Unsupervised Filtering of XML Streams for System Integration

Ingo Lütkebohle, Sebastian Wrede and Sven Wachsmuth

Faculty of Technology, Applied Computer Science
Bielefeld University, P.O. Box 100131, 33501 Bielefeld, Germany

Abstract. In the last years, computer vision research is more and more shifting from algorithmic solutions to the construction of active systems. One novel approach to system construction combines data- and event-driven architectures, concentrating on the flow of information between components. A challenge in data-driven architectures is to optimize communications behavior without changing component implementations. For example, in computer vision, a common problem is that low-level components produce many very similar results whereas on a higher level, only significant changes are of interest. This distinction can be defined as a pattern recognition task that analyzes the data flow in the system. In the following, we will first give a short introduction into the architecture, then describe a generic solution for data-flow reduction based on XML distance metrics. We present first results on the application of this component in an integration framework for a vision-based Human-Computer-Interface within an augmented reality scenario.

1 Introduction

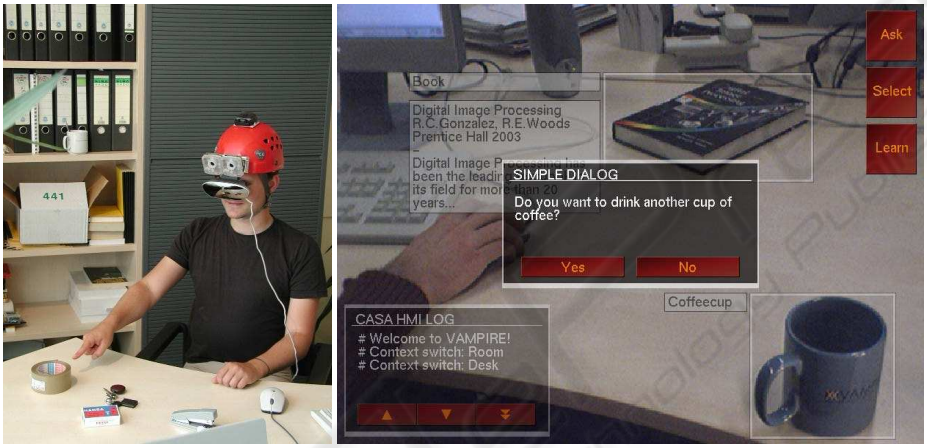
Most systems in computer vision are realized as a single multi-stage process mapping images to higher-level descriptions. While this is sufficient for specific pre-defined applications in constrained settings, more general vision systems need to combine different multi-stage components like object and action recognition, tracking, self-localization, or scene analysis [1]. From a system engineering perspective, the decoupling of these components is a key issue in the construction of more flexible, multiple-purpose vision systems [2]. However, the decoupling of processing components leads to a high communication load of the system. Lower level modules need to serve different purposes and, therefore, permanently run on a maximum updating frequency. For example, in interactive vision scenarios like augmented reality applications, object recognition results are used for multiple purposes like recognizing action contexts, scene classification, initialization of tracking, scene labelling, etc. In order to control the data flow in such systems, a generic framework functionality has to be developed which dynamically reduces data volume while keeping relevant information with regard to higher-level components.

Ahmad K. (2005).
Selective Visual Attention in Electronic Video Surveillance.
In *Proceedings of the 5th International Workshop on Pattern Recognition in Information Systems*, pages 204-212
DOI: 10.5220/0002564302040212

1.1 VAMPIRE - An Interactive Vision Scenario

Within the VAMPIRE[1] project a generic system integration platform for interactive cognitive vision systems is being developed. The project aims at the realization of a personal augmented reality assistant including capabilities for localisation, context-awareness, online learning, and information retrieval [3]. The current demonstration system serves as a basis for implementing and evaluating our approach for compacting data.

Figure 1(a) gives an impression of the user interacting with the system in an office environment in which the scene is observed from the users viewpoint by the two Fire-i cameras being part of the augmented reality kit worn by the user.



(a) AR-Kit for HCI (b) Exemplary view through HMD

Fig. 1. User with AR-Kit; Augmented scene as displayed in the users HMD.

Figure 1(b) shows a scene augmented by object recognition results and dialog displays. For the object recognition task, a two-way strategy is chosen: The detection of objects with a common and only slightly varying appearance is achieved by a detector which is quite similar to the one introduced by Viola and Jones [4]. For a fast and online trainable object recognition, VPL classifiers are applied [5]. Further recognition modules provide complementary data, e.g. about the users head pose or his 3D position. The additional information displayed to the user depends on the predicted intention of the user which is inferred by contextual analysis [6]. For more details, see [7].

