An Active Memory as a Model for Information Fusion

Sebastian Wrede, Marc Hanheide, Christian Bauckhage, Gerhard Sagerer
Bielefeld University, Faculty of Technology,
P.O. Box 100131, 33501 Bielefeld, Germany
Email: {swrede,mhanheid,cbauckha,sagerer}@techfak.uni-bielefeld.de

Abstract – *Information fusion is a mandatory prerequisite for cognitive vision systems. These are vision systems that apply reasoning and learning on different levels of abstraction and correspondingly have to deal with hypotheses from different categorical domains. Following some principles of human cognition, we present an approach to information fusion that closely couples reasoning and representation. We will discuss how processes like probabilistic contextual reasoning as well as functional and non-functional requirements in storing data from different sources can be integrated by a unified XML based data representation. Due to the interaction between active processes and data storage, we call our approach an active memory. Performance results of an implemented system as well as an evaluation of data fusion from contextual inference will be presented.*

Keywords: Cognitive Vision, Architecture, Frameworks, Bayesian Networks, XML, Information Fusion, Scene Analysis

1 Introduction

During the last decade, the idea of constructing vision systems advanced to the ambition of developing cognitive computer vision systems (CVS) [1, 2]. This term is used to characterize systems which not only involve computer vision algorithms but also employ techniques for machine learning in order to acquire and extend prior knowledge. Furthermore, they apply automatic and contextual reasoning to verify the consistency of results obtained from several computational modules as well as to manage the coordination of these modules [3]. When incorporating such different modules into one complex system, data fusion on the technical as well as on the functional level is needed.

According to Christensen [1], CVS should be embodied and consequently require at least a hybrid architecture to enable sensory bottom-up as well as top-down processing. As learning and adaption are fundamental aspects of a CVS, it requires some kind of a memory. This guides the development of an active memory, that performs information fusion for different types of knowledge obtained from different sources. Since adaption and attention control induce dynamics into the system, dynamic (re-)configuration of modules is desired.

Practical experience shows that if large scale vision systems are being implemented (in the complex but nowadays somewhat usual case by teams of researchers from different institutes located in different countries), one has to consider not only domain specific requirements but always will face problems of large scale software development. Therefore, non-functional requirements have to be taken into account, too. Common examples encountered in practice are reusability, scalability, or transparency. As a consequence, techniques and approaches from software engineering should be cared for when designing vision systems.

The presented approach to an active memory for information fusion is developed within a collaborative research project. As a part of a system, it serves as a personal assistant in an office scenario where it memorizes visually recognized objects and actions. The scene is observed by a head-mounted camera, being part of a mobile AR-gear [4] that also enables interactive retrieval of memory contents by augmenting the actual view of the scene. The detection of relevant objects like keyboard, cups, etc. is realized using an universal, illumination-insensitive object detection approach introduced by Viola and Jones [5] in conjunction with a tracking algorithm based on hyperplanes [6]. User interaction is done by recognition of pointing gestures [7]. For action recognition a particle filtering technique based on the trajectory of the hand is used to recognize actions as “reading”, “typing” and “drinking” [8].

We present a unique attempt to allow systematic data fusion of different knowledge sources in an *Active Memory Architecture* motivated from cognitive foundations presented in the next sections. With XML as unified data description language and a Bayesian network based content consistency validation, a complete framework for information fusion in cognitive vision systems is proposed.

2 Cognitive Foundations for Active Memory Architectures

While there is a great variety of definitions of cognition even within the disciplines of biology, psychology and computer science, all agree that humans are a prototypical cognitive system with an amazing though effortlessly achieved performance. The active memory concept aims at the construction of systems that are able to perform vision tasks in everyday environments and to communicate such tasks to humans. We will not try to prove if any definition of cognition is fulfilled by this framework, but use

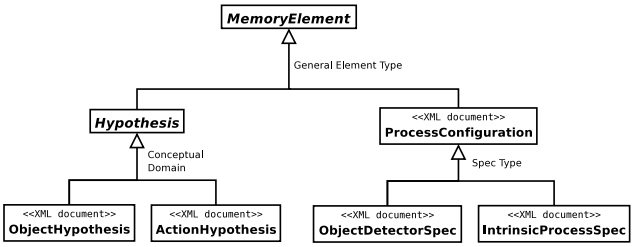


Fig. 1: Exemplary memory elements hierarchy.

cognition as orientation to guide the development of an interactive system. Cruse [9] sees an important incitement of the development of cognition in redundant tasks that involve a high number of degrees of freedom (DoF) which are only partially determined by sensory input. The ability of the system to cope with such redundant situations is often termed as autonomy and is frequently modeled by a separate deliberative or cognitive layer. This redundancy implies the need for information fusion and validation of knowledge. A further key concept is a dynamic representation or manipulable world model that responds to a given sensory input on a longer time scale than lower level modules and can be used to play around with it. This representation has to be unified to aid the fusion of information.

