New E-learning Vectorial Video Approach

José Luis Sánchez Sánchez[1],Jorge Tejedor Cerbel[1],

Jésus Bernal Bermúdez[2],Abraham Gutiérrez Rodríguez [2],Jesús Bobadilla Sancho [2]

[1]Dpto. de Organización y Estructura de la Información
[2] Dpto. de Informática Aplicada

Escuela Universitaria de Informática, Carretera de Valencia, Km 7, 28031 Madrid, Spain

jlsanchez@eui.upm.es, jtejedor@eui.upm.es
jbernal@eui.upm.es, abraham@eui.upm.es, jbobi@eui.upm.es

XI Congreso Argentino de Ciencias de la Computación
IV Workshop de Tecnología Informática Aplicada en Educación

Abstract.

The essential purpose of this paper is to describe a framework, in a simple and complete way, which enables the concurrent visualization of audio/video streaming combined with its corresponding synchronous vectorial information. The architecture presented uses Java JMF API, which offers a simple, robust, extendable, multi-platform solution. It also requires no direct maintenance as regards the incorporation of future video cameras and *codecs* that come out on the market. Instead of the typical separate presentation of videoconferencing and its corresponding preset additional information, we offer the possibility of combining the video image with real-time vectorial information. All the combined bitmap and vectorial information is shared and can be interactively modified by the clients which are using the visualization framework. The current prototype offers a complete API to implement applications based on multiple audio/video sessions that provide a new e-learning vectorial video approach.

Key words.

Video Streaming, JMF, API, Real-Time.

Introduction

Collaborative visualization tools can improve the efficiency of the design and engineer processes [1]. Currently there are different real-time applications providing different types of cooperative visualization or cooperative design [2]. A good example of cooperative visualization tool is *Vic* [3], a real-time, multimedia application for video conferencing over the Internet. *Vic* is designed with a flexible and extensible architecture to support heterogeneous environments and configurations. *ISABEL* [4] is another characteristic multiconference application; it uses an innovative service concept, which adapts the collaboration sessions to the user needs.

The cooperative design not only keeps down the original functions of the traditional software engineering but also supports the cooperative work; however, the development of application cooperation tools is a difficult and time-consuming job [5]. Modeling the cooperative product design process is a key challenge in building up cooperative and intelligent product design information systems and environments. Since the cooperative design process is characterized by distributed workflow and complicated interaction [6].

With the improved requirements of agility and design efficiency of product design, more and more attention is paid to the platforms supporting design tasks and using multiple cooperative design tools. In this context, it is important to establish frameworks supporting multimedia and multiple design modes [7, 8]. Actually, ANTS [9] provides a useful framework, designed to facilitate computer-supported cooperative work. The concept of a cooperative template is in order to control and manage design tasks on the Net [10]; different designs and implementations [11] use the cooperative template resource.

To aid the development of multimedia applications, Sun Microsystems© provides Java Media Framework (JMF) API [12]. The most evident advantage of its use is the possibility to access the video and audio signal through a wide group of classes and interfaces; a less immediate but very important advantage is that we can rely on Sun Microsystems to provide support for new codecs and cameras that come out on the market.

Unfortunately, using the JMF API appropriately is not at all simple. The documentation that exists is based on the document JMF API Guide [13] and the few books published on the subject in which it does not specify how to access the individual frames, or the detailed way to create effects. The rest is an open search of the Internet, where at most we can find examples of the general use of streams, without reaching the level of pixel processing.

The visualization framework presented in this paper is implemented using JMF and allows a real-time video image to be integrated with different types of vectorial information, which are then combined inside each video frame by simply modifying the pixel details. The following sections of the paper will explain the main goals of the RTCIM framework, its scope, upper level architecture, communication strategies and semantic representation; finally we provide a graph with the experimental performance results that the framework can reach depending on the different dynamic loads we can apply.

