TEC2011-25995 EventVideo (2012-2014)

Strategies for Object Segmentation, Detection and Tracking in Complex Environments for Event Detection in Video Surveillance and Monitoring

D1.2v1

DIVA DOCUMENTATION

Video Processing and Understanding Lab

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Supported by

AUTHOR LIST

Juan C. San Miguel Juancarlos.Sanmiguel@uam.es

CHANGE LOG

Version	Data	Editor	Description
0.0	08-06-2012	Juan C. San Miguel	Initial version
1.0	24-06-2012	José M. Martínez	Version 1

CONTENTS

1. Introduction

In this document, we describe a framework for designing and developing distributed video processing algorithms in which the video feed can be obtained from various sources available in the Escuela Politécnica Superior of the Universidad Autónoma de Madrid (e.g., live cameras or video repositories). This framework is hereafter called DiVA (Distributed Video Analysis). An earlier version of this framework was previously published in 2008 [1].

The DiVA framework establishes a distributed environment for acquiring multiple video sources, communicating with different processing algorithms, applying sequential or parallel processing schemes, visualizing partial results and formalizing the use of contextual information in the analysis process whilst allowing to real time operation. Both the framework and the available applications have been developed and implemented by the Video Processing and Understanding Lab in the Escuela Politécnica Superior of the Universidad Autónoma de Madrid.

1.1. Software requirements

Some external libraries have been included in DiVA for providing basic functionalities such as TCP/IP connection, low-level artificial vision and file management. Currently, DiVA uses the following:

- **Microsoft Fundation Classes (MFC)** 6.0. This package is used for network connections and multithread management.
- **Intel® Open Source Computer Vision Library** version 2.0 (OpenCV). DiVA framework has been developed on top of OpenCV library that simplifies the management and creation of computer vision applications. However, some OpenCV structures have been modified and new characteristics have been included.

Due to the restrictions of the libraries used, DiVA only works on Windows OS.

1.2. Document structure

This document contains the following chapters:

- Chapter 1: Introduction to this document
- Chapter 2: Overview of the DiVA framework

- Chapter 3: Describes the physical part of the DiVA framework (i.e., the network and technologies used)
- Chapter 4: Presents all the subsystems that compose the DiVA framework for acquiring, processing and storing data.
- Chapter 5: Defines the general process for analyzing a specific domain and shows some examples of working applications.
- Chapter 6: Finish this document with some conclusions and future work.

2. DiVA Framework overview

2.1. Design criteria

The design of the DiVA framework is based on the following criteria:

- **Scalability**: it has to be flexible for including additional modules such as processing algorithms or database applications. Moreover, it has to easily support the use of new video sources (USB, IP, FireWire…) or video protocols (MPEG2, h264).
- **Efficiency:** it has to perform its operations without significantly increasing the overall computational cost. Hence, complex operations have to be avoided. In addition, it has to allow an efficient distributed processing.
- **Generality**: all its functionalities have to be developed using the simplest tools and operations (if possible). Hence, it is not recommended to include any application dependent software or code.
- **Failure tolerance:** it has to include mechanisms for detecting and correcting failures (from the processing algorithms and the main modules of the framework) during runtime execution. Thus, supervision tools have to be developed for verifying the correct behavior of the DiVA system.

2.2. Main characteristics

The DiVA framework has the following main characteristics:

- Distributed environment for research, prototyping and deployment of visual analysis systems with support for multi-camera and semantic information.
- Modular and multi-thread design for processing at frame level.
- Asynchronous operation mode based on a client-server model.
- Dynamic workflow composition (sequential or parallel connection of processing algorithms) and update (on-line scalability) based on semantic information.

This framework can be abstracted in two levels (physical and logical) which are described in the following chapters.

