ReflectionSituation $\sqsubseteq$ ObjectInteraction $\sqsubseteq$ Situation,
  Mirror $\sqsubseteq \exists$mirrorsArea.MirroredArea, etc.

Before the GTL is started, it is necessary to *annotate* the scenario. Annotating the scenario means to create instances of the SURV ontology describing the static objects. Thus, in this example we first create new instances describing the scene of Fig. 6(a): the presence of a mirror (mirror1), its location (mirror1_position), and the extension of its area of influence (mirroredArea1). Fig. 7 provides an excerpt of the corresponding ontology instances. The graphical representation highlights in red[4] italics the new instances, and in red dashed lines the new relations between instances. Besides, the corresponding OWL code (in RDF/XML syntax) is presented. These instances have been created with the Protégé[5] ontology editor.

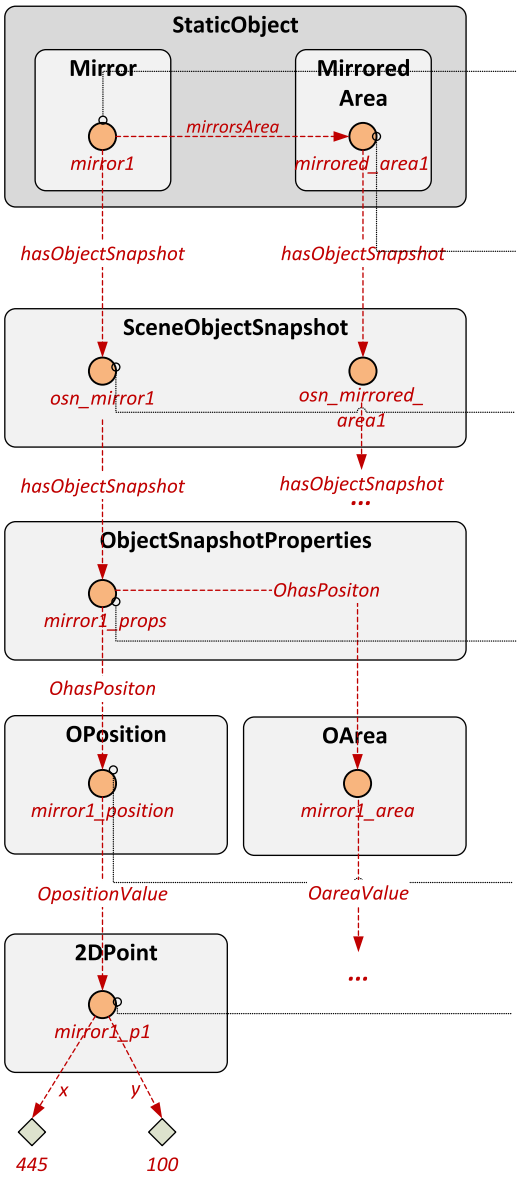
After initialization, the SURV ontology is loaded into RACER. Contextual rules for scene interpretation and feedback reasoning are also loaded. In this example, we use the four rules defined in Sections 6.4 and 6.5. These rules are naive examples to illustrate:

- Rule 1. Deduction of values: *the position values of a tracked entity and the associated track must be equal*
- Rule 2. Object association: *if a track is bigger than (25 $\times$ 40), then it corresponds to a person*
- Rule 3. Activity recognition: *if there is a person inside a mirror and a person inside its associated mirrored area, then a reflection is happening*
- Rule 4. Feedback provision: *if a reflection is happening, then the reflected track should be ignored*

[3] The Performance Evaluation of Tracking and Surveillance (PETS) dataset has available numerous scenarios. We have selected a 1-min long sequence from the PETS2002 dataset, which was intended to track pedestrians walking inside a shopping mall (http://www.cvg.cs.rdg.ac.uk/PETS2002/pets2002-db.html).

[4] For interpretation of the references to color in this text, the reader is referred to the web version of this article.