In this scenario, the goal is to increase the overall responsiveness of the system by compacting the flow of processing results reported by the various components, e.g. symbolic object recognition results. They are input to the compacting component. Besides the symbolic content, object hypotheses include a number of *meta-level variables* like position, time, or reliability.

The developed approach has to fit similar needs as the system integration framework developed for VAMPIRE, i.e. ease of use, flexibility to arbitrary data structures, and

[1] Visual Active Memory Processes and Interactive REtrieval

independence of specific processing components. The system integration framework will be described in section 2. XML messages form the basis of communication in that approach. Because XML data is coding its own structure, the compacting method can be realized as a framework component. A generic, XML-based similarity metric to compare data is described in section 3 and the compacting approach using that metric in section 4. Finally, we show some first studies and evaluation results in the VAMPIRE system and discuss future work. Prior to that, we will now give a short overview of related work in the area of system integration for vision systems and analysis of XML streams.

1.2 Related Work

Different approaches and tool-kits have been proposed to support the system and knowledge engineering tasks that come along with a specific application [8–11]. However, most of these suffer from overly rigid control strategies, narrowly defined domains, and/or insufficent support for distributed asynchronous processing and integration of externally developed modules. Resource control is typically centralized, contradicting distributed and asynchronous processing needs. Although more flexible computer vision frameworks have been proposed recently [12] or are under development [13, 14], there is a huge need for computer vision frameworks that facilitate data integration and reasoning over certain periods of time as well as offer meta-level reasoning capabilities for unsupervised resource control.

As the high-level messages exchanged in our system are XML-based, XML forms the input to our compacting process. The importance of structural information is emphasized by [15] who show that tags cannot simply be added to the content feature-set. Both [16] and [17] use tree-edit distances, the former on streams, the latter on heterogenous document collections. In both cases, structural differences are considered crucial for the comparison. Additionally, [18] makes uses of tree-edit information to create merged representative documents.

We do not compute tree-edit distances, but interpret XML elements as *annotations* of the content. Nesting is taken to convey semantic differences between content while element names and hierarchy are used to identify matching points where meaningful comparison is possible. One could say that we take the “mark-up” metaphor for XML quite literally.

Prior work on distance metrics for structured data will be discussed with the algorithm proposed in section 3.

2 Active Memory Infrastructure

Accompanying the development of all the recognition processes described in section 1.1, a system architecture had to be developed which allows for a flexible connection of all these components. This task led us to the development of a generic software framework for integration of computer vision systems, the *Active Memory Infrastructure* (AMI).

The interactive application scenario imposes several constraints on the software framework. System components should be able to exchange data with soft real-time performance and to memorize episodes, events and scenes. Given these requirements we focussed on a shared repository and the distribution of algorithms. The AMI framework consists of (1) the *XML enabled Communication Framework* (XCF) [19] that allows to distribute components over several computing nodes, (2) the *Active Memory* server and interface for coordination and data management, and (3) a supporting library named *XMLTIO* that supports users with an XPath-based API for simple XML processing. Although written in C++ for performance reasons, there are also bindings for Java and Python. The complete integration software is licensed under the GPL and available for download at Sourceforge [20].

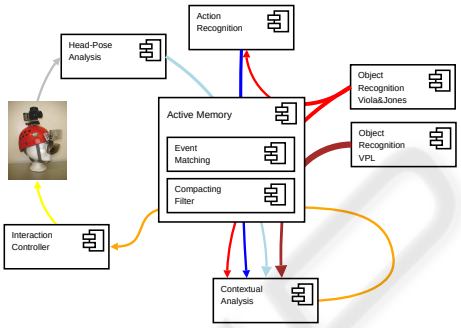
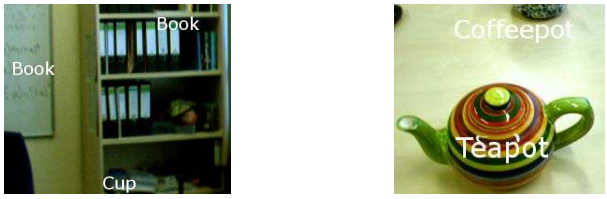


Fig. 2. Exemplary data-flow in the AMI.