From neuro-physical studies it is assumed that memory is a time-dependent process that can be divided into a short-term and a long-term memory. Based on content, the memory is structured as a successive hierarchy where the content of the higher system is at least partially grounded in lower systems (SPI-model) [10]. Such hierarchies demand for fusion of the underlying information retrieved from the lower systems, but also for an integrative architecture that enables the development of such systems.

3 Active Memory Implementation

Guided by the foundations of cognition, we propose an active memory concept as a *Hierarchically* organized and *Dynamic* information storage with dedicated *Memory Processes*. These perform computations on the actual memory content including reasoning, fusion and learning. They also gather new knowledge from perceptions, or allow interaction with the user. With this active memory concept, *Memory Elements* as atomic information entities are managed by the so called *Memory Infrastructure* and processed by *Intrinsic* (IMP) as well as *Extrinsic Memory Processes* (EMP). Examples for such processes and their contribution to information fusion in the system are discussed in the remainder of this section.

Deriving from the requirements of hierarchical structuring and multimodality of data of the proposed active memory concept, *XML* is used as the underlying basis for data description [11]. In conjunction with advanced XML standards like *XPath*, *XSLT*, *XML Schema* and *XLink* [12] it serves as a standardized fundament for an active memory implementation. This allows to store, retrieve and process information from different abstraction levels and semantic domains and provides a *Unified Data Model*.

```
<OBJECT>
  <HYPOTHESIS>
    <GENERATOR>Object Recognizer BU(N)</GENERATOR>
    <TYPE>OR_RESULT</TYPE>
    <TIMESTAMPS>
      <CREATED value="3452345"/>
      <UPDATED value="3452398"/>
    </TIMESTAMPS>
    <RATING>
      <RELIABILITY value="0.6"/>
    </RATING>
  </HYPOTHESIS>
  <CLASS>Cup</CLASS>
  <REGION image="img_office210703_122">
    <RECTANGLE x="335" y="245" w="65" h="80"/>
  </REGION>
</OBJECT>
```

Fig. 2: XML memory hypothesis example.

3.1 Hypotheses as Memory Elements

In our cognitive vision system, the different modules generate symbolic information from sensing their environment and encode these results, i.e. the memory elements, as XML documents. These include results of for instance *object localizations*, *spatial relations* of objects, and *actions*. As different senses often share the same attributes [13], parts of the knowledge fragments are common for different memory elements and some are specific for the respective type, which leads to a *hierarchy of memory element types*. It follows an object-oriented paradigm validated with inheritance-based XML schema specifications [12]. Figure 1 depicts an exemplary specialization hierarchy of memory elements. This hierarchy allows processes to handle different types of knowledge fragments transparently, since they can only consider the data relevant for the process.

A fundamental type for information fusion is the *Hypothesis*. Since perceptive modules typically do not provide complete accurate results, cognitive systems should not store information as irrevocable facts but as hypotheses with a given *Reliability*. This common data structure describing the uncertainty of a knowledge fragment are called the *Metadata* of the hypothesis. Figure 2 shows an example of a hypothesis where the metadata part is highlighted. As this metadata is available for any kind of hypothesis, processes are developed that only consider this information and can therefore handle any kind of hypothesis, as for instance the “forgetting”-process that is described in section 3.3 in more detail.

3.2 Memory Infrastructure

All memory elements are collected in a persistent storage that relies on a distributed *Repository* style architecture [14]. Modules can access this repository through the so called *Memory Interface*. It serves as a facade to the memory functionality and ensures consistency in terms of transactions and parallel access.

Memory Element Repository: Since we focus on great flexibility, relational databases as persistence solution for XML data are infeasible. Instead, the native Berkeley DB XML database [15] provides the core component of the

memory infrastructure. DB XML is packaged as an embedded C++ library and provides support for transactions and multi-threaded environments. In contrast to classical database servers it offers only core features and is not a complete server application. Its therefore small footprint allows for an efficient implementation of the memory interface. Through the use of XPath [12] developers of memory processes can easily specify queries in a declarative manner. Additionally runtime reindexing is supported for fast access to memory elements.

DB XML also allows for mixed XML and relational environments which is useful for binary data like cropped image patches that are not stored in XML but can still be referenced as active memory content through XLinks [12].

Memory Interface: Based on the DB XML API, a memory interface was implemented that encapsulates complexity and connects the basic insert, select and update methods to the intrinsic and extrinsic memory processes. Through our data centered memory approach an event concept is introduced. Events for notification of EMP event listeners and IMP trigger listeners that are constituted of the called method type and the type of the involved memory element are generated by the memory interface.