[5] http://protege.stanford.ed.




```
[...]

<!-- mirror1 instance -->
<owl:Thing rdf:about="#mirror1">
  <rdf:type rdf:resource="#Mirror"/>
  <mirrorsArea rdf:resource="#mirrored_area1"/>
  <scob:hasObjectSnapshot rdf:resource="#osn_mirror1"/>
</owl:Thing>

<!-- area mirrored by mirror1 -->
<owl:Thing rdf:about="#mirrored_area1">
  <rdf:type rdf:resource="#MirroredArea"/>
  <scob:hasObjectSnapshot rdf:resource="#osn_mirror1"/>
</owl:Thing>

<!-- object snapshot of mirror1 -->
<owl:Thing rdf:about="#osn_mirror1">
  <rdf:type rdf:resource="&scob;SceneObjectSnapshot"/>
  <scob:hasObjectProperties rdf:resource="#mirror1_props"/>
  <tren:isValidInEnd rdf:resource="&tren;unknown_frame"/>
</owl:Thing>

<!-- properties of mirror1 snapshot (area and pos.) -->
<owl:Thing rdf:about="#mirror1_props">
  <rdf:type rdf:resource="&scob;ObjectSnapshotProperties"/>
  <scob:OhasArea rdf:resource="#mirror1_area"/>
  <scob:OhasPosition rdf:resource="#mirror1_position"/>
</owl:Thing>

<!-- mirror1 position -->
<owl:Thing rdf:about="#mirror1_position">
  <rdf:type rdf:resource="&scob;OPosition"/>
  <scob:OpositionValue rdf:resource="#p5"/>
</owl:Thing>

<!-- mirror1 position value -->
<owl:Thing rdf:about="#p5">
  <rdf:type rdf:resource="&tren;2DPoint"/>
  <tren:y rdf:datatype="&xsd;float">100</tren:y>
  <tren:x rdf:datatype="&xsd;float">445</tren:x>
</owl:Thing>

[...]
```

Fig. 7. Initial scene annotation (graphical and OWL XML-based notation).

7.2. Processing in the framework

The GTL begins by processing frame 1. Since no moving objects are detected, no track data updates are communicated to the CL.

After that, the GTL processes frame 2. In this case, two new tracks are detected: *track 1* and *track 2*, with the properties depicted in Fig. 6(b). The GTL invokes the GTL/CL interface, and notifies the CL that two new tracks have been created. This call triggers the reasoning procedures in the CL layer:

1. *Abstraction*. Two Track instances (track1, track2) are created and added by the GTL/CL interface to the scene model managed by the RACER reasoner. These instances are assigned property values according to the values calculated by the GTL. For instance, size of track1 is (25 × 45) and position is (360, 105). Fig. 8 graphically shows some of the new instances and connections between instances added to model as a result of the creation of track1 (new instances created in this step are marked in red italics; new relations are in red dashed lines).
2. *Correspondence*. The new track1 (and related instances) match `Rule 2` with bindings: `?t` := track1; `?ts` := track1_size; `?d` := d1. Accordingly, a new instance of Person, named person1, is created and associated to track1 (with hasAssociatedTrack). Analogously, track2 matches `Rule 2`, and consequently a new instance of Person, named person2, is created and associated to track2. Fig. 9 shows an excerpt of some of these new instances, focusing on person1. It can be seen that the position values of person1 have not been yet assigned.
3. *Additional correspondence*. The instances track1 and person1 match `Rule 1` with bindings: `?o` := person1; `?t` := track1; `?tp` := track1_position; `?op` := person1_position; `?tpt` := mirror1_p1; `?opt` := person1_p1.
   Consequently, the (*x*, *y*) values of point person1_p1 are made equal to (380, 105). Analogously, track2 and person2 match `Rule 1`, and therefore the (*x*, *y*) values of point person2_p1 are made equal to (425, 100). Fig. 10 shows an excerpt of the ontology after the assignment of (*x*, *y*) position values to person1.