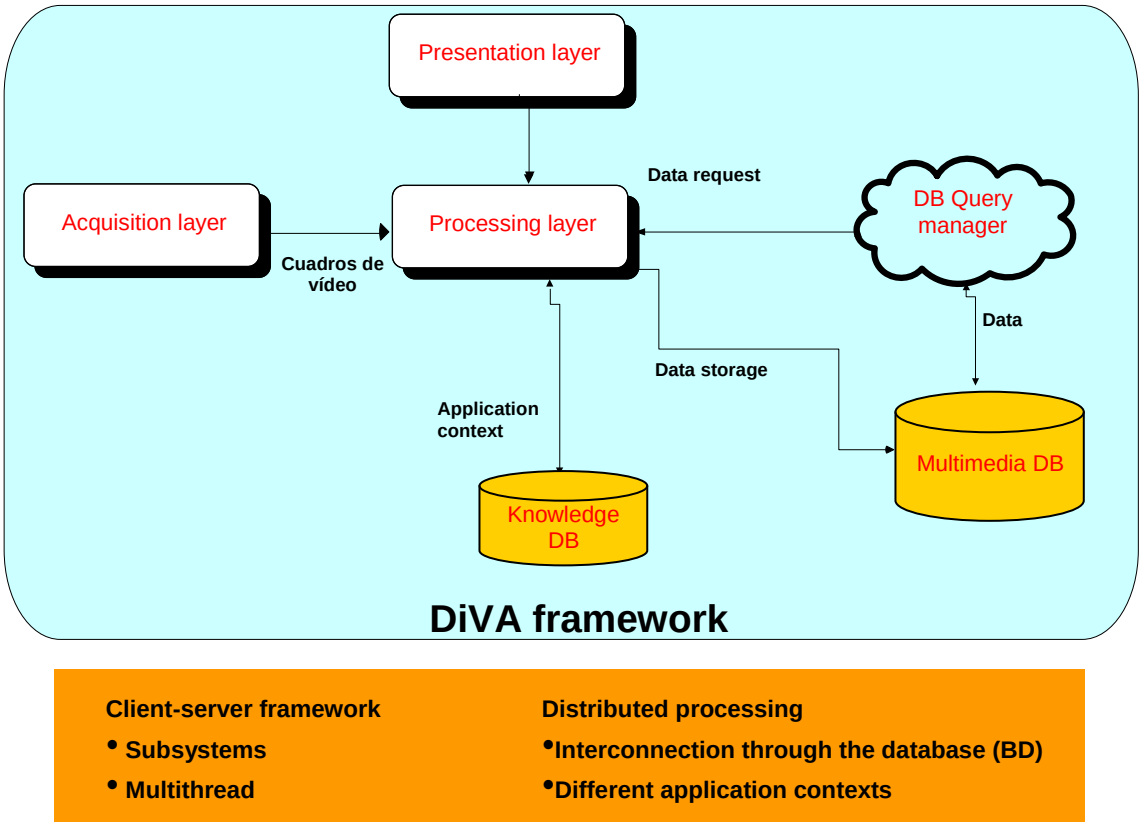

Figure 1. Global view of the DiVA framework

3. Physical part

3.1. Structure

The physical part (see Figure 1) is composed of the required hardware: the cameras and a cluster of standard computers (PCs) connected through a fast Ethernet network. To cope with bandwidth restrictions and to allow operation at real-time, the framework architecture is composed of two networks. The critical framework modules are a set of rack-mounted PCs interconnected by a dedicated Gigabit Ethernet (core network). The other framework modules (mainly processing ones) are distributed in a 100BaseT Ethernet network around the core network. Different types of cameras are plugged either to an acquisition card on a PC or directly to the network for IP cameras. The processing modules are used for video acquisition, algorithm execution and data storage. The main advantage of this architecture is its flexibility. Future needs in computing power can be addressed by simply adding PCs (or replacing the oldest with more powerful ones) in the cluster.

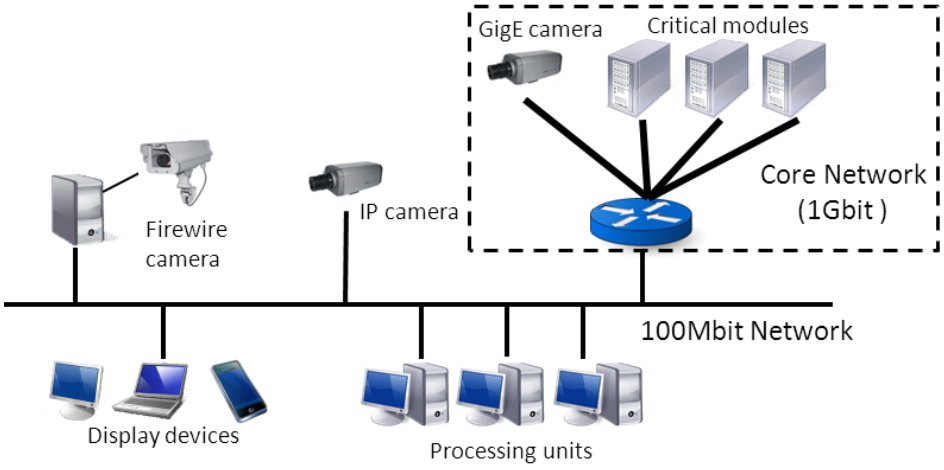

Figure 2. Physical description of the proposed framework.

3.2. Employed technologies

In Table 1, we can observe the technologies employed for the network in DiVA following an OSI model. They have been chosen due to their versatility and their ease of use.

OSI model	Technology
Transport layer	TCP
Network layer	IP
Link layer	Ethernet
Physical layer	Twisted pair

Table 1. DiVA technologies for communication

4. Logical part

The logical part is composed by three independent layers designed in a modular way with a specific role (see **Figure 3**). The different modules of each layer can be distributed in several ways allowing flexible configuration. The communication is based on a client-server model; the flow control is realized through a TCP-based network. Data buffering between modules is supported at both sides for avoiding network delay problems. The system also supports the addition and removal of modules at operation time. Depending on application requirements, layers can be combined into one single component with the required functionality.

