The SCRIBO Module of the Olena Platform: a Free Software Framework for Document Image Analysis

Guillaume Lazzara, Roland Levillain, Thierry Géraud, Yann Jacquelet, Julien Marquegnies, Arthur Crépin-Leblond
EPITA Research and Development Laboratory (LRDE)
14-16, rue Voltaire, FR-94276 Le Kremlin-Bicêtre, France
E-mail: *firstname.lastname*@lrde.epita.fr

Abstract—**Electronic documents are being more and more usable thanks to better and more affordable network, storage and computational facilities. But in order to benefit from computer-aided document management, paper documents must be digitized and analyzed. This task may be challenging at several levels. Data may be of multiple types thus requiring different adapted processing chains. The tools to be developed should also take into account the needs and knowledge of users, ranging from a simple graphical application to a complete programming framework. Finally, the data sets to process may be large. In this paper, we expose a set of features that a Document Image Analysis framework should provide to handle the previous issues. In particular, a good strategy to address both flexibility and efficiency issues is the Generic Programming (GP) paradigm. These ideas are implemented as an open source module, SCRIBO, built on top of Olena, a generic and efficient image processing platform. Our solution features services such as preprocessing filters, text detection, page segmentation and document reconstruction (as XML, PDF or HTML documents). This framework, composed of reusable software components, can be used to create full-fledged graphical applications, small utilities, or processing chains to be integrated into third-party projects.**

Keywords-**Document Image Analysis, Software Design, Reusability, Generic Programming, Free Software**

I. Introduction

Today's information tends to be more and more electronic, stored and processed as digital data using computers and networks. Digital processing allows users of document-related software tools to benefit from the properties of computer-aided document processing: (semi-)automated computations, efficient processing of (possibly large) data sets, adaptable and configurable methods, etc. However, there is still a considerable amount of non-digital (paper) documents in use. One of the challenges of Document Image Analysis (DIA) is to acquire, process and integrate them just like born-digital documents.

Digitized data includes many different categories of documents: books, magazines, newspapers, invoices, maps, camera pictures and videos, etc. Each kind of document has its own peculiarities: some of them are only composed of text, others mix text with images, tables, graphical data; documents may exhibit a structure (e.g. articles) or not (e.g. maps or pictures); etc. Each of these categories requires specific processing chains to produce the desired results. Furthermore, the corresponding applications may have to provide efficient implementations of these workflows so as to be able to handle the amount of input data.

The exponential growth of the digital world requires versatile software tools to both acquire and process non-digital data, in particular in the field of paper documents. Such tools are expected to handle large data sets (i.e. numerous and/or large digital images) and address many use cases depending both on the nature of the data (invoices, articles, etc.) and the operations to be processed (image enhancing, feature extraction, etc.). In addition, to meet the complexity of the tasks to perform and the knowledge of the various users, these tools should offer different interfaces including Graphical User Interfaces (GUIs), interactive execution environments, Command Line Interfaces (CLIs), Application Program Interfaces (APIs) of software libraries, etc.

DIA tools usually propose one or several workflows starting from a digitized document (and sometimes featuring this acquisition step), and delivering one or several results (image(s), structured or unstructured data, etc.). Each workflow constitutes a toolchain that can be reused on similar documents. The data which needs to be extracted may differ from the use case. For instance, in automatic document indexing, retrieving only the text may be sufficient. For Entreprise Content Management (ECM) and paper archiving, where users should be given the possibility to operate on the whole initial document, preserving every element (text, pictures, tables, etc.) as well as their spatial organization is essential. A DIA framework should therefore support the creation and use of many toolchains (as well as variants of these) to match the variety of use cases and kinds of documents.

In this paper, we try to circumscribe the properties of a DIA software tool able to handle this variety using a single extensible framework, as presented in Section II. Section III proposes an organization of such a framework, its components, and the featured user interfaces. The ideas expressed in this paper have been implemented in a DIA framework, which is a dedicated module of a generic and efficient Free Software image processing platform, Olena [2]. Section IV presents the contributions of the SCRIBO module and compares our approach with similar projects. In addition to the presentation of our framework, we also show some research results in DIA

This work has been conducted in the context of the SCRIBO project [1] of the Free Software Thematic Group, part of the "System@tic Paris-Région" Cluster (France). This project is partially funded by the French Government, its economic development agencies, and by the Paris-Région institutions.

produced by our solution in Section V. Section VI summarizes the goals and status of the project.

II. Designing a Modern DIA Framework

A. Features and Properties

We believe that a reusable DIA framework should feature two main design features: flexibility and efficiency.

1) Flexibility: To handle the variety of DIA inputs while addressing as many uses cases as possible, a software framework should be constructed from *reusable building blocks*. Each block would provide an elementary operation (binarization, deskew method, etc.) and processing chains would be set up by plugging the output of a block to the input of another block. This idea of building flexible software tools is one of the design choices of the Gamera project [3].

2) Efficiency: Processing speed may be a major concern in some use cases, when a DIA toolchain is expected to handle a large quantity of data per day, for instance in an industrial context (e.g., analyzing batches of invoices from clients).