Trigger listeners are notified inside the database transaction and their success is checked. On failure, the whole transaction is rolled back and processing is aborted. The execution of registered intrinsic memory processes, which are explained in greater detail in the following section, is controlled by a runtime environment where processes like, e.g., *forgetting* can be realized. Event listeners are notified about a memory interface action once it has been successfully committed to the database and connect the extrinsic memory processes to a concrete instantiation of an active memory, a so called *Memory Instance*. The memory interface is realized in C++ with bindings for MATLAB and Python allowing for rapid prototyping of active memory components. Using the trigger listeners for invocation of intrinsic memory processes and execution in the runtime environment as well as the subscription model for distributed event listeners that attach extrinsic memory processes via a suitable communication framework, the memory instances indeed become *active*.

Distribution of Algorithms: The requirement to connect memory instances and corresponding algorithms across different machines in order to guarantee fast system reaction led to the development of a framework for distributed processing.

The XML enabled Communication Framework (XCF) [16] encapsulates the complexity of building distributed systems and introduces additional features to achieve a low coupling between active memory instances and extrinsic memory processes. XCF itself uses the Internet Communication Engine (ICE) [17] as technical basis and is written in C++. It features a pattern based design as well as communication semantics like publisher-subscriber that allow one-to-many communication, (non-)blocking remote procedure calls (RPC) and event channels. XCF provides

```
<imp name="forgetting" lang="python">
 <init>
  objects = []
  reliabilities = {}
 </init>
 <trigger type="insert" xpath="/HYPOTHESIS">
  <code>
   ts = int(get_xpath('//TIMESTAMPS/UPDATED/@value'))
   rel = float(get_xpath('//RATING/RELIABILITY/@value'))
   reliabilities[vamid] = rel
   objects.append((ts, vamid))
  </code>
 </trigger>
 <trigger type="update"
       xpath="/HYPOTHESIS/RATING/RELIABILITY">
  <code>
   reliabilities[vamid] =
    float(get_xpath('//RATING/RELIABILITY/@value'))
  </code>
 </trigger>
 <trigger type="timer" interval="2">
  <code>
   current_time = time.time()
   for o in objects:
    timestamp, vamid = o
    if timestamp > (current_time - 5000):
      break
    if reliabilities[vamid] < 0.5:
      mi.remove(vamid)
      del reliabilities[vamid]
      objects.remove(o)
  </code>
 </trigger>
</imp>
```

Fig. 3: An IMP specification for a forgetting process.

location, access and migration transparency [18]. Similar to the data storage in the memory interface described previously, data exchange between different modules is based on XML. Since interfaces are specified using XML schema, runtime type safety can be ensured and high performance native datatype transmission is possible. Additionally, (inter-)active system introspection is directly supported that helps in debugging and monitoring a running distributed system for complex cognitive vision tasks.

3.3 IMP: Forgetting

According to Christensen, memory is a limited resource [1], thus *forgetting* is fundamental for our active memory architecture, which can be thought of as some kind of *garbage collector*. It discards hypotheses from the VAM's repository considering their age and reliability, which are available from accessing only the metadata structure. Thus, another memory process can indirectly cause a hypothesis to be removed by lowering the respective reliability. Due to the close coupling of this task to the memory content, the forgetting is realized as an IMP.

Such processes can be declared with so called IMP specifications that consist of an initialization block, one or more trigger listeners and the corresponding algorithms expressed in a scripting language. The type of memory element, as for instance hypothesis, as well as the memory interface action type, e.g. *insert*, *query* or *update* yield a complete trigger specification. An example of an IMP specification for the above mentioned forgetting process is depicted in Figure 3. In this example, three trigger listeners were defined that are registered for the insert and update actions of the memory interface as well as for a special *timer*

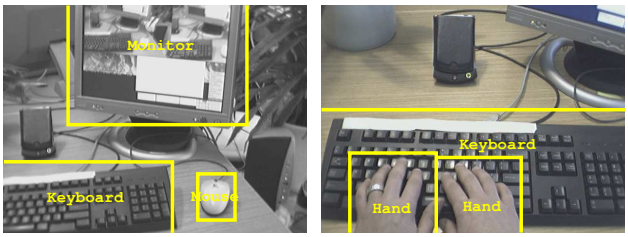


Fig. 4: Two images of a sequence with annotated observations.

event that is generated by the runtime environment in the given interval. The provided XPath specifications match every hypothesis element that is inserted or updated by the memory instance.

The respective algorithms of the IMP listeners are therefore executed when the trigger condition is met. Up to now, the runtime environment executes and compiles Python code which allows for the ad-hoc description of memory processes as shown in this example. Additionally, it performs management tasks, e.g., saving local variables used by a trigger listener and restoring them for listeners of the same IMP upon activation. Objects that allow for repository access within the current transaction are made available and the actual memory element is made available as an XML document that can be manipulated by the IMP listeners. When all registered listeners have finished their computations the XML information is written to the repository backend. Furthermore, the runtime environment is responsible for resource control and scheduling of sequenced listeners from different IMPs.