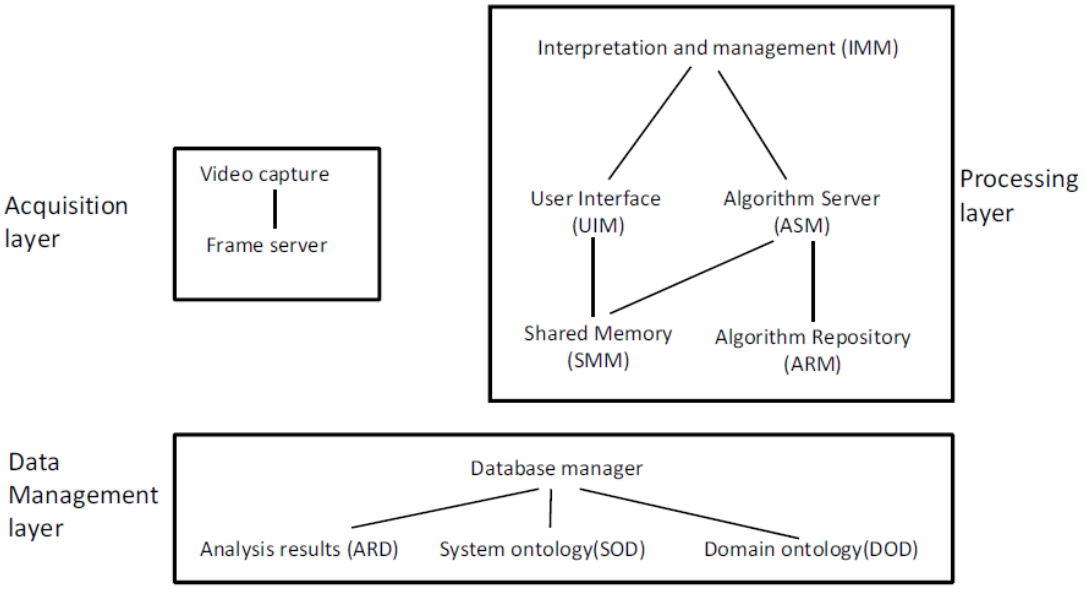

Figure 3. Logical description of the DiVA framework

4.1. Interconnection between layers

For performing the interconnection between the different subsystems available in the DiVA framework, we use TCP/IP sockets using standard Windows libraries available through the MFC Windows library. The communication protocol has been simplified as much as possible. However, this simplification does not impose constraints for extending the communication protocol with, for example, authentication mechanisms and others.

Each subsystem contains a client and a server. The former allows the connection to other DiVA subsystems (through their corresponding servers) and the latter is responsible for providing access to the data contained by itself (e.g., video frames in the acquisition layer) to the entire DiVA framework.

Before initializing a connection for starting the data exchange, the client application perform the following steps:

1. Connect to the derived server using a predefined port.
2. Request ID by the client for the data exchange
3. Send of an ID by the server
4. The data exchange begins. For each client connected to the server, an independent thread is created to serve such client (hence, many clients can be simultaneously connected to the server).

4.2. Acquisition layer

This layer acquires the video from multiple video feeds and distributes it frame-by-frame to the whole framework using a client-server model. For performance issues, the captured data is sent to a storage module in the processing layer (Shared Memory Module). Video frames are currently exchanged using baseline JPEG (ISO/IEC 10918-1) or uncompressed format. A time stamp is attached to each frame at grabbing time and is used in the processing stage (e.g., tracking algorithms). Due to its modular design, the framework can easily support the addition of new camera connection protocols by developing the corresponding video capture interfaces. Currently it handles IP, IEEE1394, GigE and USB protocols, as well as input via video files.

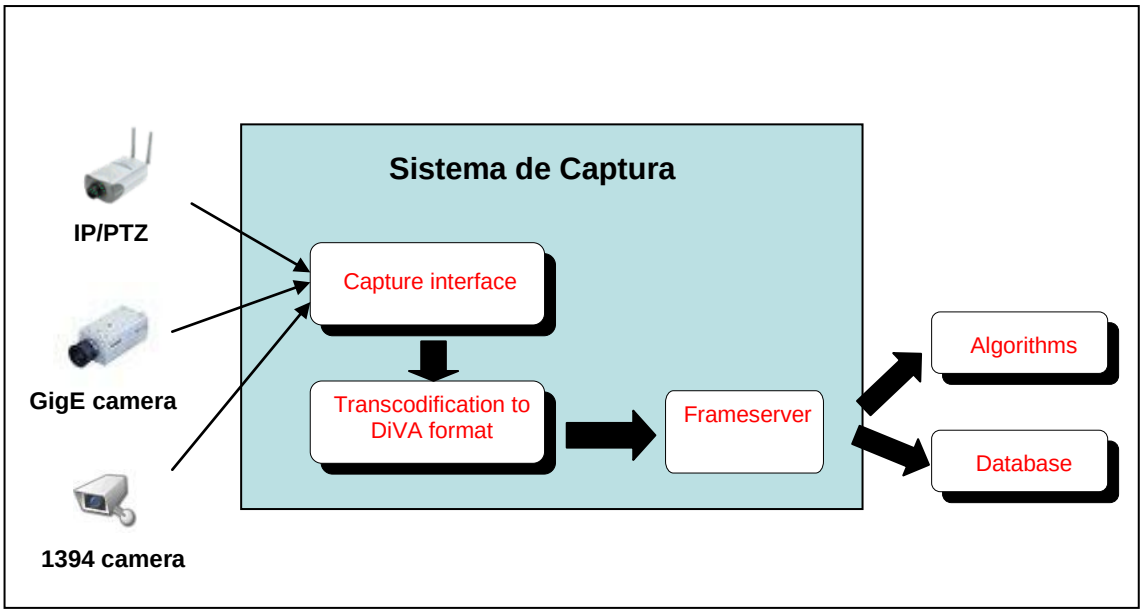

Figure 4. Capture subsystem of the DiVA framework

4.3. Data Management layer

This layer stores and distributes non-visual information required for analysis purposes or the metadata obtained by the processing layer. A database manager is included to control the use of such information. This layer is composed of three database sub-systems:

- The Domain Ontology Database (DOD) provides the information of the modeled application domains. Currently, it contains the domain knowledge description [2].
- The System Ontology Database (SOD) manages the description of the available analysis tools. Currently, it is based on a recent knowledge description [2].
- The Analysis Results Database (ARD) stores the metadata generated by the processing modules making them available for further analysis[1]. Hence, it allows the exchange of the obtained results between processing modules in a distributed configuration.

Moreover, two applications for managing the queries and requests of data by the processing algorithms are included. The structure and dataflow of this layer is depicted in **Figure 5**.

