TOWARDS A SCALABLE ESPINA FOR NEUROSCIENCE DATA ANALYSIS

ABSTRACT

ESPINA is an image segmentation tool designed to analyse microscopy images in order to identify neuronal structures and to produce 3D models of these structures. This tool allows to display three-dimensional volumes using auto-stereoscopic monitors. It was initially designed for workstations, but when data volume management or its processing complexity makes impossible the implementation of the new tools on these computers, it is necessary to resort to computing servers that delimit response times or by means of scalable solutions and algorithmic optimizations. This paper analyses the migration of this tool from the original implementation to a scalable solution and describes the experience achieved during the development of the workstation version. The proposed alternative is a distributed version of the tool that delegate heavy-computational processes to a cluster, improving the performance of the system in a master/slave architecture.

INTRODUCTION

ESPINA is an image segmentation tool designed to analyse microscopy images in order to identify neuronal structures and to produce 3D models of these structures [1]. It has been developed for the Cajal Blue Brain Project (CBBP) [2]. This tool provides a semi-automatic supervised operation mode and an automatic mode, currently being fully operational. The system is being incrementally extended for incorporating new functionalities not initially implemented, like the stereoscopic viewer available.

Experts work is based in the observation of neural tissues for establishing new hypothesis about the behaviour of brain microcircuits [3]. Main features of this work are variability and uncertainty, since new priorities are fixed depending on the available technologies and the scientific hypothesis that will guide the works. This situation causes that the new supporting computer tools will be developed taking into account these parameters, and must be turned into deliverables after a very short time from the identification of the need.

On the other hand, faced with the initial difficulty for implementing fully automatic image analysis procedures, supervised analysis techniques are applied, granting an efficient and friendly
interaction for minimizing the learning curve of the new available tools. Later, these techniques will be extended, if it is possible, to become fully automated procedures.

In addition to the mentioned problems, we must take into account another one when the data volume management or its processing complexity makes unfeasible the implementation of the new tools on workstations, being necessary to resort to computing servers and to scalable solutions and algorithmic optimizations that delimit response times. The proposed alternative is a distributed version of the tool that delegates heavy-computational processes to a cluster of processing nodes, improving the performance of the system in a master/slave architecture (distributed version mode). Parallel architectures based on clusters have spread very widely over the last decade thanks to their excellent scalability, fault tolerance and flexibility conditions, and above all, to their excellent price/performance ratio [4].

Finally, developed tools are required to grant their diffusion to the widest community of potential users. Therefore, the development of multi-platform tools will be an objective in this context. Additionally, this version is more versatile than the original. Even if the user does not have available a computing cluster, it is possible to run the new version in a normal workstation (with the computational limitations derived from the workstation itself).

The contents of this paper describe the design issues followed for creating the tool ESPINA, the methodology and technologies applied during its development, the restrictions found working in a real-world situation and the analysis made to migrate the current version to a scalable solution. Finally, we will present the main conclusions extracted and the future work planned.

**DESIGN ISSUES**

This Section will describe the objectives guiding the tool design, the methodology followed to accomplish these goals and the main features of the current version.

**Design Methodology**

User-centered design is a design philosophy in which all the efforts made for developing an interface try to optimize how people are, need or wish working [5]. This idea involves the user in an iterative process not only in the analysis and design stages, but also in the test phase working in a real-world environment. Once a prototype had been implemented, it was validated by real users testing its functionality (usability study) and ease of interaction (user studies). Quantitative and qualitative information were collected from a reduced and selective set of expert users, providing invaluable information for the further analysis and design stages of the subsequent cycle. This approach is only feasible to do in a short period of time with a multidisciplinary group like the one involved in the CBBP, composed by neuroscience experts and computer engineers.

Also, we have followed several well established principles based on the Model Human Processor proposed by Card et al. [6], to create an efficient prototype with an intuitive interface [7, 8].

Taking into account the background of the users, the interface must be as user friendly as possible in order to minimize the learning curve of the prototype, avoiding the discouragement new users might experience when faced with a very complex system. In this sense, we have followed the most remarkable usability heuristics proposed by the NN Group [9–11].

