Modular Design of Media Retrieval Workflows using ARIA*

Lina Peng[1], Gisik Kwon[1], Yinpeng Chen[1], K. Selçuk Candan[1], Hari Sundaram[1], Karamvir Chatha[1], and Maria Luisa Sapino[2]

[1] Arizona State University, Tempe, AZ 85287, USA
{lina.peng,gkwon,yinpeng.chen,candan, sundaram}@asu.edu
[2] University of Torino, Italy
{mlsapino}@di.unito.it

Abstract. In this demo, we present the use of the ARIA platform for modular design of media processing and retrieval applications. ARIA is a middleware for describing and executing media processing workflows to process, filter, and fuse sensory inputs and actuate responses in real-time. ARIA is designed with the goal of maximum modularity and ease of integration of a diverse collection of media processing components and data sources. Moreover, ARIA is cognizant of the fact that various media operators and data structures are adaptable in nature; i.e, the delay, size, and quality/precision characteristics of these operators can be controlled via various parameters. In this demo, we present the ARIA design interface in different image processing and retrieval scenarios.

Keywords: Media retrieval workflows, modular design, image retrieval

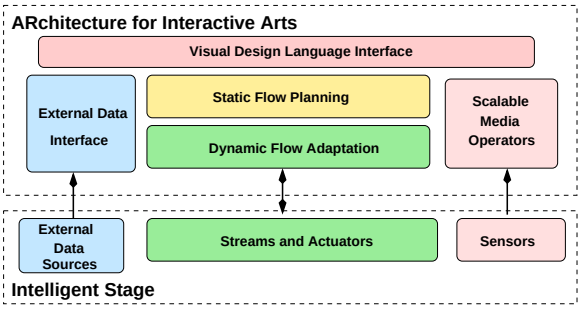
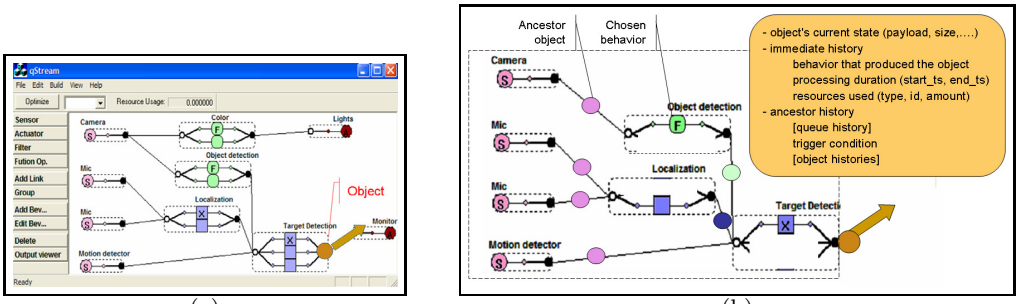


Fig. 1. ARIA modular media processing and retrieval middleware overview

1 Introduction

ARIA [4] is a middleware for describing and executing media processing workflows to process, filter, and fuse sensory inputs and actuate responses in real-time (Figure 1). In this demo, we present the use of the ARIA platform for modular design of media processing and retrieval applications. The objective of ARIA is to

* This research is funded by NSF grant # 0308268, "*Quality-Adaptive Media-Flow Architectures to Support Sensor Data Management.*"



(a) (b)

Fig. 2. (a) An example workflow (note that operators have multiple, alternative implementations to choose from) and (b) visual representation of the history of an object