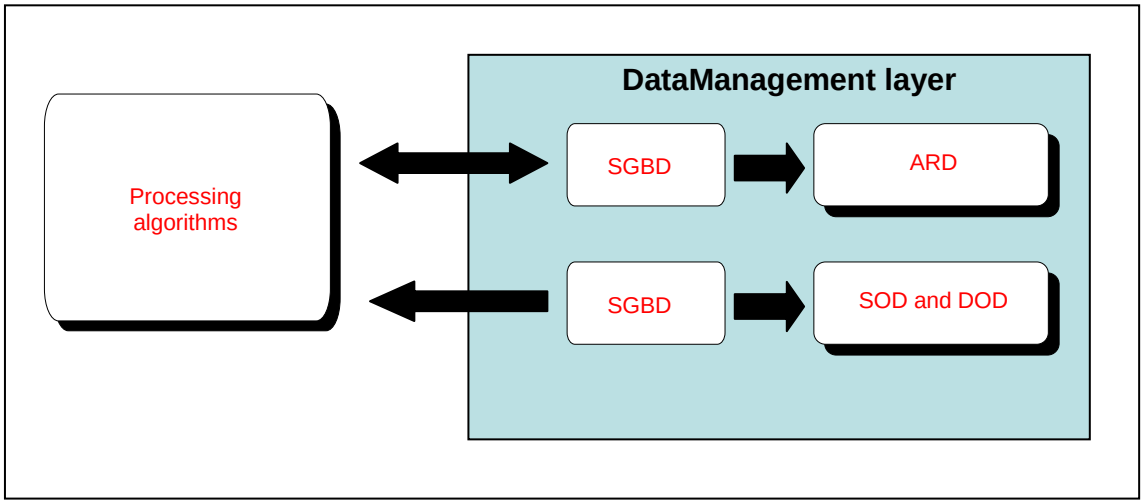

Figure 5. Data Management layer

[1] This database sub-system can be extended for developing query-based applications.

4.3.1. DataServer

The ARD database (also called *DataServer*) is in charge of storing the results (partial or final) generated by the processing algorithms. Moreover, it also includes information about all the active modules of the DiVA system.

The DataServer has been implemented using standard MySQL language. Its design has been done following relational database scheme. **Figure 6** depicts the main elements included in the data and their relations.

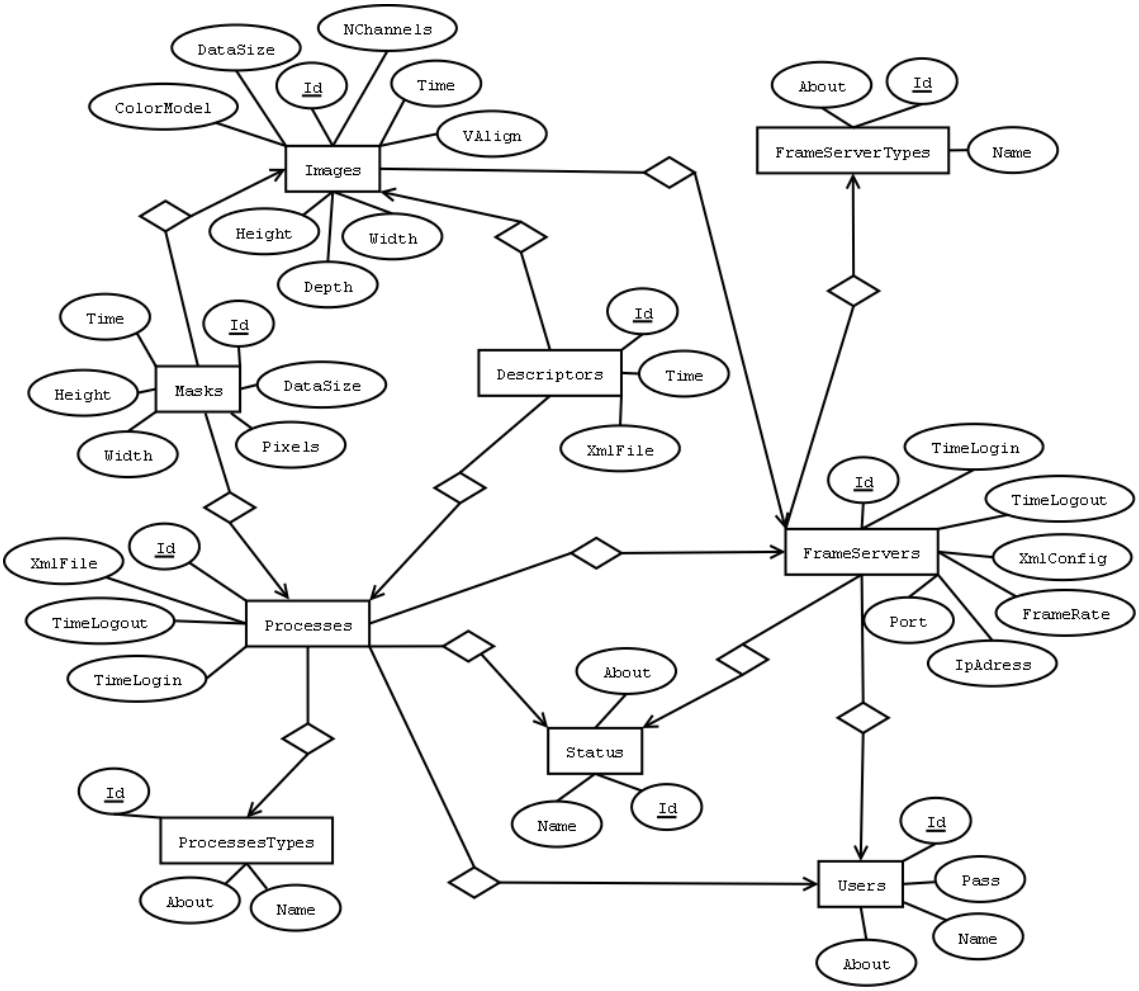

Figure 6. Class structure of the ARD

For allowing queries, a management application has been included using a multi-threaded server. It allows to access to all data as well as to perform maintenance operations.

4.3.2. ContextServer

This database comprises the SOD and ARD databases. It allows to provide a context or domain of application for each algorithm. It also includes a server and a manager for communicating with the processing algorithms.