The aim of the CBBP development team is to apply an incremental and iterative agile software development methodology. We followed a development methodology similar to *scrum* because the principles proposed in this method were very much like the scenario we faced and fit very well to the user-centered design philosophy [12]. This methodology enables software to be developed rapidly on the basis of small sprints, the result of each representing a further improvement in the tool. In our case, each stage of the work-flow was the result of the corresponding sprint and new improvements will be introduced in the near future.

**User Tasks**

Regarding with end user activities, the following processes have been identified:

- **P1** Data exploration (navigation)
- **P2** Data processing (realignment, filtering, correlation, transformation, segmentation)
- **P3** Data visualization (rendering)
- **P4** Data manipulation (edition)
- **P5** Data management (storage, normalization, labelling, naming, etc.)
- **P6** Work-flow management (specification, control)

Instead of images, we talk about data because of the different nature of the digital samples involved in the work of neuroscience experts: optical microscopy, transmission microscopy, confocal microscopy, electronic microscopy, temporal series, simulations, video animations, symbolic information, etc.

ESPINA has been designed for P1, P2 and P3 processes dealing with FIB/SEM data (Focused Ion Beam combined with Scanning Electron Microscopy) [13]. The available microscope is a Crossbeam Neon40 EsB, from the manufacturer Carl Zeiss NTS GmbH (Oberkochen, Germany). But at present, ESPINA deals with stacks of images captured from confocal and electronic microscopy, although it is foreseeable that there will be no problem to easily expand the type of data managed by the tool.

We are working for integrating P4 in the next version. Related with the P5 and P6 processes, currently ESPINA works with local data sets, although in a near future work will be done with a...
central data repository involved in the data and work-flow management issues.

Figure 1 shows a high-level diagram of the user tasks implemented in ESPINA. User interaction is performed in all current processes implemented (P1-P3). Even when automatic segmentation is performed, users must supply several parameter or accept default values. Synapses are key elements in the functional organization of neuronal circuits and thus, revealing their location, size and the proportion of the two different morphological types is extraordinarily important. ESPINA performs the segmentation and 3D reconstruction of synapses in the cerebral cortex, helping the user to examine large tissue volumes and to interactively validate the results provided by the software. The scale of the available data varies between a magnification of 10,000× and 30,000×. Such variation demands continuous user intervention for setting the parameters of the corresponding scale and explains, together with the complexity of the images involved, the difficulties found for achieving fully automatic segmentation and reconstruction techniques. Currently, there is no fully automatic segmentation technique for the type of images handled.

**Design Goals**

Besides all the general objectives pursued in the implementation of a software tool using agile software development methods [14, 15], in short, the developed tool must accomplish the following specific objectives:

1. The tool must be adapted to the end users methodology, not opposite.
2. Interactive analysis of neuroscience data will be granted.
3. User interface will be as friendliest as possible.
4. The technologies used will allow a wide diffusion of the tool.
5. The user will be able to analyse the information without loosing data dimensionality in the analysis.
6. From the segmented structures found by the analysis and processing engines, the system will extract new features and relevant data from the user point of view.
7. The tool will use both as input and output, standard formats.

**Current System Architecture**

ESPINA current architecture is aimed to execute in a workstation. ESPINA can be decomposed in the following modules (Fig. 2):

- Application Framework (APF): specific and not reusable by other application. It is the ESPINA application logic.
- Image Analysis and Processing module (IAP): basic operations for data filtering, feature extraction, counting procedures, etc.
• Segmentation module (SEG): manages all segmented data.
• I/O Module (IO): implements all input/output operations.

There is another transversal module, the Graphical User Interface (GUI), which gives supports to the previous modules in all tasks involving user interaction. Memory management is delegated to the operating system. This is the main bottleneck of the application. Figure 2 shows the relationships among the above modules.

Figure 3 shows a snapshot of the current user interface, the 3D data visualization windows, the object manager and the data viewer. In the snapshot, the user has segmented three asymmetric synapses (green) and one symmetric synapse (red) in a plane of the 3D FIB/SEM stack.