Scope of the Framework

The framework presented (Figure 1) can handle several sources of audio and video real-time streams, as well as vectorial stream sources (slides, commentaries, isolated texts and arrows, etc.). All these streams use the real-time protocol (RTP) [14] to communicate via the network system. People involved in the cooperative process (mostly consumers), receive the streams from the net-

work (using RTP) and are able to request the inclusion of vectorial information (text, circles, rectangles, etc.) within the streams. To achieve this type of request, the TCP/IP [15] protocol is used.
To understand the framework, we can imagine a situation where NASA acts as source of live video coming from the Shuttle. This video is sent to a small group of scientist (consumers) who are trying to solve a difficult problem in real-time. Each scientist can incorporate the necessary vectorial information to the video images to focalise his collaborative idea.

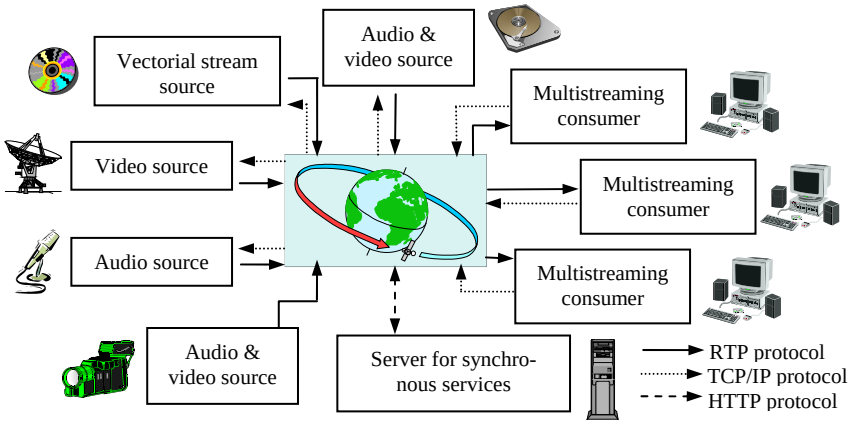

Fig. 1. Framework capabilities

In fact, we are offering a shared blackboard displayed inside the real-time video/s transmitted. This facility is complemented with the typical synchronous services (see Figure 1). At this time, we have finished the software required to use Microsoft Power Point synchronized with the video streams in order to present a more complete scenario to the consumers; this can be a powerful tool, but it is not sufficiently universal as it is platform dependent.
As we can see in Figure 1, source entities act as servers, while consumers act as clients. Sources send the RTP streams (audio, video, vectorial information) combined with the vectorial information provided by the clients' requests. Consumers' requests are received using TCP/IP server diamonds and processed with a JMF Effect [13], which is implemented in the framework presented.

Communication Strategies

The framework presented is just the upper level diagram of our information system; the intermediate level comes from the realization of the technical framework: API. Since we are using Java and JMF, our API contains Java packages (sets of classes and interfaces). Finally, using API, we can easily implement different applications or a highly parameterized project.
The success of the final applications greatly depends on the communications performance. To improve the communications performance it is necessary to adapt the application designs to the bandwidth and the network topology we are dealing with. It is not the same to multicast real-time streams via an intranet system than to multicast them via the open Internet.
Figure 2 shows the communications design using a common intranet bus. Each streaming source broadcasts its real-time streams to all the consumers, receiving the vectorial requests via TCP/IP protocol.

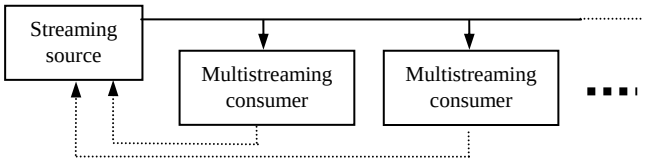

Fig. 2. Common intranet bus communication design

(Dotted line: TCP/IP, continuous line: RTP)

Using the open Internet, the ideal scenario is to have an IP multicast layer available [16]; unfortunately, routers do not support this protocol over the entire Internet. The simplest approach is to open a different session for each consumer, however, this solution is not scalable. The design we have chosen and tested is based on a virtual multicast implemented as a pipeline or chain of consumer transmissions (Figure 3). This solution offers reasonable bandwidth utilization, but is very poor in fault-tolerance and delay aspects.