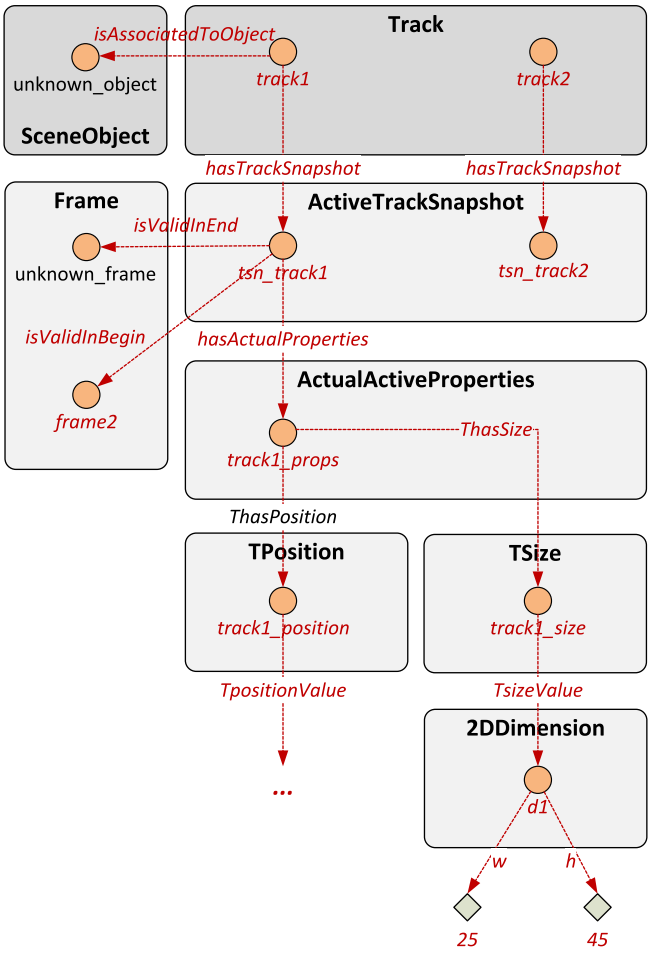


Fig. 8. Ontology instances added after the creation of track1.

4. *Recognition*. With the previous assignments, the instances of the ontology match `Rule 3` with the following variable bindings: `?real_person` := person1; `?olp` := person1_position; `?mirror_person` := person2; `?o2p` := person2_position; `?mirror` := mirror1; `?ma` := mirror1_area; `?mirrored_area` := mirrored_area1; `?maa` := mirrored_area1_area.
   As a result of this rule, a new instance of ReflectionSituation, named reflection1, is created. mirror1, person1, and person2 are associated to reflection1 through the properties mirror, mirroredPerson, and personInMirror, respectively. These new instances are shown in Fig. 11.
5. *Feedback*. The new Reflection instance make possible matching of `Rule 4` with the following variable bindings: `?s` := reflection1; `?personInMirror` := person2; `?t` := track2.
   As a result of this rule, a new instance of IgnoreTrack, named ignore_action1, is created, with track2 as the track to be ignored by the GTL (Fig. 12).

After firing `Rule 4`, reasoning stops, because no more rule antecedents are matched. The feedback action obtained after the reasoning process – the *ignore track* recommendation – is placed in the recommendations queue. In synchronous mode, the queue will be consulted by the GTL before processing the next frame of the video sequence. The recommendation will be interpreted by the GTL, and suitable actions will be carried out.