Currently, the *ContextServer* contains two types of information for representing domain and system knowledge. The former defines the physical space where a real event occurs and which can be observed by one or several cameras. It includes the scene objects, their interactions (events) and the scene context. The latter describes the visual agent that analyzes the video content (in our case, the algorithms of the DiVA system) including the analysis capabilities of the system, the possible responses to the detected events and the system status. Figure 7 depicts the entities that compose such database to describe the context information.

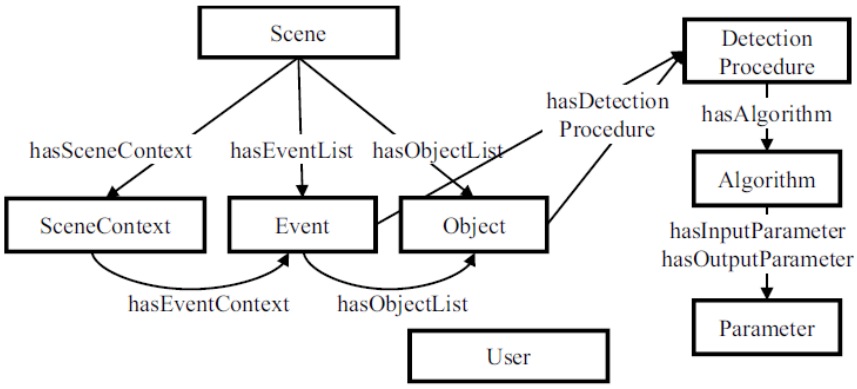

Figure 7. Domain and system knowledge included in the ContextServer

4.4. Processing layer

In DiVA, a processing module is a component responsible for some particular task not related to the other layers (e.g. video analysis module, player module). The modules run concurrently and asynchronously allowing to develop distributed applications: typically each module will run on its own processor, but this is not mandatory. Moreover, some module templates have been created for easy algorithm development and integration in the framework.

This layer communicates with the Acquisition and the Data Management layers to request data (e.g., video frames, previous analysis results) and to store the obtained results. This data exchange allows the distribution of processing capabilities. Moreover, this layer includes several analysis algorithms that can be selected and combined for solving specific analysis problems.

An example of such distribution is depicted in Figure 8. It can be observed that the most computational demanding tasks (e.g., motion detection) could be performed in more powerful processing units and metadata analysis could be done in other units with less computational power.

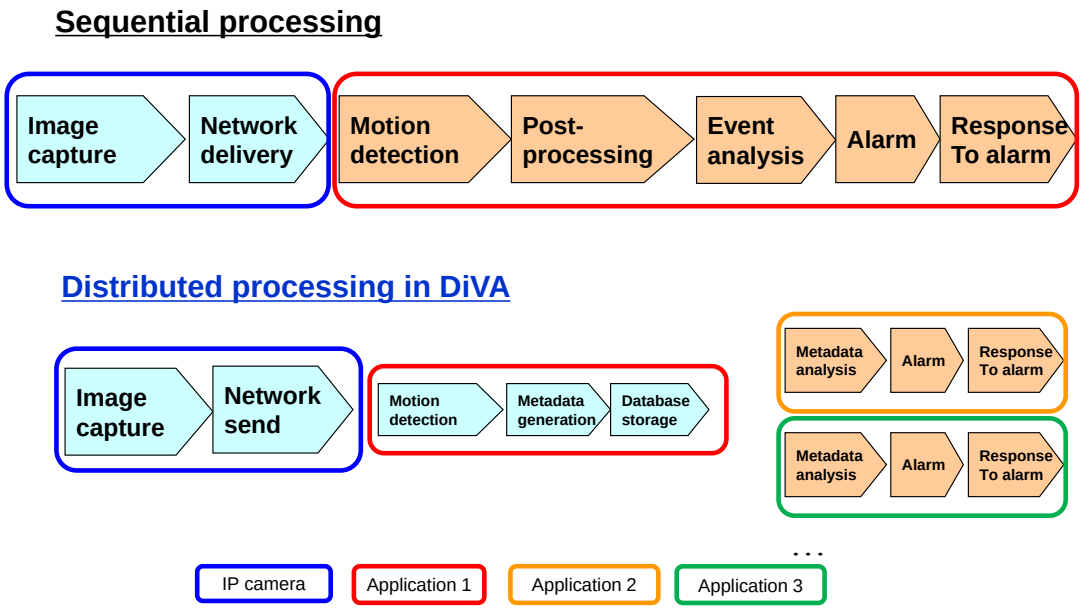

Figure 8. Distributed processing scheme within DiVA framework

Currently, this framework performs two tasks: ontology interpretation and video analysis, making use of the following modules:

- The Interpretation and Management Module (IMM) processes the knowledge encoded in the domain and system descriptions, then combines it with user preferences and finally requests the execution (to the Algorithm Server Module).
- The Algorithm Server Module (ASM) provides the processing capabilities to the entire framework. It makes the visual analysis tools usable through a server.
- The Algorithm Repository Module (ARM) indexes the available analysis tools and stores their compiled versions in order to provide the processing capabilities.
- The User Interface Module (UIM) interacts with the content consumer (e.g., human user, software agent) to get its input and to show the obtained results.

4.4.1. Template for Processing algorithms

For creating a new component in the DiVA framework, an encapsulated component has been developed with the objective of providing a template for all the algorithms that are to be developed. Its objective is to encapsulate all the communication and management operations that have to be performance.

This template has been developed in C++ having the following built-in functionalities:

- Data request&capture:
    - It includes a client for connecting to the Acquisition Layer in order to choose the appropriate *FrameServer* that provides the video frames. This capture could be performed in sequential or continuous mode (in the latter case, using a frame buffer).
    - It includes a client for connecting to the *ContextServer* in order to provide an application context to the processing algorithm.
    - It includes a client for connecting to the *DataServer* in order to provide an access to the data storage in such database (results of other algorithms).
- Data Display: it incorporates a standard method for displaying data in the screen.
- Image processing: it includes specific functions and locations to add the operations to perform the video analysis.