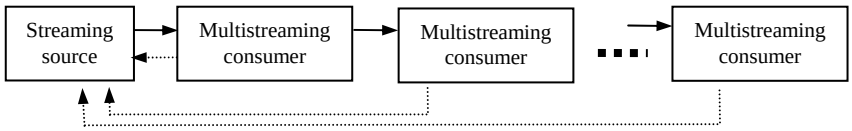

Fig. 3. Internet communication design

(Dotted line: TCP/IP, continuous line: RTP)

It is possible to combine the intranet/Internet approaches, using the Internet to send real-time streams to the different intranets users. In this case, each network would have a computer bridge to receive the streams and act as a streaming source for its own network consumers.

Information System Architecture

Figure 4 shows a simplified diagram of the current RTCIM prototype software. At the lowest level, it is supported by Java Development Kit (JDK) and Java Media Framework, using both their presentation/processing classes and the RTP classes. At the upper level, we separate the source side, the consumer side and the synchronous services.

The ‘StreamSource’ class is responsible for sending both the vectorial streams and the bitmap real-time streams to the network. These streams are merged using the implemented ‘Effect’, producing a resulting bitmap containing the vectorial information. The ‘RTPSend’ class provides the necessary methods to send the streams using RTP.

The ‘VectorialDiamond’ class receives the TCP/IP consumers’ requests (see Figure 1) and stores them in a suitable data-structure to be used in the dynamic merging process implemented on the stream source side. The ‘StreamConsumer’ class receives the RTP streams (‘RTPReception’ class) and, when the Internet communication designs is used, it sends (‘RTPSend’ class) these streams to other consumers (see Figure 3). Finally, the ‘SynchronousServices’ class provides the typical text-oriented synchronous services.