**Tool Features**

We can enumerate the following list of features present in the current version of ESPINA:

1. The system is robust, allowing the user to complete working sessions without any problem. It is stable, using contrasted technologies among a wide range of users.
2. The user interface is intuitive and friendly for reducing the learning curve and presents contextual information and help.
3. Response time minimization is sought in supervised data analysis procedures for granting tolerable interactivity levels.
4. The system interface (or the client to the system in a client/server architecture) is launched on a personal computer.
5. The system provides some kind of modular architecture, in order to be able to use different algorithms provided by different programmers, to change the image analysis and/or processing behavior.
6. The user is able to focus his attention on regions of interest of the stack or on sets of segmented structures (individually identified) to apply on the selected choice any operation defined in the system. Also, the user is able to interactively define the “unbiased counting brick” and the tool shows its dimensions both in real world units and in relative stack volume units. The set of active structures inside the new frontiers is refreshed activating or moving the limits of the “unbiased counting brick”. The set of structures touching the exclusion borders of the “unbiased counting brick” are visually distinguishable from the active set.
7. Once the structures are segmented, the tool stores behind the segmented data the parameters that allowed their extraction and the user is able to retrieve this information at any time. Also, the user is able to associate visual labels (a combination of text and colours) to the structures found as result of the stack processing. Furthermore, the user is able to define new entities to identify objects not implemented until that moment in the system functionality.
8. The user is able to reconfigure at any moment the data visualization setup for adjusting it to his/her specific needs.
9. The tool has different mouse pointers, which appearance changes depending on the task the user is performing in the tool (e.g. “segmenting” or just “navigating”), in order to inform the user of the possible result of any action performed in a certain moment.
10. The tool simultaneously shows the stack from the selected user point of view and from planes X/Y, X/Z and Y/Z, including the cross-hairs in each view for visual reference. The cross-hairs are always visible with any view or window reconfiguration.
11. The user moves around the stack touching the wheel mouse. The tool is able to apply zoom transformations showing the % value of the rescaling, and independently hides or shows the data and the visual references (e.g. cross-hairs, “unbiased counting brick”).
12. The list of segmented structures includes information about their position inside the stack. Creation or removal of segmented structures is made either from the list of structures or from the visual planes. The user is able to select and copy the set of extracted features. This set is presented following a predetermined subset based on the user criteria. This subset is configurable at any time.
13. The user is able to visualize 3D data choosing between a stereoscopic or a conventional 3D viewer.

The system is portable to different environments (OSs, architectures). Current version has been successfully installed in Windows XP, Vista and 7 (32 bits) and Linux (64 and 32 bits). The implementation supports free distribution policies among scientific community. The reader can find a video showing
a demo of the tool at http://cajalbbp.cesvima.upm.es/espina.

TECHNOLOGIES USED

The following working tools have been used in the developed tool:

- Programming languages: Python, C++.
- Toolkits:
  - ITK, open code graphic library developed by Kitware specialized in image processing and analysis [16].
  - VTK, library for image visualization, also from Kitware [17].
  - Qt, library for graphical interfaces [18].
  - Wrappers that allow rapid prototyping with Python: WrapITK for ITK [19] and PyQt for Qt [20]. VTK supports direct wrapping to Python.

The choice of free distribution tools as development tools is focused on to facilitate in the future the free dissemination of the tool ESPINA on as many systems as possible.

On the other hand, Python is a programming language that fits very well into the development methodology followed [21]. C++ is the most common programming language nowadays in the computer software community and some of the ESPINA dependencies are implemented in this language. The existence of well tested wrappers facilitates the communication between these programming languages and the experience achieved has shown that we have found no problem for mixing them.

Figure 4 shows a diagram of the technologies currently used by the ESPINA tool version oriented to workstations. ESPINA integrates the classes and objects, defined in the sense of Object Oriented Programming (OOP), provided by the toolkits creating new ones when needed.

We will give additional details about the main ESPINA dependencies.
FIGURE 4. Technologies involved in the current ESPINA implementation.

ITK

ITK provides a large catalogue of image filters [16]. Rather than the image processing algorithms actually used by ESPINA, the main reasons to maintain this dependency are:

- Easy management of label maps for representing the segmentation and obtaining statistics from them.
- Wide range of Input/Output image formats.
- Well designed framework to write new filters, easier to use than coding from scratch.
- Future needs might be already implemented.
- Most ITK filters run in parallel, creating so many processes as available cores in the machine.