Based on our experience, a good strategy to address both flexibility and efficiency issues is to use the Generic Programming (GP) approach. Generic Programming is a way to design algorithms and data structures so that they are not tied to a set of fixed types (e.g. an image with only binary or 8-bit gray-level data), but applicable on inputs defined by free parameters (e.g. an image with values of type V). GP is a programming paradigm resolved at compile-time, which means that it does not induce run-time penalties *per se*, and is therefore a good choice as far as performance is concerned.

In addition to these design traits, the design framework should also take the following elements into account:

3) Multiple Interfaces: Another facet of the diversity of DIA use cases is expressed through the diversity of *users*. Some of them will need a very simple graphical interface with little or no configuration to handle a pre-built task; some will want to easily arrange existing toolchains or building blocks using an interactive environment or CLI; others may have to design new building blocks by using the framework's API to create new algorithms.

4) Easy to Integrate: DIA is often an element of a larger IT workflow, for instance ECM, which is concerned with the storage, management and processing of an organization's data.

B. A Module over a Library

During the past ten years we have been designing and implementing a generic and efficient Image Processing (IP) platform, Olena [2]. Lately we have added a module dedicated to DIA to Olena, following the ideas listed above. In the remainder of this paper, we present the components, applications and capabilities of this software tool, designed to be flexible and efficient.

To address these design constraints, we propose to build the DIA framework as Free Software based on a generic C++ library. C++ is indeed a widespread, well-known and well-supported language supporting the GP style. There are several free and proprietary compilers producing efficient code for most architectures in use. This approach has already been chosen by other projects taking an interest in DIA like Qgar [4] to produce reusable software tools. Alternative strategies to handling the diversity of documents include generative methods as demonstrated by the DMOS project, where a grammar formalism is used to describe relationships between the elements of a structured document [5].

On top of this C++ library, the framework shall provide multi-level User Interfaces (UIs) like scripting language interfaces (Python, Ruby, Perl, etc.), command line programs (to be used from a Unix shell), graphical interfaces to set up and run processing chains and analyze their results. Extensibility should be preserved: an algorithm or a data structure added to the generic C++ library should be easily made available to all UIs.

C. Free Software and Reproducible Research

The whole module has been developed as an open source software. It is freely available [6], providing a collection of tools to design DIA applications. It will be offered as separated packages and some toolchains will be available as standalone applications.

For the purpose of testing and evaluation, online demonstrations of the module are also available [7]. One of them performs the detection of text in pictures; another one handles the task of document binarization. A third demonstration generates HTML and PDF documents from an image of document, producing results such as the ones shown in Fig. 1.

Distributing a package as Free Software has many advantages: it helps to spread the project through the Internet; it makes integration and extension easier; and in a research context, it enables reproducible research, i.e., the possibility to reproduce, study and compare published results [8], [9].

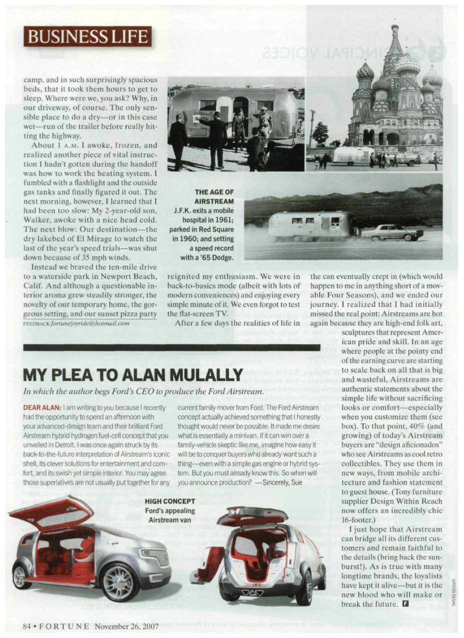
III. Framework Insight

This section presents the SCRIBO module and the architecture of the Olena project. The core of Olena is a generic and efficient C++ IP library called Milena. On top of this general purpose IP framework, Olena aims at providing components addressing more specific needs of various application domains. The SCRIBO module, dedicated to DIA, is the first of these components. Olena is a Free Software project developed following open source principles.

A. Generic Image Processing Foundations

We only sketch the Milena library in this section, as this is not the main topic of this paper. Milena [10] contains generic data types (including images, points, point sets, values, neighborhoods, structuring elements, weighted windows, functions and accumulators), and generic IP algorithms. By "generic", we mean these elements have a general definition that does not limit their actual reusability. Hence an algorithm can be written once and reused with many input types, as long as their combination is valid.

Milena uses a multi-paradigm approach: its is based on a programming technique that we have developed, which



BUSINESS LIFE

camp, and in such surprisingly spacious beds, that it took them hours to get to sleep. Where were we, you ask? Why, in our driveway, of course. The only sensible place to do a dry—or in this case wet—run of the trailer before really hitting the highway.