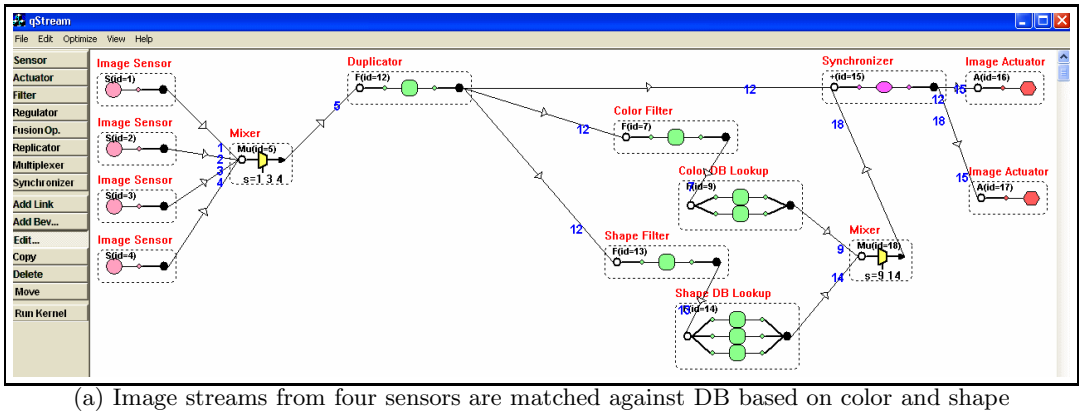
incorporate real-time and archived media into live performances, on-demand [4]. This involves development of (1) an adaptive and programmable kernel that can extract, process, fuse, and map media processing workflows while ensuring quality of service guarantees, (2) a specification interface capable of specifying the components of the media processing workflows, and (3) QoS scalable operators.

2 Overview of ARIA

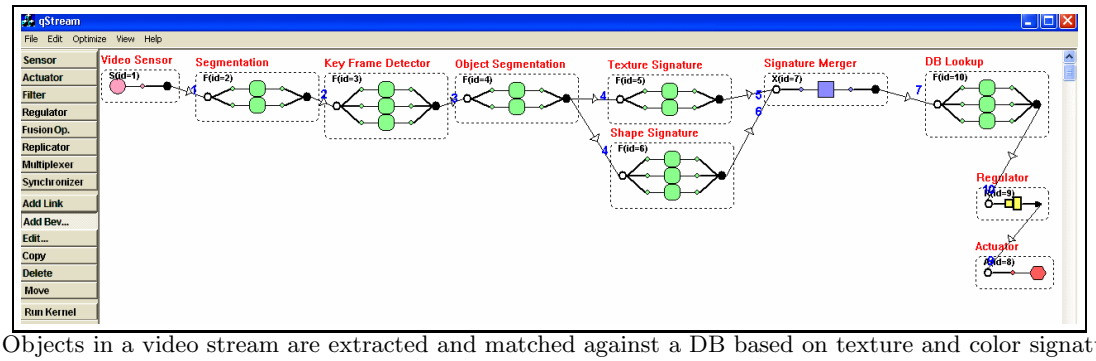
ARIA media processing workflows are modeled as directed graphs where nodes represent sensors, filters, fusion operators, and actuators, while edges represent connections that stream objects between components [5]. The basic information unit is a data object (Figure 2(a)). Depending on the task, an object can be as simple as a numeric value (such as an integer denoting the pressure applied on a surface sensor) or as complex as an image component segmented out from frames in a video sequence. Filter and fusion operators provide analysis, aggregation, and filtering semantics. In particular, they may perform complex media processing and database lookup tasks.

Each object in ARIA is annotated with a header, which includes an *object history descriptor*, consisting of the set of *resource usage stamps* and *timestamps* acquired by the object's predecessors as they go through various operators (Figure 2(b)). Among other things, the history descriptor enables the synchronization of objects in the system based on various applicable temporal criteria, queue management decisions, and per-object evaluation of trigger conditions.

Each ARIA operator has a number of input and output queues and a set of behaviors. Each behavior is essentially a different implementation, with different processing delay and quality characteristics. A given behavior of an operator can be executed only when its execution conditions are satisfied. A behavior may not be in an executable state for various reasons, including (but not limited to) resource shortages. The behavior trigger constraints are described in terms of local hardware resources, temporal regulation of the service stream, object property and history (size, precision), and end-to-end workflow conditions (e.g. end-to-end optimization).