In general terms ESPINA current version uses these ITK features:

- Read/write “.mha/.mhd” file formats.
- Gaussian blur, binary threshold, connected components, pad/crop image filters.
- Representation of the result segmentation as a label map, providing individual management for the segmented objects.
- Wide range of statistics computed for each segmented object.
- Visual pipeline connection via VTK.

VTK

VTK is a library for 3D computer graphics visualization [17]. It was chosen first because of its integration with ITK, and second because it is focused on medical visualization and high level visualization techniques are ready to use.

In VTK is easy to show stacks of images and also to reconstruct the segmentation either with volume rendering or mesh rendering. Also, as mentioned above, ESPINA is able to perform stereoscopic rendering taking advantage of the facilities provided by VTK. Dresden and RedBlue modes are currently supported. Choosing RedBlue mode, the user can visualize 3D scenarios with a very cheap device like passive glasses. Configuring the tool for Dresden mode, the user obtains a better visualization performance, in terms of resolution and comfort, but more expensive active stereoscopic devices are required. Figure 5 shows a snapshot of the Dresden mode.

We have used auto-stereoscopic SeeReal monitors for displaying three-dimensional information. Two alternative technologies have been tested, active monitors (model C-i) and passive displays (model C-n), but in both cases, users have some fatigue after a period of about 15 minutes of work due to the technology used in these monitors (made in 2007). We will do further experiments using stereoscopic glasses because users have found 3D visualization a very powerful tool for the analysis and reconstruction of synapses and this type of output devices give better and more comfortable visual feedback than auto-stereoscopic displays. In contrast, monitors give more freedom to users, being the reason they were selected for the first version of the tool.

It is expected that this type of stereo rendering will be used by experts in the near future as the 3D visualization technique to inspect the data in their regular work, as they have already transmitted in the tests performed.

Qt

Qt is a library for interface development [18]. QT supplies all the widgets and controls that the user needs in a very friendly interface.

Current ESPINA version manages 271 classes with the following distribution among toolkits and own source code:

- ITK: 32
- VTK: 90
- Qt: 58
- New classes created: 91

It is noticeable the high levels of re-usability achieved, allowing faster development times.
TABLE 1. Object distribution in function of toolkits and ESPINA modules implemented: absolute value / %. First row shows the number of lines of Python code (loc) per module.

<table>
<thead>
<tr>
<th></th>
<th>IAP</th>
<th>APF</th>
<th>IO</th>
<th>SEG</th>
</tr>
</thead>
<tbody>
<tr>
<td>loc</td>
<td>1324 / 24.45%</td>
<td>1400 / 25.85%</td>
<td>604 / 11.15%</td>
<td>2087 / 38.54%</td>
</tr>
<tr>
<td>ITK</td>
<td>33 / 8.99%</td>
<td>0 / 0%</td>
<td>4 / 7.70%</td>
<td>43 / 8.81%</td>
</tr>
<tr>
<td>VTK</td>
<td>26 / 7.08%</td>
<td>57 / 25%</td>
<td>14 / 26.92%</td>
<td>110 / 22.55%</td>
</tr>
<tr>
<td>Qt</td>
<td>308 / 83.93%</td>
<td>171 / 75%</td>
<td>34 / 65.38 %</td>
<td>335 / 68.64%</td>
</tr>
</tbody>
</table>

Considering the number of lines of code, ESPINA has required around 5,415 lines of Python code. Table 1 summarizes the distribution of the objects programmed, showing absolute values and percentages of the total number.

The only setback suffered in the use of these toolkits has been that the development team has found some serious problems when porting to 64 bits Windows versions, regarding some of the third-party libraries used in the solution. In particular, with the WrapITK library.

MIGRATION PROPOSAL TO THE SCALABLE IMPLEMENTATION

The main limitation of the current version of ESPINA is that it does not grant interactivity with stacks composed by 200 images or more at full resolution (approx. 400 MB or greater) managed in a PC with 6GB of RAM. Users must reduce the working resolution to one third of the maximum given by the Cross-Beam in order to keep interactivity at good levels, and although it is enough for some specific tasks, for example synapse counting, some other tasks require the maximum resolution available. Therefore, this problem requires that a solution must be sought for the current implementation.