About 1 A.M. I awoke, frozen, and realized another piece of vital instruction I hadn't gotten during the handoff was how to work the heating system. I fumbled with a flashlight and the outside gas tanks and finally figured it out. The next morning, however, I learned that I had been too slow: My 2-year-old son, Walker, awoke with a nice head cold. The next blow: Our destination—the dry lakebed of El Mirage to watch the last of the year's speed trials—was shut down because of 35 mph winds.

Instead we braved the ten-mile drive to a waterside park in Newport Beach, Calif. And although a questionable interior aroma grew steadily stronger, the novelty of our temporary home, the gorgeous setting, and our sunset pizza party reignited my enthusiasm. We were in back-to-basics mode (albeit with lots of modern conveniences) and enjoying every simple minute of it. We even forgot to test the flat-screen TV.

After a few days the realities of life in the can eventually crept in (which would happen to me in anything short of a movable Four Seasons), and we ended our journey. I realized that I had initially missed the real point: Airstreams are hot again because they are high-end folk art, sculptures that represent American pride and skill. In an age where people at the pointy end of the earning curve are starting to scale back on all that is big and wasteful, Airstreams are authentic statements about the simple life without sacrificing looks or comfort—especially when you customize them (see box). To that point, 40% (and growing) of today's Airstream buyers are "design aficionados" who see Airstreams as cool retro collectibles. They use them in new ways, from mobile architecture and fashion statement to guest house. (Tony furniture supplier Design Within Reach now offers an incredibly chic 16-footer.)

I just hope that Airstream can bridge all its different customers and remain faithful to the details (bring back the sunburst!). As is true with many longtime brands, the loyalists have kept it alive—but it is the new blood who will make or break the future. F

FEEDBACK *fortunejoyride@hotmail.com*

THE AGE OF AIRSTREAM J.F.K. exits a mobile hospital in 1961; parked in Red Square in 1960; and setting a speed record with a '65 Dodge.

MY PLEA TO ALAN MULALLY

In which the author begs Ford's CEO to produce the Ford Airstream.

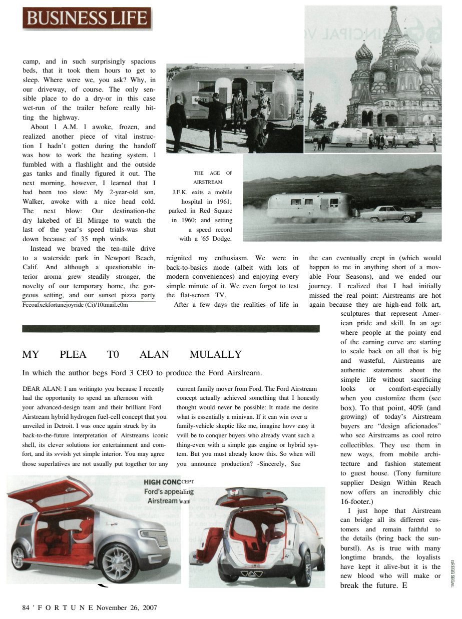
DEAR ALAN: I am writing to you because I recently had the opportunity to spend an afternoon with your advanced-design team and their brilliant Ford Airstream hybrid hydrogen fuel-cell concept that you unveiled in Detroit. I was once again struck by its back-to-the-future interpretation of Airstream's iconic shell, its clever solutions for entertainment and comfort, and its swish yet simple interior. You may agree those superlatives are not usually put together for any current family mover from Ford. The Ford Airstream concept actually achieved something that I honestly thought would never be possible: It made me desire what is essentially a minivan. If it can win over a family-vehicle skeptic like me, imagine how easy it will be to conquer buyers who already want such a thing—even with a simple gas engine or hybrid system. But you must already know this. So when will you announce production? —Sincerely, Sue

HIGH CONCEPT Ford's appealing Airstream van

GREGG SEGAL

84 • FORTUNE November 26, 2007

(a) Original document image.



BUSINESS LIFE

camp, and in such surprisingly spacious beds, that it took them hours to get to sleep. Where were we, you ask? Why, in our driveway, of course. The only sensible place to do a dry-or in this case wet-run of the trailer before really hitting the highway.

About 1 A.M. l awoke, frozen, and realized another piece of vital instruction I hadn't gotten during the handoff was how to work the heating system. l fumbled with a flashlight and the outside gas tanks and finally figured it out. The next morning, however, I learned that I had been too slow: My 2-year-old son, Walker, awoke with a nice head cold. The next blow: Our destination-the dry lakebed of El Mirage to watch the last of the year's speed trials-was shut down because of 35 mph winds.

Instead we braved the ten-mile drive to a waterside park in Newport Beach, Calif. And although a questionable interior aroma grew steadily stronger, the novelty of our temporary home, the gorgeous setting, and our sunset pizza party reignited my enthusiasm. We were in back-to-basics mode (albeit with lots of modern conveniences) and enjoying every simple minute of it. We even forgot to test the flat-screen TV.