With these features, a forgetting process can easily be defined in a declarative way and can itself be reconfigured and stored as an element in the repository as shown in Figure 3. Through the execution of IMPs in the active memory itself, processes can efficiently be realized that are highly coupled to vast amounts of the processed data.

3.4 EMP: Consistency Validation

Extrinsic Memory Processes (EMP) are the main computational modules of the cognitive system. In contrast to IMPs these components work asynchronously on the memory instances and are therefore only loosely coupled, which allows for very flexible algorithms and architectures. Note that there is no direct communication between these modules (except communication with low-level modules) but all data exchange is mediated through the memory. The use of XML as exchange data format between an EMP and the Memory Infrastructure makes it straightforward to include existing algorithms and modules in our system since they only have to be extended by the Memory Interface for data input and output. Furthermore, XPath and XLink provide methods to access data stored in the repository.

Consistency Validation: This EMP for consistency validation is chosen as an example to describe the interaction between memory infrastructure and memory processes in more detail. It also outlines the central role of consistency validation for information fusion and how it can be

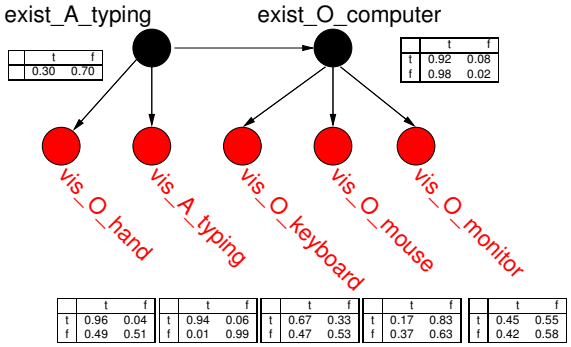


Fig. 5: BN for a computer setup scenario.

achieved. Its aim is to consolidate the memory content by fusing different hypotheses and rejecting inconsistent and unreliable knowledge. Motivated by foundations of cognition [13] the evaluation of *Context* of hypothesis leads to more robust interpretation of sensor data. To be more precise, memory elements are usually not independent of each other, but have *Functional Dependencies*, which allow for determining inconsistencies and to infer advanced knowledge from given sensory data. A common approach to model conditional dependencies is to use *Bayesian networks*. These graphical models proved to be qualified for multi-modal scene understanding [19] and in the field of multi-cue fusion for object recognition [20]. In the following, we describe how Bayesian networks can be applied for consistency validation in a visual active memory architecture.

If perception processes feed their results into the memory, it might occur that some of the assumed hypotheses are conflicting with others. This may be due to flawed results of the perceptions or to a change of situation that made a hypothesis invalid. The task of a consistency validation is to detect these conflicts and solve them by e.g. changing the metadata of the respective hypotheses. The consistency validation can affect the forgetting of conflicting hypotheses by lowering their reliability factor. Consistency validation is thus defined as an EMP that uses *Functional Dependency Concepts* (FDC) to rate hypotheses in the memory. These FDCs basically consist of Bayesian networks, and XPath statements as addressing schema for referencing hypotheses in the repository.

One of the realized FDCs of the office scenario is chosen to explain the approach in more detail: The user is located in front of a computer and occasionally performing a “typing” action. Exemplary images of this scenario are shown in Figure 4. Obviously, it is impossible to perform the action “typing” without having a computer in the scene. If this situation occurs, the involved hypotheses have to be doubted, since they are not expected by the underlying model in the given context.

The Bayesian network representing the described relation of a computer-typing setup is shown in Figure 5 with the particular conditional dependency tables attached to each node.

Based on this definition of FDC, a conflict in the memory content is detected as follows: A Bayesian network defines the expectation of allocation (*beliefs*) of its vari-

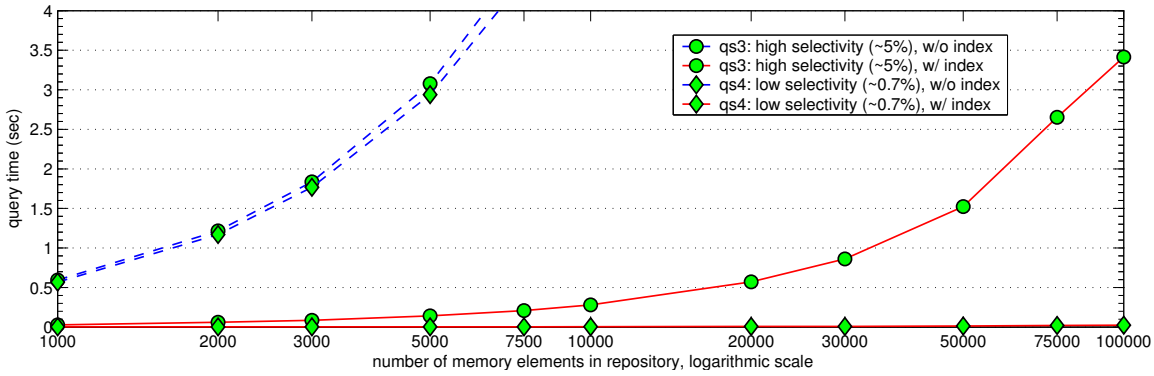


Fig. 6: Query performance of memory element repository.