In summary, the development of applications or algorithms in the DiVA framework requires to include the processing operations in the particular functions that have the provided template. Hence, connecting with other DiVA components is done in an automatic and transparent way allowing to the designer to focus on the algorithm itself and not on its integration in DiVA.

5. DiVA for specific domain analysis

For the analysis of video content from a specific domain, the following sequence of operations is performed:

1. Initialization. The UIM gets the necessary data for the analysis (e.g., domain to analyze, user preferences) and configures the IMM. Then, the IMM requests the semantic information of the domain and the analysis capabilities to the DOD and SOD modules.

2. Semantic-based workflow composition:

   a. The IMM requests to the ASM the analysis tools available for the selected domain by using the data indexed in the ARM. Then, the instances of the existing visual analysis tools are created and properly linked to the domain knowledge.

   b. The IMM inspects the semantic system information to calculate the necessary resources (parameters) and to allocate memory for them in the SMM. Instances of the parameters are created and linked with the Algorithm instances.

   c. The IMM interprets the system and domain semantics to select the required visual analysis tools among the available ones for domain analysis. Then, this information is sent to the ARM (via the ASM) for the creation of resources. An example of such interpretation is provided in this work[3]

3. Analysis. Finally, the IMM begins the sequential processing of the analysis workflow via execution requests to the ASM. The analysis is performed until the video file has been finished or the system is turn off (for live on-line video analysis). Results obtained by each execution are stored in the SMM that made them available for further analysis or display purposes. During run-time operation, the update of the analysis workflow (addition or removal) is performed.

6. Conclusions and future work

This document has described a distributed framework for video analysis that allows flexible and dynamic configuration at run-time. It provides support for acquiring, transmitting, processing and storing data (video content and metadata). It is structured into different layers in charge of each type of tasks in the framework. Due to low computation management cost, it operates at real-time over standard computers

Additionally, it defines a flexible environment to develop video-based applications via easy component integration. Hence, the effort of the developer is focused on the image processing operations instead on the integration of the algorithm.

For the future, other extensions and improvements will be made on the global system, like integration of a compressed video analysis path or adaptation of the system to work under Linux (using POSIX threads for multitasking scheduling). In particular, a communication restriction has been observed when dealing with high-resolution images as this data requires high bandwidth. Moreover, DiVA currently operates using uncompressed frames and, therefore, a high transmission delay is introduced in this situation. Hence, a future improvement would be to focus on developing efficient transmission technics for images and video (using codification approaches) as well as adapting existing ones (JPEG, MJPEG…).

References

[1] J. C. SanMiguel, J. Bescós, J. M. Martínez, and A. Garcia. Diva: A distributed video analysis framework applied to video-surveillance systems. In Proc. of IEEE Int. Workshop on Image Analysis for Multimedia Interactive Services, pages 207-210, 7-9 May 2008

[2] J. C. SanMiguel, J. M. Martínez, and A. García. An ontology for event detection and its application in surveillance video. In Proc. of the IEEE Int. Conf. on Advanced Video and Signal based Surveillance, pages 220-225, Genoa (Italy), 2-4 September 2009

[3] J.C. SanMiguel, José M. Martínez, "A semantic-guided and self-configurable framework for video analysis", Machine Vision and Applications, Springer, ISSN 0932-8092 (Print) 1432-1769 (Online, December 2011) (Digital Object Identifier: 10.1007/s00138-011-0397-x)

Glosary

ASM Algorithm Server Module

ARM Algorithm Repository Module

ARD Analysis Results Database

DiVA Distributed Video Analysis

DOD Domain Ontology Database

IMM Interpretation and Management Module

IP Internet Protocol

JPEG Joint Picture Experts Group

MFC Microsoft Foundation Classes

OpenCV Open Computer Vision library

SOD System Ontology Database

TCP Transport Control Protocol

USB Universal Serial Bus

UIM User Interface Module

Appendix

A. Example of usage

Here, we provide an example of usage of the DiVA framework for foreground segmentation algorithms showing some screen shots of the previously presented modules.

After installing the DiVA package version 1.3. Download the demo and unzip the package. For running the components of the demo, you have to go to the directory in which it has been installed and execute the desired BAT file.