(a) Image streams from four sensors are matched against DB based on color and shape



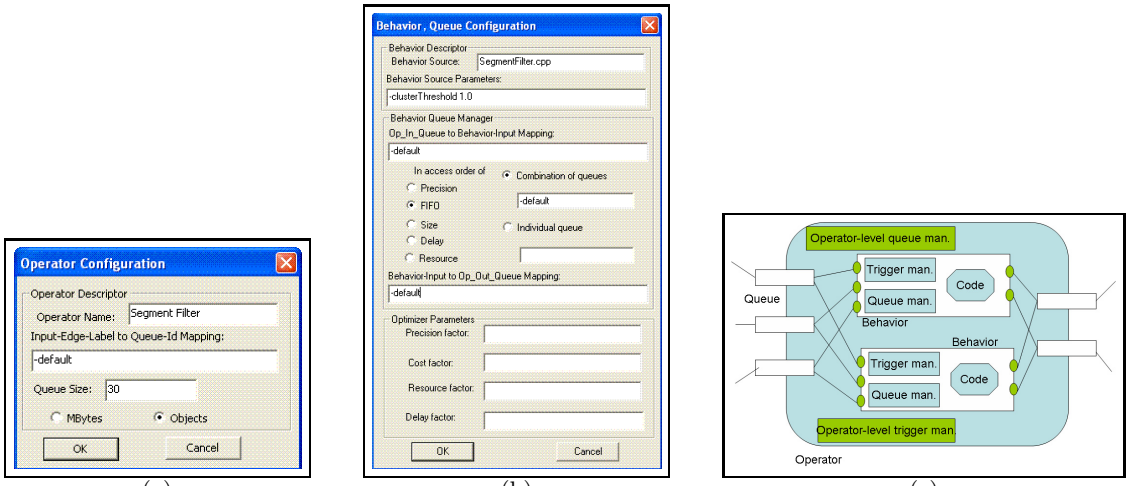
(b) Objects in a video stream are extracted and matched against a DB based on texture and color signatures

Fig. 3. Two media processing and retrieval scenarios

When there are multiple behaviors of an operator ready for triggering, it is the job of the ARIA kernel to pick the most appropriate one, based on the quality, delay, or resource constraints [1]. The way the behavior picks its inputs is also governed by resource, time, and quality constraints. Each behavior can sort and use the objects in the input queues based on different criteria (size, quality, recency etc.). When the number of input combinations to consider is larger than the capacity, then system sheds (not the individual queued objects but) combinations of objects that are not promising candidates. Therefore, each behavior also has an input combination shedding model [2].

3 Demo Scenarios

Figure 3 depicts two media retrieval scenarios. In both cases, real-time sensory data are processed, relevant features are extracted through filters, and databases are looked up based on these extracted features. Note that both scenarios include operators with multiple implementations, which will be chosen by the ARIA kernel based on the appropriate optimization and adaptation criteria. Also, the scenarios include user-defined operators as well as system provided operators (such as synchronizers and duplicators).



(a) (b) (c)

Fig. 4. (a) A filter operator description interface, (b) behavior description interface, and (c) an operator with two behaviors (only one of them active at a time)

Figure 4 shows the operator and behavior description interfaces. Note that the operator descriptor is simple in the sense that it only describes the queue properties of the operator. The behavior descriptor, on the other hand, describes the code which implements the behavior, possible input parameters, as well as how a given behavior uses the input queues and default parameters used for optimization and adaptation.

4 Conclusion

In this demo, we present ARIA middleware and its interface for defining media processing and retrieval workflows, especially suitable for sensory applications. In particular, the framework enables explicit description of alternative implementations of the operators. The ARIA kernel also enables optimization and adaptation of complex workflows through localized as well as end-to-end decision making. Currently, the ARIA kernel is being extended for distributed execution and adaptation of media processing and retrieval workflows.

References

1. Lina Peng, *et al. Optimization of Media Processing Workflows with Adaptive Operator Behaviors* accepted for publication at the MTAP Journal, 2005.
2. Lina Peng and K. Selçuk Candan. *Confidence-driven Early Object Elimination in Quality-Aware Sensor Workflows,* DMSN 2005.
3. Lina Peng, *et al. Media Processing Workflow Design and Execution with ARIA.* Demonstration at the ACM Multimedia Conference 2005.
4. Lina Peng *et al. ARIA: An Adaptive and Programmable Media-flow Architecture for Interactive Arts,* ACM MM Inter. Arts Program, 2004.
5. K. Selçuk Candan, G. Kwon, L. Peng, and M.L.Sapino. *Modeling Adaptive Media Processing Workflows.* ICME 2006.