ables. For consistency validation the value for each variables is derived from the existence of a particular hypothesis in the VAM which comes from the binary set $\{true, false\}$. As stated beforehand, the existence of a hypothesis is ascertained by evaluating the given XPath statements of the FDC on the actual repository content to fill these variables with concrete values. For instance, our system sets $vis\_O\_keyboard = true$ if the XPath-Statement `/OBJECT[CLASS/text()='KEYBOARD']` provides a valid result set. In the context of Bayesian networks, these observations are called *Evidences* $e = \{e_1, e_2, \dots e_m\}$. By considering all evidences in the network a conflict value *conf* can be calculated as a kind of emergence measure defined in [21]:

$$conf(e) \quad = \quad log_2 \frac{\prod_{i=1}^{m} P(e_i)}{P(e)}. \qquad (1)$$

There $P(e)$ denotes the overall probability of the given evidences while $P(e_i)$ are the marginal probabilities of the involved random variables of the Bayesian network. In case of a conflict, the probability $P(e)$ is expected to be small in relation to the product of the probabilities $P(e_i)$ since the evidences are not explained by the given concept. As a consequence, $conf(e)$ will be greater than zero and allows to detect the conflict.

In order to cope with uncertainty of the underlying perception processes, *soft evidences* instead of hard evidences are used for the observable nodes. Instead of exclusive evidences from the set $\{true, false\}$, we allow each variable to have an evidence-vector $\vec{e} = (e_{true}, e_{false})^T$ with $0 \leq e_{\{true,false\}} \leq 1$ and $e_{true} + e_{false} = 1$. The concrete value of a node's evidence is controlled by the reliability of the hypothesis represented by that node, which is transparently accessible via XPath due to the type hierarchy of memory elements. The more reliable a hypothesis is, the "harder" is the evidence. For a reliability factor $r = 1$, the evidence is set to $\vec{e} = (1, 0)^T$ according to

$$(e_{true}, e_{false})^T \quad = \quad (0.5(1+r), 0.5(1-r))^T, \quad (2)$$

and to $\vec{e} = (0.5, 0.5)^T$ for $r = 0$, which is equivalent to an unobserved variable with no evidence. When detecting a conflict the memory process lowers the reliability r_i of the involved conflicting hypotheses i by

$$\bar{r_i} \quad = \quad f \cdot r_i \qquad \text{with } 0 < f < 1, \qquad (3)$$

which in consequence decreases the conflict value. Thus, this EMP realizes universal hypothesis fusion in terms of improving the consistency of the memory content.

4 Results

Evaluating the presented integrative system includes very different aspects. On the one hand, the applicability of the realized XML based Memory Infrastructure in terms of performance has to be evaluated. As the active memory plays a central role in the discussed model for information fusion the throughput and access performance are fundamental to ensure the reactivity of the proposed system. On the other hand, the power of the consistency validation approach is analyzed, by evaluating different FDC in the office scenario. Furthermore, the advances in system integration arising from the presented concepts are outlined.