Focussing on great flexibility and ease of use, all information flow (e.g. object recognition results) within the VAMPIRE components is based on XML documents that can reference attached binary data (e.g. images). XCF supports (non-)blocking remote method invocation and publisher-subscriber communication semantics. Exposed methods are bound and invoked dynamically, with schemas optionally providing runtime type safety. The ICE communications engine forms the basis of this component and was chosen for its high performance, especially low latency calls.

XML data like objects, actions or 3D locations and binary data like mosaics of detected planar regions are fed into an active memory server and can be retrieved via XPath statements. Coordination between the components is provided by a flexible event-notification mechanism. The event manager of the active memory server is co-located with the persistence back-end, a native XML database, the Sleepycat DB XML. Event subscriptions can specify an XPath to narrow down documents of interest. Coordination is thus data-driven and not dependent on the presence of specific components.

A prototypical part of the system that exemplifies data flow and interaction between different components and the active memory is given in figure 2. As shown in picture 3(a) parts of the transmitted data may be false when the user looks around, other data may be redundant 3(b), e.g. when the user gazes at a static scene. In this situation object recognition will work well and send nearly identical object recognition results at 50Hz. Both, redundant and obviously false recognition results consume resources of the system in terms of network bandwidth and overall system performance. The question how



(a) Moving (b) Gazing

Fig. 3. Typical object recognition results in an unconstrained environment.

to compact processed information and detect relevant changes in the streams of XML information with a generic framework component will be answered in the following.

3 XML Kernel

We propose a similarity measure that includes a data integration mechanism and can thus process data from a variety of sources coherently. It uses the label information in XML document trees, the element name, to identify comparable values and to transparently handle missing, repeated or re-ordered occurrences of an element or sub-tree.

In any data integration task, care must be taken not to mix up data with different semantics. E.g., in object recognition, the coordinates of an object and its label are not on the same abstraction level. We therefore exploit *nesting* as a generic indicator of semantic differences, taking advantage of an existing and established way of formulating this crucial bit of information.

The similarity measure forms a kernel. It has been shown that many machine-learning methods can be *kernelized* in a straightforward manner, either by using the kernel in place of the scalar product or through a distance measure constructed from the kernel, e.g. $d(x,y) = \sqrt{K(x,x) - 2K(x,y) + K(y,y)}$ [21].

3.1 Kernel Over XML Documents

An XML document is a labeled tree rooted at the *document node*. In the following, for a node n, let $L(n)$ be its label, $V(n)$ its value and $C(n)$ be the set of children and attributes. In the XML infoset, only attributes and text nodes have a value assigned but for the purposes of this paper, we take element value to be composed of the immediate text nodes:

Definition 1 (element value). *The value of an element n with level l is the concatenation of all text nodes with root n and level $l+1$.*

For the kernel definition, two cases are special: The empty comparison and non-matching labels. For these, $k(0,0) = 1$ respectively $L(s) \neq L(t) : k(s,t) = 0$.

For nodes, we adopt the idea of Gärtner et al. [22] to exploit possible functional dependencies by combining the similarity of parent and children:

$$k(s,t) = k_{L(s)}(V(s),V(t))k(C(s),C(t))$$

Nodesets, despite the name, have document order but may be treated as both a set or a list, with the corresponding kernels (and using the above). For sets: $k(u,v) = \sum_{i,k} k_n(u_i, v_k)$ and for lists: $k(u,v) = \sum_i^n k(u_i, v_i)$. Last, but not least, for basic numeric values, a Gaussian: $k(a,b) = e^{-|a-b|^2/h^2}$ and for strings, a Hamming similarity: $k(m,n) = 1/k \sum_{i=1}^{k} \delta(m_i, n_i)$ is applied.

4 Compacting Data Flows

To reduce the burden of redundant and/or bad results we filter elements in the Active Memory. Firstly, the amount of new information present will be estimated and only if the change is *big enough* elements will be forwarded for further processing. Secondly, elements are *clustered*. When a close group is found, it is updated, otherwise a new group will be created. This is called *compacting*.

Compacting at the framework level allows us to take advantage of global information, e.g. when two redundant recognizers are present. For the implementer, it is beneficial to have a dedicated component for relevancy detection that can be changed to adapt to new challenges. Last, but not least, our approach allows components to selectively bypass compaction to receive all elements.

4.1 Detecting Relevant Elements

Detection of relevant elements requires an indication of the amount of new information contained, relative to the elements already present in the memory. We use the violation of the present clustering to determine significance: New clusters are considered relevant. To determine this, we observe the minimum distance between a new element and the existing clusters over time and estimate the change using a moving average for the parameters of a normal distribution $p_I \sim \mathcal{N}_{\mu,\sigma}$.