After a few days the realities of life in the can eventually crept in (which would happen to me in anything short of a movable Four Seasons), and we ended our journey. I realized that I had initially missed the real point: Airstreams are hot again because they are high-end folk art, sculptures that represent American pride and skill. In an age where people at the pointy end of the earning curve are starting to scale back on all that is big and wasteful, Airstreams are authentic statements about the simple life without sacrificing looks or comfort-especially when you customize them (see box). To that point, 40% (and growing) of today's Airstream buyers are "design aficionados" who see Airstreams as cool retro collectibles. They use them in new ways, from mobile architecture and fashion statement to guest house. (Tony furniture supplier Design Within Reach now offers an incredibly chic 16-footer.)

I just hope that Airstream can bridge all its different customers and remain faithful to the details (bring back the sunburstl). As is true with many longtime brands, the loyalists have kept it alive-but it is the new blood who will make or break the future. E

Feeoafxckfortunejoyride (Ci)/10tmail.c0m

THE AGE OF AIRSTREAM J.F.K. exits a mobile hospital in 1961; parked in Red Square in 1960; and setting a speed record with a '65 Dodge.

MY PLEA T0 ALAN MULALLY

In which the author begs Ford 3 CEO to produce the Ford Airslrearn.

DEAR ALAN: I am writingto you because I recently had the opportunity to spend an afternoon with your advanced-design team and their brilliant Ford Airstream hybrid hydrogen fuel-cell concept that you unveiled in Detroit. I was once again struck by its back-to-the-future interpretation of Airstreams iconic shell, its clever solutions ior entertainment and comfort, and its svvish yet simple interior. You may agree those superlatives are not usually put together tor any current family mover from Ford. The Ford Airstream concept actually achieved something that I honestly thought would never be possible: It made me desire what is essentially a minivan. If it can win over a family-vehicle skeptic like me, imagine hovv easy it vvill be to conquer buyers who already vvant such a thing-even with a simple gas engine or hybrid system. But you must already know this. So when will you announce production? -Sincerely, Sue

HIGH CONCCEPT Ford's appealing Airstream van

GREGG SEGAL

84 ' F O R T U N E November 26, 2007

(b) Document rebuilt as a PDF file.

Fig. 1: Document segmentation and reconstruction.

mixes the benefits of Generic Programming (GP) and Object-Oriented Programming (OOP) [11]. The library is written in C++ as the language is well-known, widely available, standardized, portable, mature, and versatile. The language is statically-typed; C++ compilers generates efficient code and enable error detection at compile time. Milena has also been designed to be simple (just running algorithms requires only a basic knowledge of C), user-friendly (it features automatic memory management, run-time messages about unexpected behaviors, etc.), and it proposes a syntax close to usual IP notations and idioms.

B. DIA Framework Workflow

Such a platform is dedicated to several kinds of users. *Simple users* who are expected to use the framework without knowing how it works; *Designers* who have the knowledge to write and assemble new algorithms; and *Integrators* who need DIA features and specific toolchains in their projects. To embrace this variety of users, we propose a multilayered platform with features accessible through e.g. a GUI, an interactive shell or scripts.

This platform is composed of a library of algorithms dedicated to DIA. It provides routines for preprocessing steps (including binarization, show-through removal, deskew, denoising), text lines identification, non-text objects retrieval (tables, pictures, separators, etc.), line reconstruction (rebuilding separators and tables), Optical Character Recognition (OCR) integration, document reconstruction (converting a document to XML, PDF and/or HTML outputs). All these components form the building blocks from which DIA toolchains can be assembled. Dedicated data structures are also provided to handle intermediate results. They help to keep the code readable and ensure reasonable performance. Memory is also automatically managed through these structures so as to make the work of developers easier.

We also provide pre-built generic toolchains for document reconstruction, document text extraction, and text detection in pictures. They all share most of their components. Some steps are performed differently, like text detection. The document reconstruction toolchain depicted in Fig. 2 accepts color, gray-level and binary images. Scanned document may be skewed, bear large black strips at the document's edges, display show-through effects, etc. Sometimes, the document itself is of very poor quality and needs to be cleaned up. Our platform proposes a set of preprocessing algorithms to prepare the document for analysis. The provided basic toolchain includes foreground extraction (see Fig. 5) to correct show-through effects, as well as deskew, binarization and denoising algorithms. Note that we work on binary images since our approach is based on Connected Components (CCs). We propose a set of object filters to remove unwanted object according to a specific criterion, which can be easily extended. In addition, delimiters (lines, whitespaces and tab-stops) can be detected (see Fig. 6) thanks to object alignments and morphological algorithms. The platform proposes several variants of these algorithms more or less sensitive to the quality of the delimiters. Delimiter information is useful for the next step, text extraction. In some cases, for document reconstruction for instance, we may need to extract non-text elements such as pictures, drawing, tables, etc. Our platform provides tools to locate and extract such objects separately.