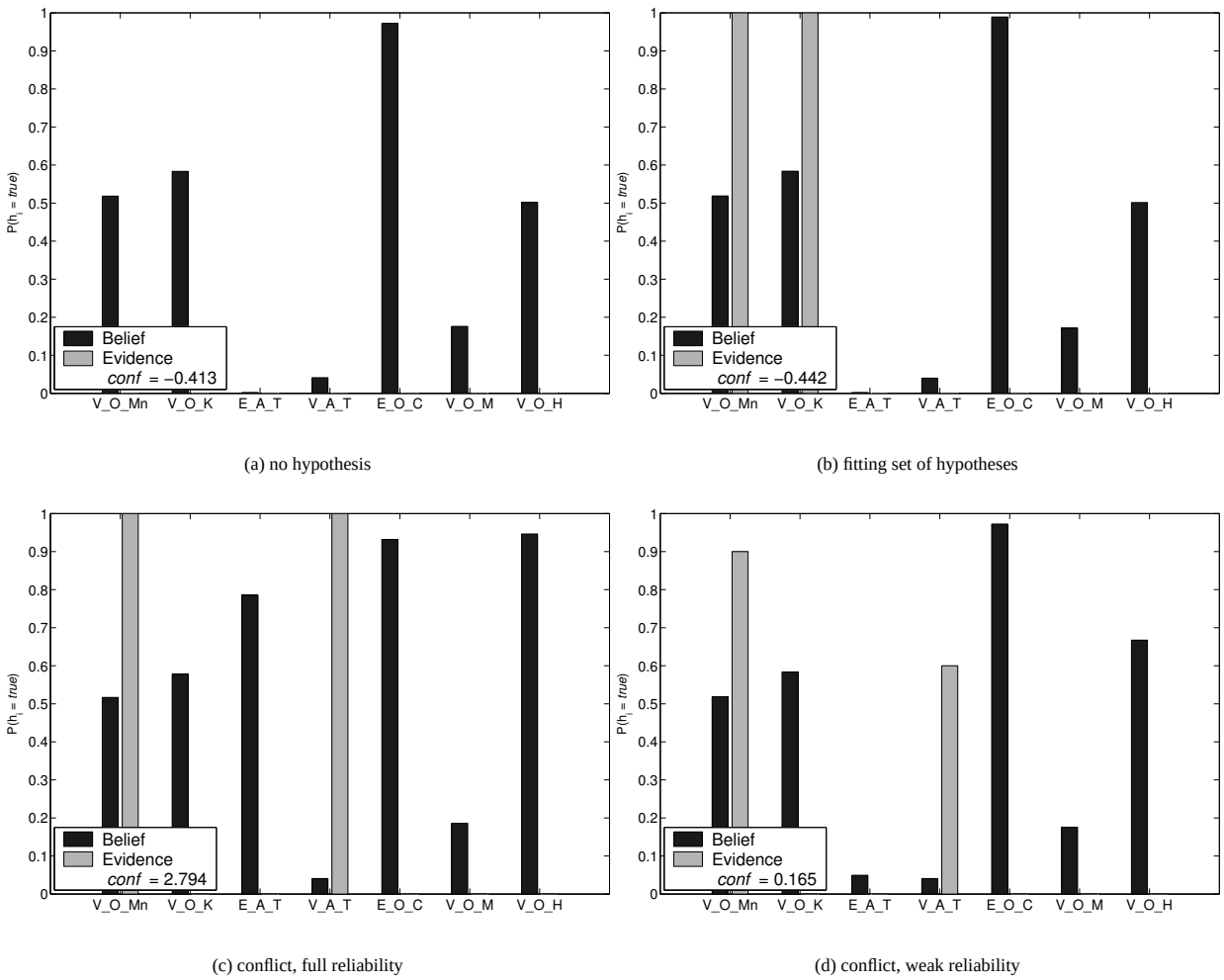
4.1 Memory Infrastructure

When fusing information in an integrated system, it has to be ensured that the technical basis performs well in order to allow a fast system reaction. Therefore a performance analysis of the XML based repository of the memory infrastructure is necessary. Of foremost interest is how query performance scales with larger datasets as they are expected in cognitive vision systems.

Our evaluation method is similar to the application independent Michigan micro-benchmark procedure for XML databases [22] but was adapted to our own dataset consisting of memory elements as shown in Figure 2. Figure 6 exemplarily depicts the mean performance of an attribute equality query like `/OBJECT/REGION/RECTANGLE/COORDS[@w=225]/@w` (*qs3*) that will return a set of XML nodes that match the given condition, described by an XPath statement.

Apart from the size of the dataset the size of the result set is also of interest. A typical query might return less than one percent of the whole dataset (*low selectivity*), no query is expected to exceed result set size of five percent (*high selectivity*) of the whole.

Looking at indexed and non-indexed queries, the latter ones are very expensive in terms of time. Also, in that case, selectivity of a query is irrelevant as disk I/O for a sequential scan of the repository seems to be the limiting factor. In contrast, indexed queries with low selectivity show almost



(a) no hypothesis

(b) fitting set of hypotheses

(c) conflict, full reliability

(d) conflict, weak reliability

Fig. 7: Beliefs and *conf*-value of 4 cases.

constantly excellent performance regardless of repository size. Even better, the performance of the indexed *qs3* query with high selectivity is also sufficient for our application as it takes e.g. $\sim$0.57 seconds to retrieve about 1000 XML hypotheses from a repository with 20000 memory elements (see Figure 6).

Summarizing, the use of XML encoded memory elements with our infrastructure and the ability of indexing the underlying database provides a fast and reliable basis for the fusion of information in active memory instances.

4.2 Consistency Validation

In order to evaluate our consistency validation approach, we defined FDCs on different constellations of objects and actions that are typical for the office scenario. Here, results are presented for the FDC analyzing "typing"-action in context of objects in the computer setup scenario, which consequently implies fusion of hypotheses generated by action and object recognition. A trained FDC is applied to actual memory contents to evaluate the consistency. Therefore, for each assignment of the memory, the conflict value *conf*(e) and the beliefs $P(h_i = true)$ for each hypothesis h_i are calculated, by propagating the evidences obtained from the current memory content into the Bayesian network. Resulting values for different memory configurations with their respective evidences are displayed in Figure 7.