Let I be the current number of elements, and d_i the minimum distance observed at element number i, then the sample mean is $\bar{\mu}_I = \frac{1}{k} \sum_{i=I-k}^{I} d_i$ and sample variance analogous. A new cluster is created if $p_I(d_{i+1}) < t$.

The parameter k allows for adaption to the result rate of the system, in our experiments it is based on frame rate. t has been chosen constant (0.05), with the variability in the system captured by the density p_I.

4.2 Online Clustering

In the VAMPIRE system, elements arrive one-by-one, not batched and due to user interaction stationarity can only be assumed short-term. The relevancy detection determines creation of new clusters but aims at fast reaction time more than at clustering quality and it has to, because of the limited amount of information and the strictly limited processing time. Fortunately, over time good clusters will acquire more support while outliers won't and this can be used to achieve good clustering quality in an online setting by determining cluster size and removing unreasonably small ones. The exact cut-off to choose depends on the variance in the input. In our experiments the mean has proven a good choice.

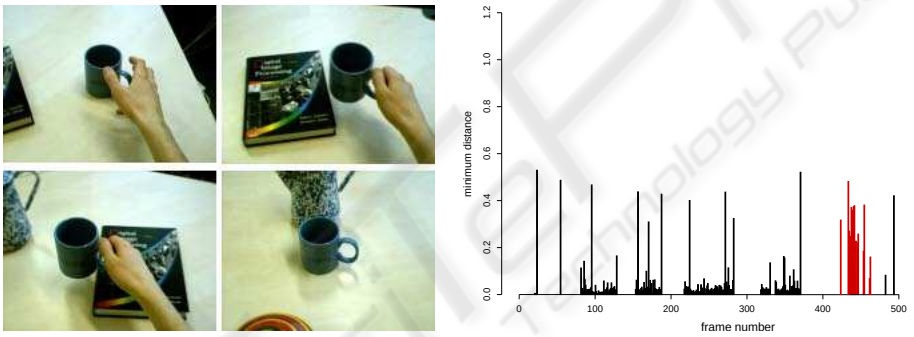
5 Evaluation

When grouping data, especially with a new similarity metric, it is crucial to ascertain that groupings are not an artefact of the algorithm but represent genuine structure in the data. Therefore, we have evaluated the kernel on real-world data and analyzed how the results relate to user activity and wether our relevancy heuristic matches at meaningful points in time.

Results are from a 600 frame (50 second) sequence with multiple objects, a moving viewpoint and objects being moved around.

5.1 Evaluation of Relevancy Detection

To verify that our simple minimum distance approach can identify relevant changes in the environment, we have computed a clustering and tracked the minimum achievable distance. The graph in figure 4(b) shows the distance for a book present in the scene. Note, how distance is high at the beginning and end of a period in time (where distances are zero, the book was not present).



(a) frames 436, 443, 449, 453 (b) minimum achieved distance

Fig. 4. Relevancy detection results with example sequence.

Note the shaded/red region which exhibits continually high distances. We have selected four frames from that period and show them in figure 4(a). As you can see, the book moves rapidly through the view and is partly obstructed at the same time, which explains why a lot of change is perceived.

5.2 Evaluating Meta-Variable Similarity Metric

Figure 5 shows pairwise distances of object recognition results pertaining to the book, from the same sequence as in section 5.1, plotted using multidimensional scaling (MDS). Clusters are surrounded by ellipses, color and symbols redundantly specify distinct periods of presence for the book. Most clusters exhibit high internal coherence and strong separation. One cluster contains results from two adjacent periods of time. The results marked with a (red) diamond pertain to the episode shown in figure 4(a), with a rapidly moving, partly obstructed object, these are grouped together with the outliers.

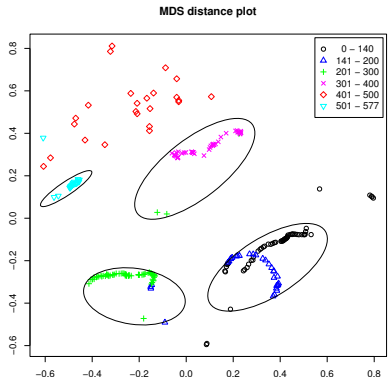


Fig. 5. Book clusters.