To meet all the requirements set by users in terms of data size and levels of interactivity, and taking into account the experience achieved with the workstation version, the new design will have the following resource and performance requirements:

- The tool (in the workstation version mode) will require a minimum of a 4GB RAM memory with a recommended configuration of 12 GB RAM Memory.
- The tool (in the distributed version mode) will require an image processing cluster of 64GB RAM memory and 4 core processors with 1Gb to 10Gb bandwidth connection.
- The tool will support up to 1GB image series in the workstation version mode, with the restricted set of processing features.
- The tool will support up to 8GB image series in the distributed version mode, with the full set of processing features.

One possibility would be to develop our own framework based on the prototype of the application ESPINA. However, due to the existence of similar frameworks, either specific (SSCRETT+NeuroTrace) [22] or general purpose (ParaView) [23], we will consider them as a alternative of development.

Cajal Blue Brain Project Framework

In the case of building our own framework, we have considered two options: one based on light clients and other on hard client.

Light Client  In a thin client-based framework, each application would run entirely on the server through a connection with access to the graphics server (using VNC-Xvnc or SSH with X-forwarding). Thus, the framework would only offer a few tools to copy on the server the input data and retrieve the results once the execution ends.

Hard Client  In the version with hard clients, each application would decompose into two parts: the client and the server side. The client will execute the user interface and will communicate with the server side using a socket connection. The framework should provide the architecture on which to support each application, providing connections to the server and access and management of the available resources.

Existing Solutions

Visualization and management of large volumes of data is common to many areas of scientific research. Therefore, we can consider several frameworks that are subject to similar characteristics given in our project.

ParaView  ParaView is a free cross-platform framework developed by Kitware, for visualizing large volumes of data that supports distributed computing models. It is based on VTK and uses an architecture based on plug-ins.

SSCRETT + NeuroTrace  SSCRETT is an application designed for manual labeling of axons by examining 2D slices. NeuroTrace is designed for interactive segmentation using interactive 3D visualizations of the data volumes. SSCRETT is being developed by Kitware and initially it will not be developed under any open source license. However, it seems that Neurotrace will be available as an open source version within less than a year.

New Proposal

The proposed solution, taking into account available resources and the requirements given, would be to create a new version of ESPINA based on hard clients built with ParaView. Thus,
New features can be added using the plug-in system on which is built ParaView or creating new client applications if deemed advantageous.

ParaView is written in C++ using the Qt libraries and VTK, allowing its use in major systems: Linux, Windows and Mac. It also supports MPI and its graphics libraries, the image analysis libraries and the ParaView framework itself have been tested in many projects and in a variety of supercomputers, including the Blue Gene, which is one of the supercomputers involved in our project. There are simulations with ParaView where rendering uses around 8 billion polygons per second (super computer Red Rose with 5000 nodes), far beyond our needs. In addition, to be an open source version, its distribution is not limited by software licenses.

The main types of standard formats are supported by ParaView, but it is also possible to add new types of formats using the schema of readers/writers.

The new version will include a tracing mechanism which can be used to recover the work in case of crash or connection fail.

Implementing the segmentation algorithm on the server, response time is minimized by the parallel data processing, being limited by the network bandwidth between the client and server.

With ParaView, both plug-ins and new clients can be developed in the future so that they work in combination with existing products.

Since both the computational and memory load are transferred to server node, the user interface can run on any personal computer.

**Implementation Progress**

To assess the feasibility of the new proposal, the ParaView version with support for Mesa off-screen has been installed in an IBM HS22 Blade.

The values of the standalone version for the workstation (labelled as “Workstation” in the Figures) are obtained on an Intel i7 with 6GB of RAM. The client-server configuration has been tested running the server with MPI installed on one HS22 Blade node with 8 cores, and the client on the same machine that the standalone version. Client-server communication is established using a 100 Mbps Ethernet network.

We haven’t used any of the processing nodes available in the cluster because the HS22 Blade with the server is able to manage by itself all the workload demanded in the tests. If in the future the workload surpasses the processing capacity of the server node, then the processing nodes will be gradually incorporated to the resources involved in the interactive working session considering the required workload.