Diagrams 7(a) and (b) depict the results for a consistent memory content in the sense of the evaluated FDC, since *conf*$(e) < 0$ is true for these two configurations. On the one hand, (a) shows the case of having no relevant hypothesis available, while on the other hand, (b) includes high-reliable hypothesis on $vis\_O\_monitor$ (v_O_Mn) and $vis\_O\_keyboard$ (v_O_K), which support each other very well. On the contrary, the two configurations 7(c) and (d) held conflicting hypotheses, as it is indicated by *conf*$(e) > 0$. Both configurations contain hypotheses of $vis\_O\_monitor$ (v_O_Mn) and vis_A_typing (v_A_T) but no hypothesis on $vis\_O\_keyboard$ (v_O_K) which violates the trained model that expects an keyboard to be visible while typing. As stated in section 3.4, in case of a conflict, the reliability of a hypothesis is decreased. The effect of incorporating the reliability using soft evidences is depicted in (d). Doubting hypotheses by lowering their reliability ($r_{v\_A\_T} = 0.6$) allows the reduction of the conflict potential. The dynamic of this process is depicted in Fig. 8, that corresponds to the case (c) with both conflicting hypotheses having an initial reliability of 1. Due to the detected conflict, this reliability is decreased according to

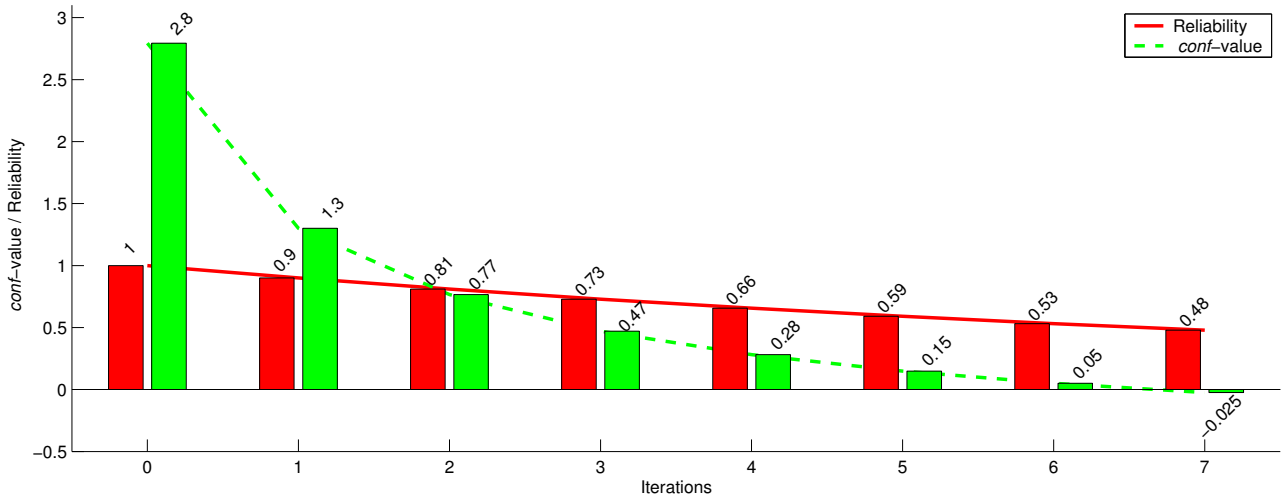


Fig. 8: Time dynamics of consistency validation. Reliability factor is decrease until $conf < 0$.

equation 3 ($f = 0.9$), which on its part decreases the conflict value. Every time the consistency validation is executed, it decreases the reliability until $conf < 0$ to solve the conflict.

4.3 Advances in System Integration

The cognitive vision system outlined in the introduction was developed with first implementations of the memory infrastructure framework. Experience shows that the use of XML helped in defining datatypes which are suitable for every involved project partner. There is no need to reduce the interchanged information type to an artificial least common denominator as this is the case when there is no flexibility allowed in exchanged datatypes. Another aspect useful both for development of the system and the runtime architecture is the ability of the memory interface to integrate new information types without having to physically restart any servers or to redeploy any schemas. The absence of fixed data schemas helps also in restructuring of content and allows for storing of completely new information structures which is useful, e.g., for learning architectures.

Besides the goal to fulfill the functional requirements for an active memory with a flexible approach, our realization is also beneficial for non-functional system integration aspects. One of these additional outcomes is the simplicity and usability through a clear and straightforward API and the use of standards based XML technologies throughout the whole framework. The proposed unified data model results in changeability, easier adaptation and integration of modules. Furthermore, openness of distributed memory instances allows developers to retrieve information in a standard fashion using declarative queries.

Low coupling as an important feature of a modular architecture is reached through a combination of memory infrastructure and XCF. The memory instances itself decouple the memory processes and serve as an information mediator while the XCF framework provides location and access transparency for components. This facilitates the exchange of components and leads to high robustness against component failure.