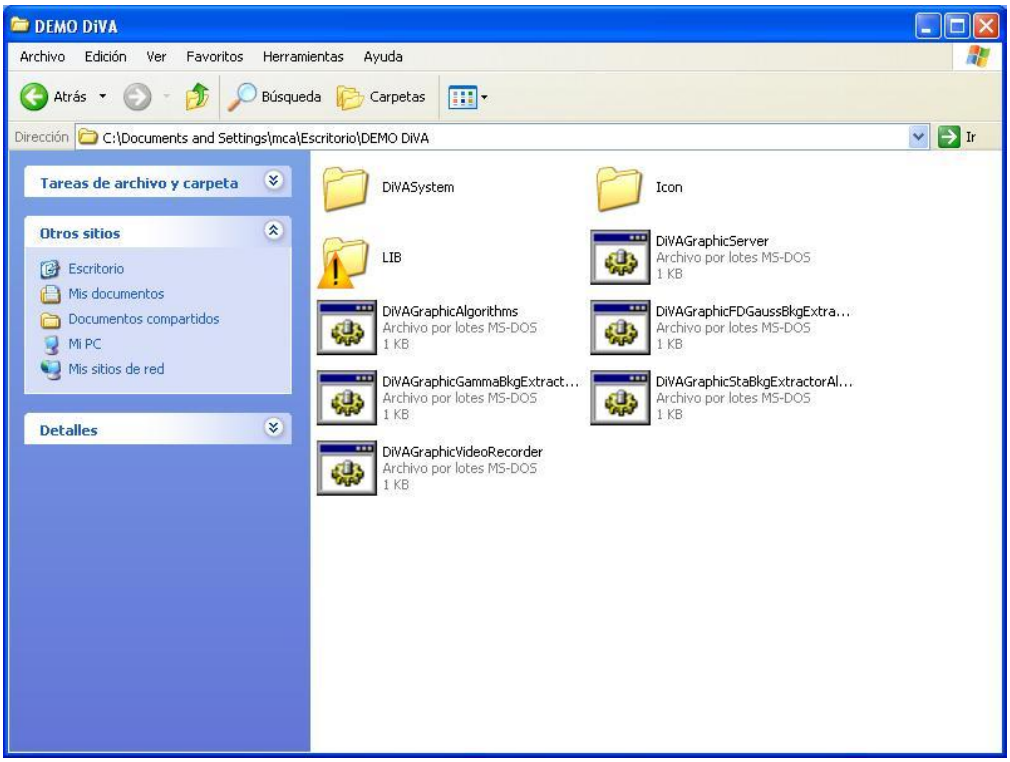

Fig. A.1: Directory for DiVA demo

As it can observed, there are 6 BAT files: DiVAGraphicServer (graphical tool for executing the FrameServers of the acquisition layer), DiVAGraphicVideoRecorder (graphical tool for recording videos from a FrameServer), foreground segmentation algorithms based on background subtraction (DiVAGammaBkgExtractor, DiVAGraphicStatBkgExtractor and DiVAGraphicFDGaussBkgExtractor) and performance assessment tools DiVAGraphicAlgorithms). In the following subsections, we briefly describe how to use them.

A.1 FrameServer manager

After executing the BAT file of the DiVAGraphicServer application, we obtain the following window:

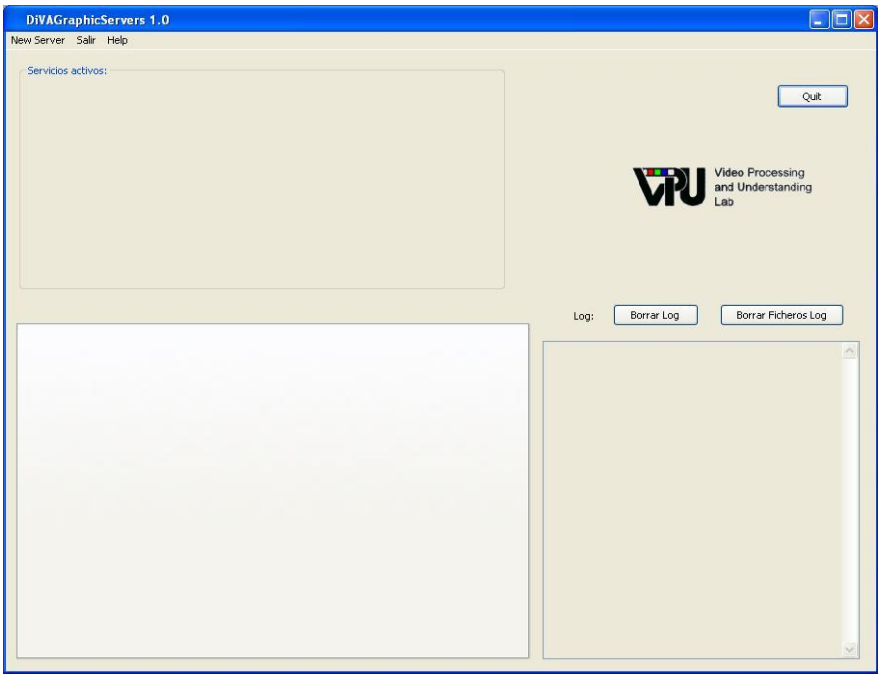

Fig. A.2: Aspect of the graphical tool for FrameServer management

By using the menu on top-left area, we can start a FrameServer:

1. Click on the button "New Server". A new menu appears for configuring each type of available FrameServers.

2. Choose the IP one and new appears a new window.

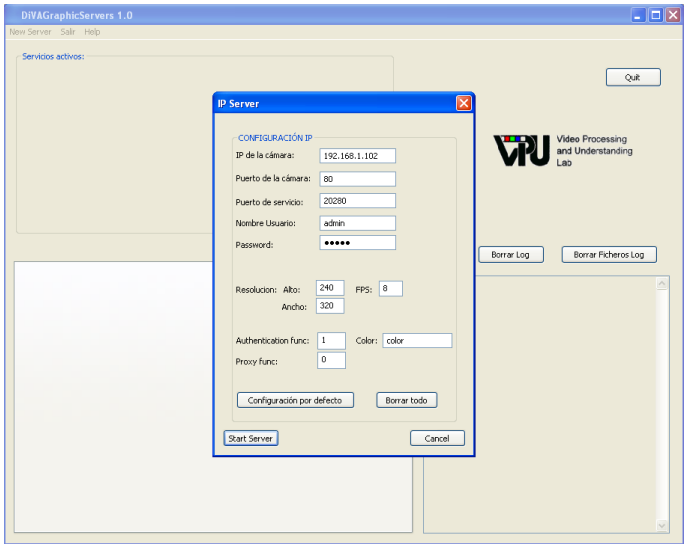

Fig. A.3: Window for configuring an IP frameserver

3. This window contains the configuration parameters of the IP FrameServer.

4. Click on the button "Start Server" and the main window will change (if everything is correct) as depicted in Fig. A.4

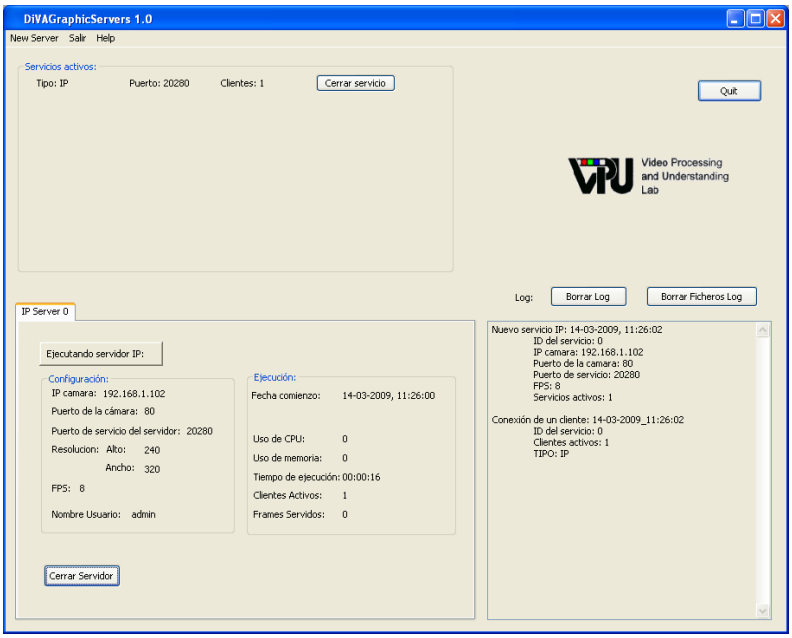

Fig A.4: FrameServer manager with IP server activated

Finally, the IP frameserver is working and capturing frames from the selected IP camera. After this moment, any application of the DiVA framework can start using the captured data.

A.2 Processing algorithms

Among the available processing algorithms in the DiVA demo, there is an application for executing up to four foreground segmentation algorithms in order to compare their results.

To start this application, please execute the corresponding BAT file and new window appears similar to Fig. A.5. On the top-left part, the user is able to select which algorithms wants to execute and to which FrameServer is going to be connected. Then, click on the "start" button to create such algorithm. For changing the parameters of the algorithm, a tab appears on the bottom-left part of the windows for such interaction. Finally, the results are shown on the sub-windows included in the application.

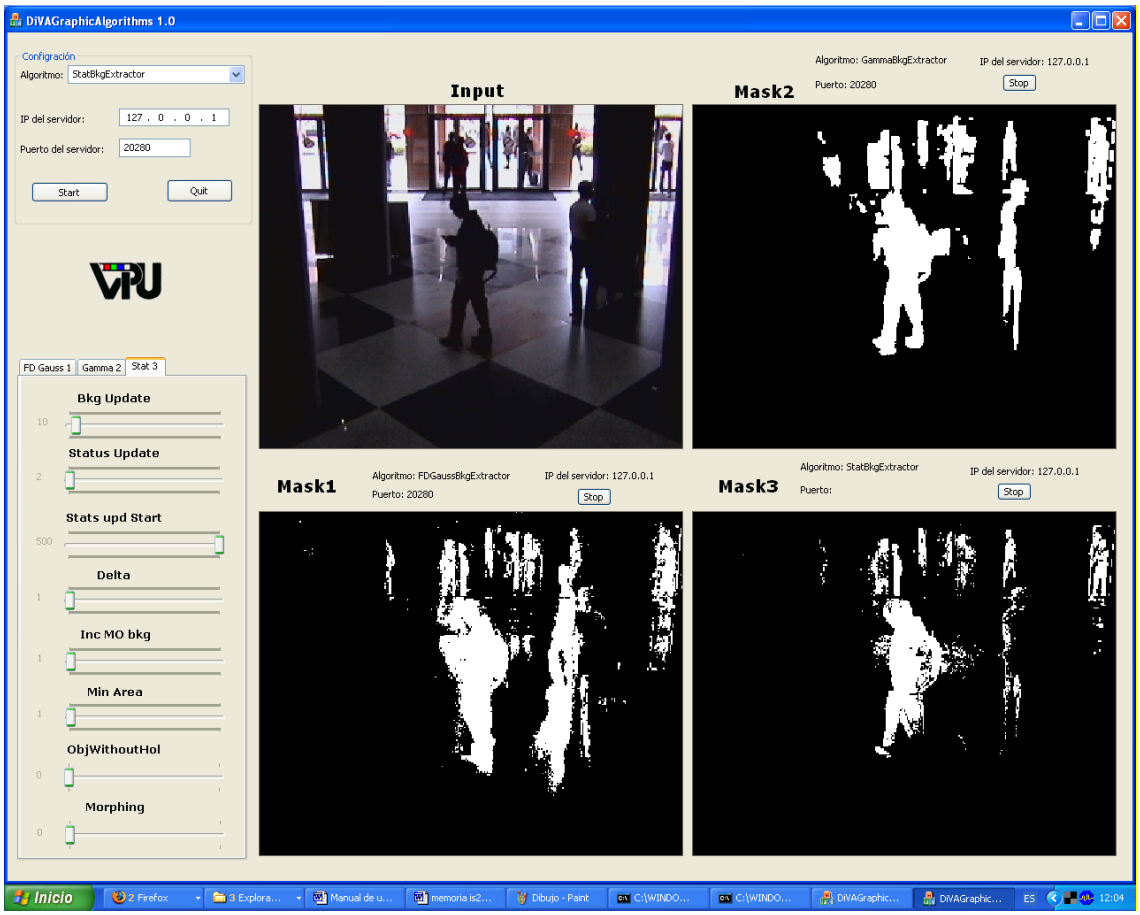

Fig A.5: 1.1.1 DiVAGraphicAlgorithms application