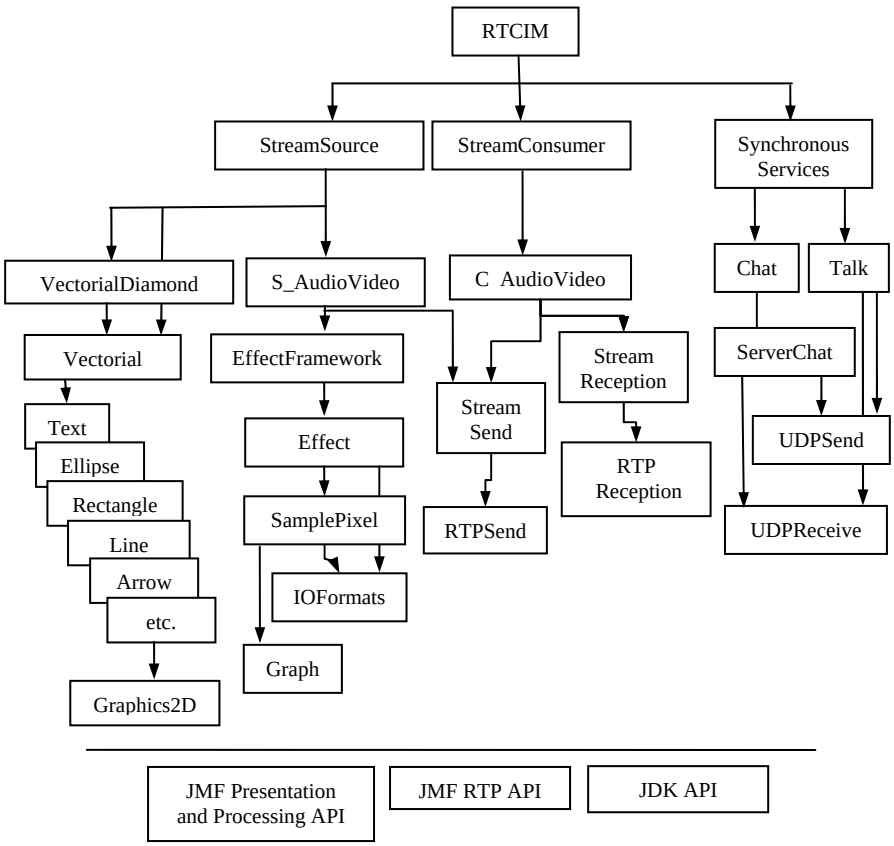

Fig. 4. Architectural design

Semantic representation

Tools and applications should incorporate a suitable semantic layer in order to facilitate high-level queries, interoperability and cross-organizational share. Our framework uses a RDF (Resource Description Framework) based on the DARPA Agent Markup Language (DAML) [17]. DAML sprang from a U.S. government-sponsored effort in August 2000, which released DAML-ONT, a simple language for expressing more sophisticated RDF class definitions than permitted by RDFS. The DAML group soon pooled efforts with the Ontology Inference Layer (OIL) [18]; DAML+OIL, a language for expressing far more sophisticated classifications and properties of resources than RDFS. Since achieving a single common ontology is not realistic, we must deal with some degree of semantic interoperability, supported by effective mechanisms to integrate and reconcile overlapping and inconsistent ontology systems [19].

The semantic kernel of our framework is based on a set of DAML+OIL classes which represent time constraints and vectorial information. The framework presented stores the information that the stream sources send and receive with DAML syntax; more specifically, it stores the TCP/IP vectorial request from the consumers. The following code shows the ‘Rectangle’ and ‘Quadrilateral’ classes as an example.

<rdfs:Class rdf:ID="Rectangle">
<rdfs:subClassOf rdf:resource="#Quadrilateral"/>
<rdfs:comment> RightAngles </rdfs:comment>
</rdfs:Class>

<rdfs:Class rdf:ID="Quadrilateral">
<rdfs:subClassOf rdf:resource="#Polygon"/>
<rdfs:comment>Any four-sided Polygon.</rdfs:comment>
</rdfs:Class>

Results

Currently, RTCIM offers a framework and an API. We have implemented several application prototypes to test the different functionalities provided by the API classes; we are also implementing a telemedicine application that will serve as a guide experience to provide on-line nose and throat medical diagnoses.

Currently, most of our tests are programmed rather than generated by users working on a graphic user interface. Reliability and efficiency are, at this stage, the most important goals we wish to reach: reliability has been strongly tested by placing the prototypes in extremely long sessions and variable environments (hardware devices, drivers, networks, multiple sessions, etc.); the efficiency study has two major branches: communication strategies impact and system efficiency.

The impact of the communication strategies has been tested by evaluating different data networks: LAN (Local Area Networks) 10 -100 Mbps, Frame Relay 2 Mbps and WLAN (Wireless Local Area Networks) 11 Mbps. At JPEG quality 0.2 and RTP packet size of 1024 Bytes, 14 fps (frames per second) are reached in LAN at 10/100 Mbps or FR at 2 Mbps, whilst only 13 are reached in WLAN. With a JPEG quality of 0.7, 11 fps are reached in LAN at 10/100 Mbps or FR at 2 Mbps, whilst only 9 are reached in WLAN.

System efficiency has been tested by evaluating the impact of the computational load applied to the video streaming: technically, the ‘Effect’ load (see Figure 4). The tests have been performed using different processors. The results are shown in Figure 5. As the computational load of the effects increases (X-axis) the number of frames per second 'fps' that the system is able to produce decreases (Y-axis). The relationship is not lineal due to the different proportions that exist between the computation load of each effect and the actual load imposed by the JMF API and the tested application prototype.

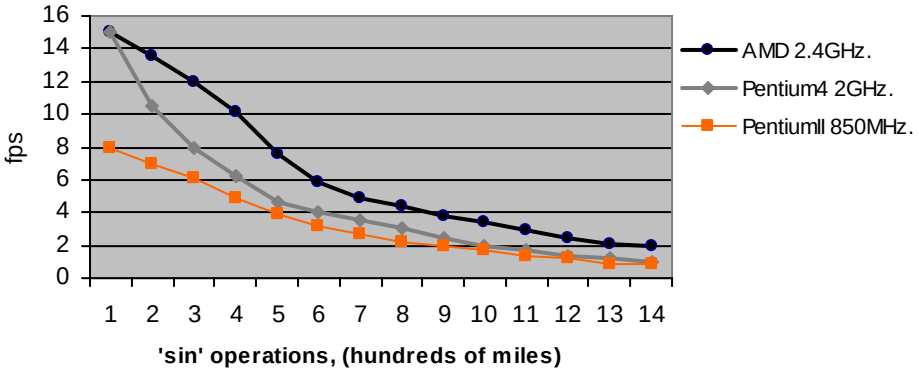

Fig. 5 System performance results

Conclusions

Merging video streaming with real-time vectorial information into unique RTP bitmap streaming facilitates concurrent visualization and the implementation of cooperative applications. Our framework will make it possible to implement applications that are suitable for different requirements: cooperative e-learning, security control systems, cooperative medical diagnoses, intelligence sharing, etc.