8. Conclusions and future work

In this paper, we have presented a context-based framework to achieve high-level interpretation of video data and to improve object tracking in complex scenarios. The framework extends a classical tracking system (the GTL) with an ontological layer (the CL) that represents and reasons with sensorial data and context knowledge, in order to avoid the issues that makes classical object tracking procedures fail in complex scenarios. Based on contextual and heuristic knowledge formally represented with ontologies and

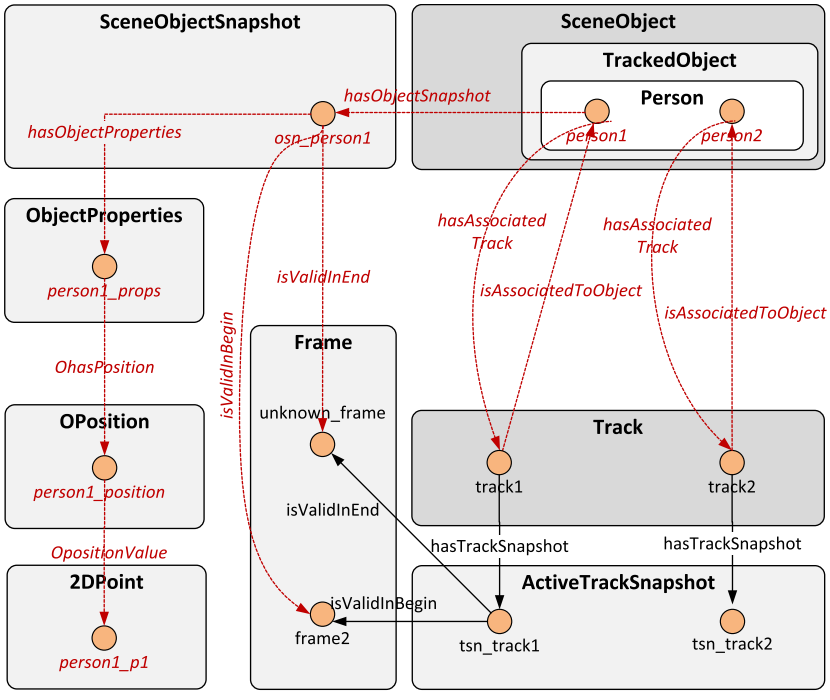


Fig. 9. Ontology instances added after the creation of the correspondence between person1 and track1.

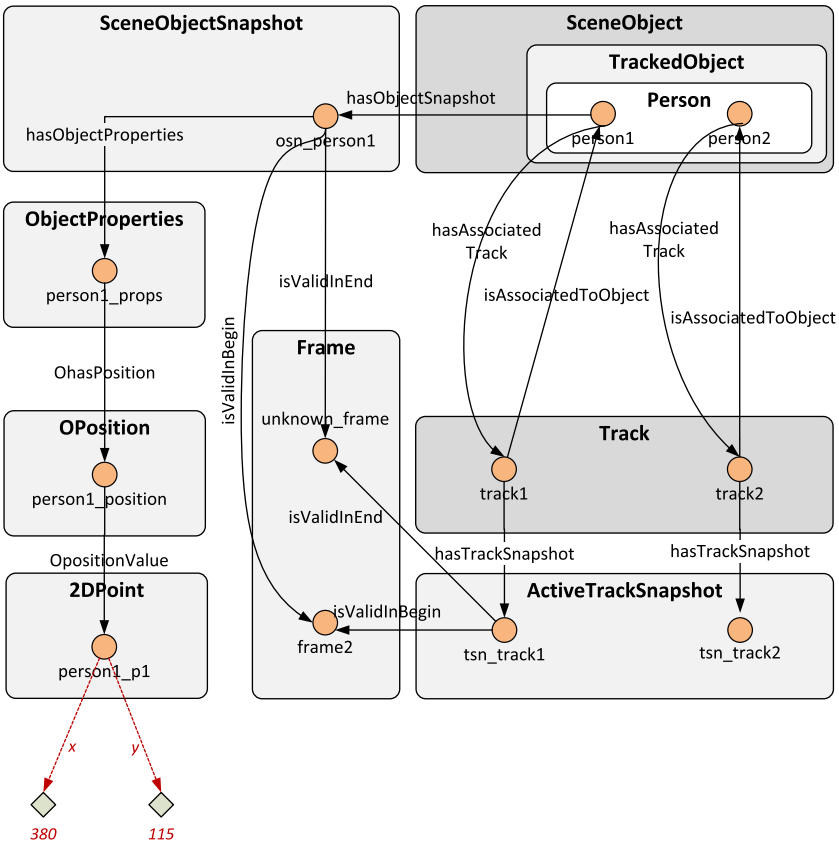


Fig. 10. Ontology instances added after deduction of position values to person1.