Debugging and Evaluation of integrated cognitive vision systems is supported by simulation of components using XCF for replaying recorded memory data with a so called module simulator. If the memory content metadata provides time information (as shown in figure 2), whole architectural layers can be replaced by this simulation tool as if they were online available. Development and evaluation of different algorithms or system configurations on comparable data has been much easier with this feature. The declarative nature of memory elements and IMPs, simple debugging and the changeability of specifications also qualifies the active memory framework for rapid prototyping in cognitive vision research.

5 Conclusion and Outlook

The successful integration of the visual active memory and the online ability of the resulting system architecture in our collaborative cognitive vision project is a first proof of concept for the proposed framework for information fusion. To allow fusion of memory elements, in particular of hypotheses using their context, an attempt to use Bayesian networks as a basis for an extrinsic memory process proved an eligible concept for our cognitive vision system. The proposed conflict measure is universally applicable on any type of functional dependency concept and provides the possibility to rate knowledge stored in the memory. In conjunction with the presented concept of a generic hypothesis data type and memory infrastructure, this provides a new and unique technique to realize a cognitive active memory, which can store, fuse and actively consolidate knowledge obtained from different perceptual modalities. The developed memory infrastructure provides a flexible runtime-environment for intrinsic memory processes, exemplarily shown for the forgetting process, and easy access to the memory contents.

Furthermore, the non-functional advances of the integrative system achieved by the use of XML as data exchange format, allow for easy integration of third-party modules as information sources into the system, which is especially important for large-scale cognitive systems.

Acknowledgements

This work was supported by the VAMPIRE project[23] of the European Union (IST-2001-34401).

References

[1] H.I. Christensen. Cognitive (Vision) Systems. *ERCIM News*, 53:17–18, April 2003.

[2] J.L. Crowley and H.I. Christensen, editors. *Vision as Process*. Springer, 1995.

[3] ECVision, 2003. `http://www.ecvision.info`.

[4] H. Siegl, M. Brandner, H. Ganster, P. Lang, A. Pinz, M. Ribo, and C. Stock. A mobile augmented reality system. In *Exhibition Abstracts of ICVS*, pages 13–14, 2003.

[5] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. CVPR*, volume 1, pages 511–518, 2001.

[6] C. Gräßl, T. Zinßer, and H. Niemann. Illumination insensitive template matching with hyperplanes. In *Proc. Pattern Recognition Symposium*, pages 273–280, 2003.

[7] G. Heidemann, R. Rae, H. Bekel, I. Bax, and H. Ritter. Integrating context free and context-dependent attentional mechanisms for gestural object reference. In *Proc. ICVS*, volume 2626 of *Lecture Notes in Computer Science*, pages 22–33. Springer, 2003.

[8] J. Fritsch. *Vision-based Recognition of Gestures with Context*. PhD thesis, Bielefeld University, 2003.

[9] H. Cruse. The evolution of cognition – a hypothesis. *Cognitive Science*, 27(1):135–155, 2003.

[10] E. Tulving. Organization of memory: quo vadis? In M.S. Gazzaniga, editor, *The Cognitive Neurosciences*, pages 839–847. MIT Press, 1995.

[11] L. Seligman and A. Rosenthal. The impact of XML on databases and data sharing. *IEEE Computer*, 34(6), 2001.

[12] M. Birbeck, J. Duckett, and O.G. Gudmundsson. *Professional XML*. Wrox Press Inc., 2nd edition, 2001.

[13] R. Murphy. Biological and cognitive foundations of intelligent sensor fusion. *IEEE Trans. on Systems, Man, and Cybernetics*, 26(1):42–51, 1996.

[14] M. Shaw and D. Garlan. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall, 1996.

[15] Berkely DB XML, Sleepycat Software, 2003. `http://www.sleepycat.com/products/xml.shtml`.

[16] Sebastian Wrede, Jannik Fritsch, Christian Bauckhage, and Gerhard Sagerer. An XML based framework for cognitive vision architectures. In *Proc. ICPR*, 2004.

[17] The Internet Communications Engine, ZeroC Inc., 2003. `http://www.zeroc.com/ice.html`.

[18] A. S. Tanenbaum and M. van Steen. *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2002.

[19] S. Wachsmuth and G. Sagerer. Bayesian networks for speech and image integration. In *Proc. of 18th National Conf. on Artificial Intelligence*, pages 300–306, 2002.

[20] J.H. Piater and R.A. Grupen. Feature learning for recognition with bayesian networks. In *Proc. ICPR*, pages 1017–1020, 2000.

[21] F.V. Jensen. *Bayesian Networks and Decision Graphs*. Statistics for engineering and information science. Springer, New York, Berlin, Heidelberg, 2001.

[22] K. Runapongsa, J.M. Patel, H.V. Jagadish, Y. Chen, and S. Al-Khalifa. The michigan benchmark: Towards XML query performance diagnostics. In *Proc. VLDB Conference*. Morgan Kaufmann, 2003.

[23] The Vampire Project, 2004. `http://www.vampire-project.org`.