The framework architecture presented provides a complete, tested API based on JMF. It is reliable, scalable, requires no direct maintenance, can be easily configured to work using different communication strategies and provides a high degree of interoperability based on its DAML+OIL semantic layer.

Jumping from the prototype scope to commercial applications will require testing the API functionality applied to different environments: networks, computers, devices, users, GUI's, etc. This type of testing involves both technical strategies and psychological focusing in order to provide an efficient, suitable and reliable cooperative interface for specific users. The most important future objectives to be carried out by our framework are the following: specific application implementations, incorporation of well-designed cooperative graphic user interfaces, RTP multicast communication approaches and the improvement of the semantic capabilities by incorporating a specific top-level logic layer

References

1. Fischer, F., Bruhn J., Gräsel, C., Mandl, H.: Fostering collaborative knowledge construction with visualization tools, Learning and Instruction, Vol. 12, Issue 2, (2002) 213-232
2. Bu-Sung, L., Chai Kiat, Y., Ing Yann, S., Keok Kee, L., Wei, S.: Design and Implementation of a Java-based Meeting Space over Internet, Multimedia Tools and Applications, (2003) 179-195
3. http://www-nrg.ee.lbl.gov/vic/
4. http://isabel.dit.upm.es/
5. Zhang, Z., Zongkai, L., Yuchai, G.: The development tool of the application cooperation software, Sixth International Conference on Computer Supported Cooperative Work in Design, (2001) 140–142

6. Gang, Z., Jiati, D.: Cooperative product design process modeling, Sixth International Conference on Computer Supported Cooperative Work in Design, (2001) 236–242
7. Yuchu, T., Dongming, L.: A flexible multimedia cooperative product design platform framework, 7th International Conference on Computer Supported Cooperative Work in Design, (2002) 198-204
8. Weiyu, W., Peng, P., Hua, L., Wei, Y., Dongmei, L.: A framework and implementation techniques for cooperative architectural engineering design systems, The Sixth International Conference on Computer Supported Cooperative Work in Design, (2001) 218–222
9. Lopez, P., Skarmeta, A.: ANTS framework for cooperative work environments, Computer, Vol. 36, Issue: 3, (2003) 56–62
10. Xiao-Ping, L., Xiao-Yun, L., Zheng-Feng, H., Xiao-Rong, T.: Research on cooperative template design, The Sixth International Conference on Computer Supported Cooperative Work in Design, (2001) 52–55
11. Pengcheng, Z., Renhou, L.: Computer Supported Cooperative Work in Design, The Sixth International Conference on Design and implementation of an intelligent cooperative design system, (2001) 198–202
12. http://java.sun.com/products/java-media/jmf/2.1.1/
13. http://java.sun.com/products/java- media/jmf/2.1.1/apidocs
14. Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V. (Audio-video Transport working group): RTP: A Transport Protocol for Real-Time Applications, RFC 1889, (1996)
15. Calvert, K.: TCP/IP sockets in Java: Practical guide for programmers, Morgan Kaufmann, 2001, ISBN: 1558606858
16. Lavian, T., Wang, P., Durairaj, R., Hoang, D., Travostino, F.: Edge device multi-unicasting for video streaming, 10th International Conference on Telecommunications (ICT)., Vol. 2 , (2003) 1441-1447
17. http://www.daml.org/
18. http://www.xml.com/pub/a/2002/01/30/DAMLOIL
19. Maedche, A., Staab, S.: Measuring Similarity between Ontologies, Proc. 13th European Conference Knowledge Enginnering and Knowledge Management (EKAW), LNAI 2473, Springer-Verlag, (2002)