DIA results can be saved to an XML file. We have chosen

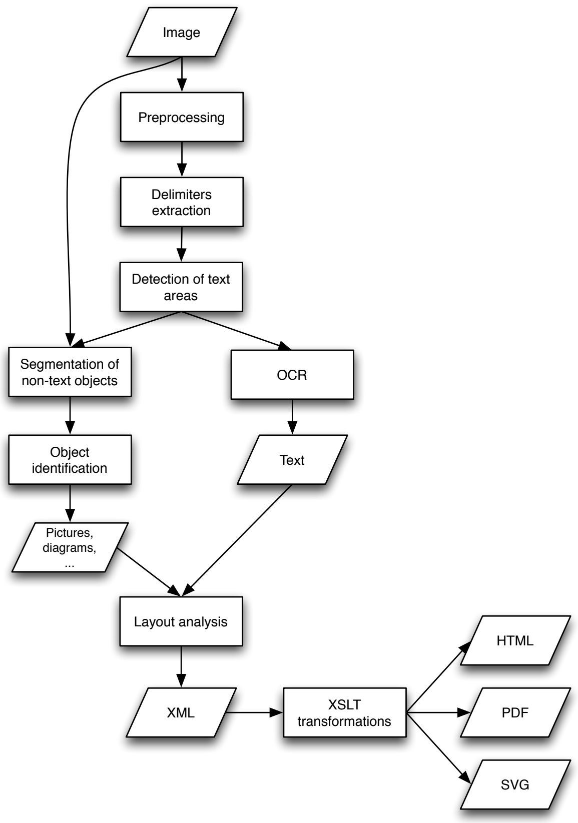


Fig. 2: Document reconstruction workflow.

the PRImA PAGE format[1] [12], which is used in the ICDAR page segmentation competition series. From the XML file, the whole document can reconstructed as a PDF or HTML document using XSL Transformations (XSLT) [13]. Recognized textual elements are rendered as blocks that can be selected as character strings. Pictures, logos and drawings, extracted from the original document image, are positioned such that the initial layout is globally preserved.

All the algorithms are provided in the platform as a library but they are also provided as command line tools to make them easier to integrate. A portable GUI has also been developed (see Fig. 3) to offer a user-friendly interface to the toolchain that can be used by non-experts. Parts of the toolchain can be customized from this graphical application; the user can then inspect results before saving them as PDF or HTML outputs.

C. Sample Code

Listing 1 shows a example of simple C++ program using our DIA module to extract text and non-text components from a document image. The result of this analysis are used to produce a structured document which is eventually saved as an XML file.

Text is processed first. Text components are usually well aligned, have a uniform size and are close to each other. So

[1]XML schema: http://schema.primaresearch.org/PAGE/gts/pagecontent/2009-03-16/pagecontent.xsd.

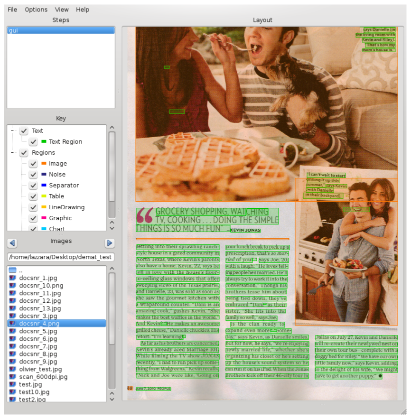


Fig. 3: A Graphical User Interface of our DIA module featuring document reconstruction services.

we try to regroup CCs by looking for their neighbors. We propose several strategies more or less robust. For instance in the case where the input is a structured document, every link between two CCs should be validated in both directions; in the case of a picture as input, there may be very few, possibly isolated text components, therefore the previous check is not enforced.

After text components are grouped together as text lines, the processing chain ensures that these groups meet some properties; corresponding links are invalidated if not. Again, several filters are proposed and new ones can easily be added. Filters can also be applied on a group of CCs to check a property not just on a link between two CCs. Such features are particularly useful when performing text extraction in pictures, where many false positive may arise.

After this step, the groups of CCs are very likely text components. Because of the nature of text characters and the way the grouping is done, we use a text line reconstruction algorithm which tries to merge groups of CCs as a single text line. During this step, typographical information are computed: font size, font color, space between character, space between words, “x” height, baseline, etc. The text can be recognized thanks to OCR software if needed. Final results of the text extraction include text locations, text metadata and the recognized character strings. The computation of each of these aspects is optional and depends on the chosen algorithms during this step.

IV. Contributions and Related Projects

A. Third-Party Projects

Table I summarizes different aspects of some popular DIA tools. On the one hand, Gamera and OCRopus are dedicated to domain experts, who have little or no knowledge in programming. They provide user interfaces or simple APIs so as to make them easy to use. These frameworks are more application-oriented. On the other hand, Leptonica is a library providing IP tools and state-of-the-art algorithms in pure C dedicated to users who both know C programming and DIA. It cannot be used as-is by an untrained user.

TABLE I: Comparison of some DIA systems.