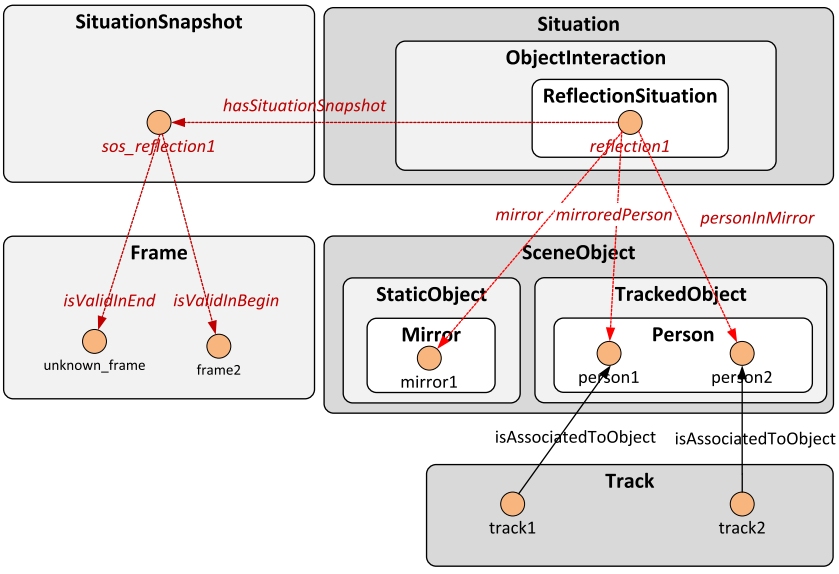


Fig. 11. Ontology instances created after recognition of situation reflection1.

rules, the CL applies abductive and deductive reasoning to build a symbolic model of the scene structured in various abstraction levels – from low-level tracking data to high-level activity descriptions –, according to the JDL Information Fusion model. The interpretation of the scenes in the CL are used to generate feedback to the GTL and enhance the quantitative tracking procedures.

Formal representation of knowledge with ontologies in our framework has several advantages. General reasoning methods (i.e. DL-deduction and rules) and existing inference engines (i.e. RACER) can be used, reducing the effort needed to implement scene interpretation procedures. We have proposed a layered set of general ontologies that can be extended in different scenarios

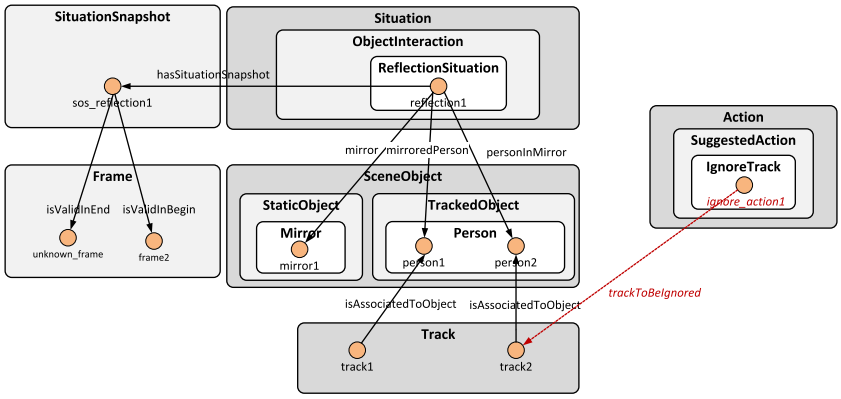


Fig. 12. Ontology instances created after creation of recommended action ignore_action1.

and example rules, facilitating the development of domain-specific models and rule bases. Consequently, the framework can be easily adapted to different domains. The structure of the knowledge model is compliant to the well-defined JDL conceptual framework, which provides a sound and formal substrate for the framework architecture. Additionally, symbolic representations of scenes are also more interpretable, which facilitates participation of human users in the system, debugging and adjusting the algorithms, and interoperation with other components and systems. Interestingly enough, our representation allows the description of the temporal evolution of the system, and not only its state in a precise instant, and spatial reasoning. Particularly, we have shown a brief example on the use of the framework in a surveillance application.