Using the mentioned configuration we estimate the scalability behavior of the ParaView client-server version. This setup can be considered a subset of the available resources in the final tar-
get architecture, Magerit [24]. This is a cluster consisting of 1219 computer nodes, of which 1036 nodes are eServer BladeCenter JS20, each one of them contains two PPC 2.2 GHz (8.8 GFlops) processors with 4 GB RAM, 168 nodes are eServer BladeCenter JS21 with four PPC 2.3 GHz (9.2 GFlops) processors with 8 GB RAM and the rest are 15 nodes eServer BladeCenter HS22 with eight Intel Xeon 2.5 GHz (10.2 GFlops) processors with 96 GB RAM, implying 2,864 CPUs and 6.97 TB RAM. All the nodes operate independently and with the same software configuration. This computer has a local storage capacity of about 192 TB, provided by 256 disks of 750 GB, using a distributed and fault tolerant file system (GPFS).

The tests performed have been to run different operations that require an interactive response in a typical work session. Images belong to brain tissue extracted from rats. Typical slice thickness is 20 nm, and in this case, image resolution is 2048 x 1536 pixels, at a resolution of 3.7 nm per pixel. Stacks may vary in the interval of 200 and 750 images, covering a brain volume around 175 µm³ for the higher number of images. Figures 7 and 8 show some results of the tests performed to validate the migration.

Figure 7 presents a typical filtering operation followed by the rendering of the result. In the workstation version, both stages are balanced, but in our current cluster, there is no chance to perform local rendering in a GPU like in the case of the workstation and this value fires. It could be possible to use a cluster with GPUs in addition if it is required, but rendering needs in ESPINA are not very demanding. We have measured also the overhead introduced by the workload management in the cluster. For all the tests performed, this value is negligible compared with the net workload. All time values are given in seconds.

When the relationship between processing time and rendering time is decanted to the former, as seen in the benchmark proposed in Figure 8, there is no doubt about the advantages of the version of the cluster and the scalability of this type of image processing algorithms: median filter, threshold segmentation and contour extraction.

In short, we were able to run on the server algorithms requiring greater memory capacity, our bottleneck with the current version, fulfilling the performance requirements outlined above. When the setup server has only one processing node, performance is worse since the absence of more cores in the workstation version was mitigated by the presence of a GPU. But when the server can distribute the workload across multiple nodes, server results always improve the performance of the workstation version taking advantage of the good scalability of the operations involved. This property will provide lower response times when using Magerit because much more processing nodes will be available than those used in the experiments.

CONCLUSIONS AND FUTURE WORK
In this paper we have described our experiences migrating ESPINA, a tool for counting synapses in FIB/SEM images, from a design oriented to workstation architectures to a scalable implementation on a cluster. Tools like the one described in this paper are key elements for finding new discoveries in the field of neuroscience. Experts need new tools to analyze the vast amount of data that electron microscopy technologies are able to provide. They also need new ways of analyzing the data. For example, using display devices that allow them to interact better with the data, such as the stereo visualization included in ESPINA.

Migration to the new version is made without any setback due to the development methodology followed and the technologies chosen to carry out the implementation. The planning is being implemented without delay. The workload associated with the development of the original version of the tool is estimated in 10 man-month. It is expected that the new version of ESPINA will be running after 6 man-month thanks to the chosen path.

The scalability of the implementation is guaranteed by the tests. We have reproduced the typical operations in a work session with ESPINA demonstrating the feasibility of this solution.
The expansion of the tool to include new features will be made very simple thanks to the plug-ins framework available with ParaView.

In addition to achieving a scalable system, major improvements included in the new version will be:

- The system will record working sessions, results and sequences of actions that produce these results, and will be able to retrieve these recordings.
- The tool will support both EM and light microscopy (confocal) image processing and manipulation (identified as P4).
- New structures will be segmented, like neurotransmitter vesicles (EM), myelin (EM), mitochondria (EM), dendrite (confocal) or spines (confocal). The inclusion of new structures or elements is open for future reviews.
- The tool will interact with the future Image Data Repository or any other tool that provides the catalogue of the microscopy images within the project, instead of working with local data.

ACKNOWLEDGMENT

We thank I. Alonso and J. Vila for their technical assistance, and the members of the Laboratorio Cajal de Circuitos Corticales at the UPM for their comments and suggestions during the development of the tool. This work was supported by grants from the Spanish Ministry of Education, Science and Innovation (TIN2010-21289 and the Cajal Blue Brain Project, spanish partner of the Blue Brain Project initiative from EPFL).

REFERENCES