Aspects	**Systems**				
	Gamera [3]	Leptonica [14]	OCRopus [15]	Qgar [4]	SCRIBO module
Target users	Domain experts	C Programmers	Domain experts	Domain experts and C++ programmers	C++ programmers and non-initiated users
Target uses	Development of applications dedicated to specific data sets	State-of-the-art implementations for prototyping and comparisons	DIA for books and old documents	Development, evaluation and tests in a dedicated environment	Development of DIA processing chains. DIA for non-expert users.
Main characteristics	Graphical platform for DIA and pattern recognition	General state-of-the-art IP library	OCR oriented	Platform for testing and evaluating DIA algorithms	Framework for creating new DIA algorithms
Status	Active	Active	Active	Stalled	Active
Documentation	Well documented	Well documented	Man pages	Minimalist	Minimalist
User language	Python, C++ (plug-ins)	C	Python	C++	C++
Implementation language	Python, C++	C	C++	C++	C++
User interfaces	GUI	CLI	CLI	GUI, CLI	GUI, CLI
IP tools	Limited, but can use external libraries	Present, not generic	Limited	Present, not generic	Present, generic
Document reconstruction	None	None	Native HTML output; PDF output with external tool	None	HTML and PDF outputs with external tool
License	GNU GPL v2	Creative Commons v3	Apache v2.0	GNU LGPL/QPL	GNU GPL v2
Supported OS	GNU/Linux, Mac OS, Windows	GNU/Linux, Mac OS, Windows	GNU/Linux, Mac OS, Windows	GNU/Linux, Windows	GNU/Linux, Mac OS, Windows (Cygwin)

Listing 1: Object extraction and document reconstruction.

```
// Open input image.
typedef image2d<value::label_16> L;
document<L> doc("input.png");
doc.open();

// Preprocessing.
toolchain::text_in_doc_preprocess(doc);

// Find delimiters.
image2d<bool>
  delimiters = extract::delimiters(doc, 81);
doc.set_delimitiers(delimiters);

// Find components.
value::label_16 ncomponents;
component_set<L> components =
  extract::components(doc, c8(), ncomponents);

// Build a set of lines and compute typological
// attributes for each text line.
line_set<L> lines = make::text_line_set(components);

// Recognize text.
lines = text::recognition(lines, "eng");
doc.set_text(lines);

// Extract non-text elements.
component_set<L> elements = extract::elements(doc);
doc.set_elements(elements);

// Save results.
scribo::io::xml::save(doc, "out.xml");
```

Qgar and Olena's SCRIBO module are in-between projects, both proposing a standard C++ library for DIA, command line tools and GUIs. These libraries provide state-of-the-art algorithms and research results in order to build new toolchains, and aims to provide reference code bases. Thus they remain easily extensible. However the design of Qgar is more oriented towards research concerns (including evaluation of methods), while the SCRIBO module is geared towards application development.

The DIA systems presented in this section use different strategies to provide IP tools, to be used as building bricks of more complex DIA methods. Gamera features some IP routines, but to provide a full-fledged IP framework, its harnesses external libraries, such as VIGRA [16]. OCRopus uses an IP library internally, but does not expose it through its user interface. Leptonica and Qgar both provide basic tools for IP. Finally, to our knowledge none of the aforementioned frameworks support true generic algorithms definitions in their user interfaces.

B. Limitations of Classical Tools

The aforementioned frameworks enable the development, test and evaluation of DIA techniques. However their utility regarding the maintenance, dissemination, reuse and extension of the implemented methods is often limited. The applicability of these tools is often circumscribed to a finite domain and it may be hard to extend them to support new use cases or new data types. Examples of limitations include:

1) Fixed Value Types: Most tools are able to process binary, 8-bit gray-level and 24-bits RGB color images, but they often do not support other types, ranging from simple ones (for instance gray-level values with better quantization, other color spaces, etc.) to more complex values (e.g. user-defined features vectors).

2) Lack of Flexibility: It is often useful to alter the behavior of an algorithm or of a processing chain. For instance, one may want to consider only a subset of an image (using a computed or user-defined mask), or process only the red channel of an RGB image, etc. When they are supported by a DIA framework, these variations of algorithms are often implemented by duplicating existing routines, or by creating intermediate data. The former strategy does not scale with the evolution of software and is error prone, while the latter introduces space and run time penalties.

3) Limited Core IP Framework: The extensibility and usability of a DIA framework not only depend on the existence of high-level document-related methods, but also on the availability of lower-level IP routines. Of the third-party projects presented above, only Leptonica and Qgar really provide an IP core (though a non-generic one).

C. *Contributions of the SCRIBO Module and Olena*

The proposed SCRIBO module aims at providing a very open framework to overcome these kinds of issues and enables the development of many various processing chains. It already contains components for every step of a classical DIA workflow, from the inputs to the results (processed images, recognized text, reconstructed documents, generation of outputs in PDF or HTML formats).

SCRIBO's main trait is that it is based on a mature, stable and generic IP library. It has not been designed as a standalone DIA platform but as a module of an IP framework, Milena. SCRIBO benefits from the flexibility, reusability and expressiveness of Milena. More precisely, this library contains many algorithms, written in a concise and legible manner, thanks to many objects representing usual IP entities (sites, domains, neighborhoods, windows, functions, etc.) as well as an intuitive syntax. These algorithms are reusable and many of them provide optimized generic variants for common input types [17]. The library offers many image data types, including less common ones like graph-based images or cell complex-based images, but also useful non-image types such as histograms.