We plan to continue this research work various directions. First, we will fully integrate the CL with the tracking software. We will test our proposal with existing datasets to demonstrate – beyond the presented example – that the contextual layer effectively interprets the perceived scene and reduces tracking errors, and we will also measure the improvement with respect to other methods. This may imply further refinements or simplifications of the current model, which has been developed with a very broad scope. Likewise, suitable descriptive ontologies extending the model and abduction rules will have to be created (manually or semi-automatically), which poses a notable challenge because it may demand a considerable effort. Machine learning methods could be considered in this case. Real-time applications will require further studies on the performance of the framework, which has been briefly considered. The eventual objective of the future research works is to incorporate uncertain information and reasoning in the framework, which is inherent to most fusion and vision applications.

Acknowledgements

This work was supported in part by Projects CICYT TIN2008-06742-C02-02/TSI, CICYT TEC2008-06732-C02-02/TEC, SINPROB, CAM MADRINET S-0505/TIC/0255 and DPS2008–07029-C02–02.

References

Arndt, R., Troncy, R., Staab, S., Hardman, L., & Vacura, M. (2008). COMM: Designing a well-founded multimedia ontology for the web. In *Proceedings of the sixth international semantic web conference (ISWC 2007)* (pp. 30–43). Busan, South Korea.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., & Patel-Schneider, P. F. (2003). *The description logic handbook: Theory, implementation, and applications*. Cambridge University Press.

Baader, F., Horrocks, I., & Sattler, U. (2005). Description logics as ontology languages for the semantic web. *Mechanizing Mathematical Reasoning*, 228–248.

Baader, F., Horrocks, I., & Sattler, U. (2008). Handbook of knowledge representation. In *Description logics* (pp. 135–180). Elsevier.

Bossu, G., & Siegel, P. (1985). Saturation, nonmonotonic reasoning and the closed-world assumption. *Artificial Intelligence, 25*, 13–63.

Boult, T. E., Micheals, R. J., Gao, X., & Eckmann, M. (2001). Into the woods: Visual surveillance of non-cooperative and camouflaged targets in complex outdoor settings. *Proceedings of the IEEE, 89*(10), 1382–1402.

Brdiczka, O., Yuen, P. C., Zaidenberg, S., Reignier, P., & Crowley, J. L. (2006). Automatic acquisition of context models and its application to video surveillance. In *Proceedings of the 18th international conference on pattern recognition (ICPR 2006)* (pp. 1175–1178). Hong Kong, China.

Bremond, F., & Thonnat, M. (1996). A context representation for surveillance systems. In *Proceedings of the workshop on conceptual descriptions from images at the fourth european conference on computer vision (ECCV'96)*. Cambridge, UK.

Das, S. (2008). *High-level data fusion*. Artech House Publishers.

Dey, A., & Abowd, G. (2000). Towards a better understanding of context and context-awareness. In *Proceedings of the workshop on the what, who, where, when, and how of context-awareness (CHI 2000)*. The Hague, Netherlands.

Elsenbroich, C., Kutz, O., & Sattler, U. (2006). A case for abductive reasoning over ontologies. In *Proceedings of the OWL workshop: Experiences and directions (OWLED'06)*. Athens, Georgia, USA.

Fernández, C., & González, J. (2007). Ontology for semantic integration in a cognitive surveillance system. In *Proceedings of the second international conference on semantic and digital media technologies* (pp. 260–263). Genoa, Italy.