6 Conclusion

This paper introduced a flexible framework for data-driven software integration which was used successfully for the realization of vision-based HCI in an augmented reality application. The main contribution was the presentation of a framework component for unsupervised reduction of data-flow volume in XML streams based on a new application of kernels for structured data and a compacting approach utilizing a clustering method. Regarding a first evaluation, we showed how this metric can be used to detect relevant changes in streams of XML data and that the similarities relate to the events in the application scenario.

Future work will focus on more robust methods for online clustering, as well as the evaluation of the overall system performance in terms of an increased responsiveness and lower resource consumption.

Finally, we think, that as the proposed method is based on a generic XML distance metric, it will make other kernel-based approaches from pattern recognition available for direct use in XML-based information systems. Furthermore, we believe that compacting of XML streams may also be usefully applicable in other XML exchanging information architectures.

References

1. Aloimonos, Y.: Active Vision Revisited. In Aloimonos, Y., ed.: Active Perception. Lawrence Erlbaum Associates (1993)
2. Wachsmuth, S., Wrede, S., Hanheide, M., Bauckhage, C.: An Active Memory Model for Cognitive Computer Vision Systems. Künstliche Intelligenz (2005) to appear.
3. Bielefeld University: The Vampire Project (2005) http://www.vampire-project.org.
4. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proc. CVPR. Volume 1. (2001) 511–518
5. Heidemann, G., Rae, R., Bekel, H., Bax, I., Ritter, H.: Integrating context free and context-dependent attentional mechanisms for gestural object reference. In: Proc. ICVS. Volume 2626 of LNCS., Springer (2003) 22–33

6. Hanheide, M., Bauckhage, C., Sagerer, G.: Memory Consistency Validation in a Cognitive Vision System. In: Proc. ICPR. Volume 2., IEEE (2004) 459–462
7. Bauckhage, C., Hanheide, M., Wrede, S., Sagerer, G.: A Cognitive Vision System for Action Recognition in Office Environments. In: Proc. CVPR. Volume 2. (2004) 827–832
8. Draper, B., Brolio, J., Collins, R., Hanson, A., Riseman, E.: The Schema System. Int. J. Comput. Vision **2** (1989) 209–250
9. Mundy, J., Binford, T., Boult, T., Hanson, A., Beveridge, J.R., Haralick, R., Ramesh, V., Kohl, C., Lawton, D., Morgan, D., Price, K., Strat, T.: The Image Understanding Environments Program. In: Proc. of DARPA Image Understanding Workshop. (1992) 185–214
10. Rasure, J., Kubica, S.: The Khoros Application Development Environment. In Christenson, H., Crowley, J., eds.: Experimental Environments for Computer Vision and Image Processing. World Scientific Press, Singapore (1994)
11. Williams, T.: Image understanding tools. In: Proc. ICPR. Volume IV. (1990) 606–610
12. Draper, B., Bins, J., Baek, K.: ADORE: Adaptive Object Recognition. Videre **1** (2000)
13. Ponweiser, W., Umgeher, G., Vincze, M.: A reusable dynamic framework for cognitive vision systems. In: Workshop on Computer Vision System Control Architectures. (2003)
14. Bruyninckx, H.: Open Robot Control Software: the OROCOS project. In: Proc. ICRA. (2001) 2523–2528
15. Doucet, A., Ahonen-Myka, H.: Naïve clustering of a large XML document collection. In: INEX. (2002) 81–87
16. Garofalakis, M., Kumar, A.: Correlating xml data streams using tree-edit distance embeddings. In: PODS '03: Proc. 22nd ACM SIGMOD Symposium on Principles of Database Systems, ACM Press (2003) 143–154
17. Dalamagas, T., Cheng, T., Winkel, K.J., Sellis, T.K.: Clustering XML Documents Using Structural Summaries. In: EDBT Workshops. (2004) 547–556
18. Costa, G., Manco, G., Ortale, R., Tagarelli, A.: Clustering of XML Documents by Structure based on Tree Matching and Merging. In: SEBD Workshops. (2004) 314–325
19. Wrede, S., Fritsch, J., Bauckhage, C., Sagerer, G.: An XML Based Framework for Cognitive Vision Architectures. In: Proc. ICPR. Volume 1. (2004) 757–760
20. Wrede, S.: Active Memory Infrastructure (2005) Software and documentation available at http://xcf.sf.net/.
21. Haussler, D.: Convolution Kernels on Discrete Structures. Technical Report UCS-CRL-99-10, UC Santa Cruz (1999)
22. Gärtner, T., Lloyd, J.W., Flach, P.A.: Kernels and Distances for Structured Data. Machine Learning **57** (2004) 205–232