Moreover, Milena supports lightweight image types implementing objects transformations named *morphers*. An image morpher is an object based on an existing image; depending on its nature, it alters the behavior of the initial image object. A morpher does not own nor duplicate image data: it relies on the underlying image for this. Morphers are a general tool to implement operations such as views, proxies, decorators, adapters, observers, etc. For instance they are useful to:

- adjust the interface of a type to a another one without duplicating data (e.g., "virtually" thresholding a gray-level image on-the-fly before passing it as input to a connected components labeling algorithm expecting a binary image);
- change the behavior of an image, e.g. by restricting its domain using a predicate function or a subset object; or by adding tracing or visualization mechanisms to an input image (so as to observe the execution of an algorithm without modifying its code);
- create a lightweight image based on one or several other images(s). For example, one may create a virtual RGB color image by combining three gray-level images, each one corresponding to a red, a green and a blue channels;
- apply a function to an image in a lazy fashion (rotation, subsampling, color-space conversion, etc.): the resulting image object behaves as the result of the application of the function to the original image.

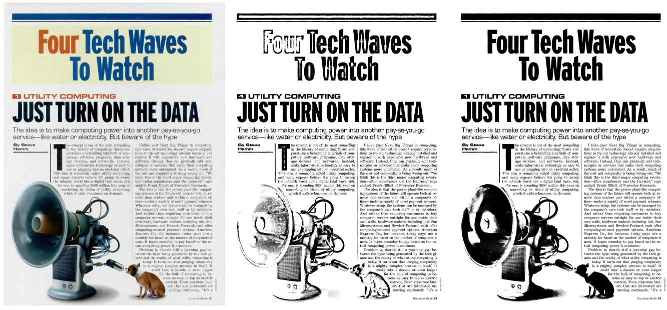

Four Tech Waves To Watch
UTILITY COMPUTING
JUST TURN ON THE DATA
The idea is to make computing power into another pay-as-you-go service—like water or electricity. But beware of the hype
(a) Original picture. (b) Binarization (c) Multi-scale variant.

Fig. 4: Multi-scale binarization.

Thanks to the generic design of Milena, image morphers are considered just like any other image inputs by algorithms. Their definition is also generic, so they are usable on any compatible image type. Using morphers is cost-effective with respect to memory and computation time, as they avoid expensive intermediate images. They are concise both in their definition and in their use, and they can be combined with no limitations.

V. Research Results

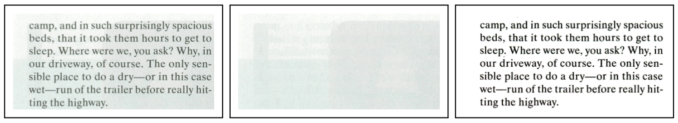
In addition to providing a DIA framework, the SCRIBO module also contains novel algorithms and techniques for some of the tasks performed during the analysis of a document. Since the main topic of this paper is the framework itself, we only mention these methods here. We will present these results in depth in future publications.

A. *Multi-Scale Binarization*

One of our binarization algorithm is based on the technique developed by Sauvola and Pietikäinen [18]. This algorithm is currently one of the best for document binarization [19], though some defects remain, including parameters which need to be set up according to the content. We provide a fast multi-scale version which automatically adjusts the parameters and significantly improves the results (see Fig. 4).

B. *Fast Robust Deskew*

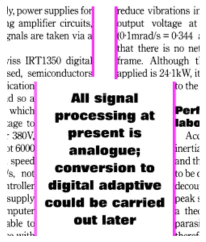
Our deskewing method has been designed to provide a quick detection of little angles, between -25 and $+25$ degrees, which covers most skews in practice. It is based on the combination of a grayscale filtering algorithm, the Sobel edge detection filter, and the classical Hough Transform [20]. The computational time of the Hough Transform is reduced thanks to a filtering algorithm selecting relevant pixels at the beginning of the deskewing process. The Sobel filter is then used to find a range in which the Hough voting scheme is to be applied.


camp, and in such surprisingly spacious beds, that it took them hours to get to sleep. Where were we, you ask? Why, in our driveway, of course. The only sensible place to do a dry—or in this case wet—run of the trailer before really hitting the highway.

camp, and in such surprisingly spacious beds, that it took them hours to get to sleep. Where were we, you ask? Why, in our driveway, of course. The only sensible place to do a dry—or in this case wet—run of the trailer before really hitting the highway.

(a) Original picture. (b) "Background" (c) "Foreground"

Fig. 5: Show-through effects removal.


ly, power supplies for
ıg amplifier circuits,
gnals are taken via a

/iss IRT1350 digital
sed, semiconductors
ication
d so a
which
age to
380V,
ot 6000
speed
's, not
ıtroller
supply
nputer
able to

reduce vibrations i
output voltage at
(0·1mrad/s = 0·344
that there is no ne
frame. Although t
applied is 24·1kW, it
to the

All signal processing at present is analogue; conversion to digital adaptive could be carried out later

Perf
labo
Acc
inertia
and th
to be
decou
peak
a the
parasi

Fig. 6: Tab-stop separators found using whitespace detection.