François, A. R., Nevatia, R., Hobbs, J., Bolles, R. C., & Smith, J. R. (2005). VERL: An ontology framework for representing and annotating video events. *IEEE Multimedia, 12*(4), 76–86.

Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., & Schneider, L. (2002). Sweetening ontologies with DOLCE. In *13th international conference on knowledge engineering and knowledge management (EKAW02)* (pp. 223–233). Sigüenza, Spain.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition, 5*(2), 199–220.

Grüntter, R., Scharrenbach, T., & Bauer-Messmer, B. (2008). Improving an RCC-derived geospatial approximation by OWL axioms. In *Proceedings of the seventh international semantic web conference (ISWC 2008)* (pp. 293–306). Karlsruhe, Germany.

Häarslev, V., & Möller, R. (2001). Description of the RACER system and its applications. In *Proceedings of the international workshop on description logics (DL2001)*. California, USA: Stanford University.

Hall, D. L., & Llinas, J. (2009). Multisensor data fusion. In *Handbook of multisensor data fusion* (pp. 1–14). CRC Press.

Haritaoglu, I., Harwood, D., & David, L. S. (2000). W4: Real-time surveillance of people and their activities. *IEEE Transactions Pattern Analysis and Machine Intelligence, 22*(8), 809–830.

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., & Rudolph, S. (2008). OWL 2 Web Ontology Language primer. Online, W3C recommendation. Available from http://www.w3.org/TR/owl2-primer/.

Hobbs, J., & Pan, F. (2006). Time ontology in OWL. Online, W3C working draft. Available from http://www.w3.org/TR/owl-time/.

Horrocks, I., & Patel-Schneider, P. (2004). Reducing OWL entailment to description logic satisfiability. *Web Semantics: Science, Services and Agents on the World Wide Web, 1*(4), 345–357.

Horrocks, I., & Patel-Schneider, P. F. (2004). A proposal for an OWL rules language. In *Proceedings of the 13th international conference on World Wide Web (WWW 2004)* (pp. 723–731). New York, NY, USA.

Huang, Y., & Huang, T. (2002). Model-based human body tracking. In *Proceedings of the 16th international conference on pattern recognition (ICPR 2002)* (Vol. 1, pp. 552–555).

Katz, Y., & Cuenca Grau, B. (2005). Representing qualitative spatial information in OWL-DL. In *Proceedings of OWL: Experiences and directions workshop (OWLED 2005)*. Galway, Ireland.

Kokar, M., & Wang, J. (2002). Using ontologies for recognition: An example. In *Fifth international conference on information fusion* (Vol. 2, pp. 1324–1330). Annapolis, MD, USA.

Kokar, M. M., Matheus, C. J., & Baclawski, K. (2009). Ontology-based situation awareness. *Information Fusion, 10*(1), 83–98.

Lambert, D. (2003). Grand challenges of information fusion. In *Proceedings of the sixth international conference of information fusion* (Vol. 1, pp. 213–220). Cairns, Australia.

Lee, W., Bürger, T., & Sasaki, F. (2009). Use cases and requirements for ontology and API for media object 1.0. Online, W3C working draft. Available from http://www.w3.org/TR/media-annot-reqs/.

Little, E. G., & Rogova, G. L. (2009). Designing ontologies for higher level fusion. *Information Fusion, 10*(1), 70–82.

Llinas, J., Bowman, C., Rogova, G., Steinberg, A., Waltz, E., & White, F. (2004). Revisiting the JDL data fusion model II. In *Proceedings of the seventh international conference on information fusion* (pp. 1218–1230). Stockholm, Sweden.

Maillot, N., Thonnat, M., & Boucher, A. (2004). Towards ontology-based cognitive vision. *Machine Vision and Applications, 16*(1), 33–40.

McGuiness, D., & van Harmelen, F. (2004). OWL web ontology language overview. Online, W3C recommendation. Available from http://www.w3.org/TR/owl-features/.

Motik, B., Sattler, U., & Studer, R. (2005). Query answering for OWL-DL with rules. *Web Semantics: Science, Services and Agents on the World Wide Web, 3*(1), 41–60.

Neumann, B., & Möller, R. (2008). On scene interpretation with description logics. *Image and Vision Computing, 26*, 82–101.

Nowak, C. (2003). On ontologies for high-level information fusion. In *Proceedings of the sixth international conference on information fusion* (Vol. 1, pp. 657–664). Cairns, Australia.

Noy, N., & Rector, A., 2006. Defining n-ary relations on the semantic web. Online, W3C semantic web best practices and deployment working group note. Available from http://www.w3.org/TR/swbp-n-aryRelations/.

Orwell, J., Remagnino, P., & Jones, G. (1999). Multi-camera colour tracking. In *Second IEEE workshop on visual surveillance (VS'99)* (pp. 14–21). Fort Collins, Colo, USA.

Patricio, M. A., Castanedo, F., Berlanga, A., Pérez, O., García, J., & Molina, J. M. (2008). Computational intelligence in multimedia processing: Recent advances. In *Computational intelligence in visual sensor networks: Improving video processing systems* (pp. 351–377). Springer.

Pinz, A., Bischof, H., Kropatsch, W., Schweighofer, G., Haxhimusa, Y., Opelt, A., et al. (2008). Representations for cognitive vision. *ELCVIA: Electronic Letters on Computer Vision and Image Analysis, 7*(2), 35–61.

Randell, D. A., Cui, Z., & Cohn, A. G. (1992). A spatial logic based on regions and connection. In *Proceedings of the third international conference on principles of knowledge engineering and reasoning* (pp. 165–176). Cambridge, MA, USA.

Remagnino, P., Baumberg, A., Grove, T., Hogg, D., Tan, T., Worrall, A., & Baker, K. (1997). An integrated traffic and pedestrian model-based vision system. In *Proceedings of the eighth british machine vision conference (BMVC97)* (pp. 380–389). Essex, UK.

Sánchez, A. M., Patricio, M. A., García, J., & Molina, J. M. (2009). A context model and reasoning system to improve object tracking in complex scenarios. *Expert Systems with Applications, 36*(8), 10995–11005.

Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., & Katz, Y. (2007). Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web, 5*(2), 51–53.

Snidaro, L., Belluz, M., & Foresti, G. L. (2007). Domain knowledge for surveillance applications. In *Proceedings of the 10th international conference on information fusion* (pp. 1–6). Quebec, Canada.

Steinberg, A. N., & Bowman, C. L. (2004). Rethinking the JDL data fusion levels. In *Proceedings of the MSS national symposium on sensor and data fusion*. Columbia, SC, USA.

Steinberg, A. N., & Bowman, C. L. (2009). Revisions to the JDL data fusion model. In *Handbook of multisensor data fusion* (pp. 45–67). CRC Press.

Steinberg, A. N., & Rogova, G. (2008). Situation and context in data fusion and natural language understanding. In *Proceedings of the 11th international conference on information fusion* (pp. 1–8). Cologne, Germany.

Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data Knowledge Engineering, 25*, 161–197.

Vernon, D. (2008). Cognitive vision: The case for embodied perception. *Image and Vision Computing, 26*(1), 127–140.

Wessel, M., & Möller, R. (2005). A high performance semantic web query answering engine. In *Proceedings of the international workshop on description logics (DL2005)*. Edinburgh, Scotland.

Westermann, U., & Jain, R. (2007). Toward a common event model for multimedia applications. *IEEE Multimedia, 14*(1), 19–29.

Yang, M., Wu, Y., & Hua, G. (2009). Context-aware visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 31*(7), 1195–1209.

Yilmaz, A., Javed, O., & Shah, M. (2006). Object tracking: A survey. *ACM Computing Surveys, 38*(4), 1–45.