C. Morphological Show-Through Removal

In a document image, show-through effects can be considered as a background image over which objects like text and images are superimposed. We process the original image using morphological connected operators (area closing and opening) [21] removing objects in order to obtain a background image. From this image we deduce the foreground image (see Fig. 5).

D. Whitespace/Tab-Stop Detection

We provide algorithms to detect both whitespace and tab-stop delimiters in a document [22] (see Fig. 6). Text components are first connected and the resulting objects are grouped together vertically when they are aligned. The presence of whitespace nearby those groups enables the removal of false positives. This step allows the identification of the document's layout and columns.

VI. Conclusion

In this paper we have presented a new DIA platform, provided as a module of a generic and efficient image processing platform, Olena. The DIA toolchain described in this paper takes approximately 2 seconds for a whole analysis without OCR, with an Intel Core 2 Duo CPU at 3 Ghz and 2 Go of RAM on a 300 dpi A4 document (2340 × 3150 pixels). It is already used by the semantic desktop Nepomuk [23] of the KDE environment, to extract text and associate metadata to document images. All these tools will be part of the Mandriva Linux distribution in the next few months both as binary and source packages.

This module also offers yet unpublished original methods for binarization, show-through extraction, deskew and tab-stop detection, released as open source software under the GNU GPL. This DIA framework is scheduled for a release along with the next version of the Olena platform [2] this year.

Acknowledgment

The authors thank the reviewers for their useful comments on this paper.

References

[1] "SCRIBO, Semi-automatic and Collaborative Retrieval of Information Based on Ontologies," http://www.scribo.ws, 2010.

[2] EPITA Research and Developpement Laboratory (LRDE), "The Olena image processing platform," http://olena.lrde.epita.fr.

[3] M. Droettboom, K. Macmillan, and I. Fujinaga, "The Gamera framework for building custom recognition systems," in *Proceedings of the Symposium on Document Image Understanding Technologies (SDIUT)*, 2003, pp. 275–286.

[4] J. Rendek, G. Masini, D. Philippe, and T. Karl, "The search for genericity in graphics recognition applications: Design issues of the Qgar software system," in *Proceedings of the 6th IAPR International Workshop on Document Analysis Systems (DAS)*, S. Marinai and A. Dengel, Eds., vol. 3163. Springer, 2004, pp. 366–377.

[5] B. Coüasnon, "DMOS: a generic document recognition method, application to an automatic generator of musical scores, mathematical formulae and table structures recognition systems," in *Proceedings of the Sixth International Conference on Document Analysis and Recognition (ICDAR)*, 2001, pp. 215–220.

[6] EPITA Research and Developpement Laboratory (LRDE), "SCRIBO module for Olena," http://olena.lrde.epita.fr/Modules#SCRIBO, 2010.

[7] ——, "Online demo of the SCRIBO module," http://olena.lrde.epita.fr/Demos#Scribo, 2010.

[8] J. B. Buckheit and D. L. Donoho, "WaveLab and reproducible research," Stanford University, Department of Statistics, Tech. Rep. 474, 1995.

[9] S. Fomel and J. F. Claerbout, "Guest editors' introduction: Reproducible research," *Computing in Science and Engineering*, vol. 11, pp. 5–7, Jan. 2009.

[10] R. Levillain, Th. Géraud, and L. Najman, "Why and how to design a generic and efficient image processing framework: The case of the Milena library," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, Hong Kong, Sep. 2010, pp. 1941–1944.

[11] Th. Géraud and R. Levillain, "Semantics-driven genericity: A sequel to the static C++ object-oriented programming paradigm (SCOOP 2)," in *Proceedings of the 6th International Workshop on Multiparadigm Programming with Object-Oriented Languages*, Paphos, Cyprus, 2008.

[12] S. Pletschacher and A. Antonacopoulos, "The PAGE (Page Analysis and Ground-Truth Elements) format framework," in *Prooceedings of the 20th International Conference on Pattern Recognition (ICPR)*. Istanbul, Turkey: IEEE Computer Society, Aug. 2010, pp. 257–260.

[13] W3C, "XSL transformations (XSLT) version 2.0," http://www.w3.org/TR/xslt20/, Jan. 2007, W3C Recommendation.

[14] D. Bloomberg, "Leptonica: An open source C library for efficient image processing, analysis and operation," http://code.google.com/p/leptonica/.

[15] T. M. Breuel, "The OCRopus open source OCR system," in *Document Recognition and Retrieval XV*, B. A. Yanikoglu and K. Berkner, Eds., vol. 6815. San Jose, CA, USA: SPIE, Jan. 2008, p. 6815 0F.

[16] U. Köthe, "Reusable software in computer vision," in *Handbook of Computer Vision and Applications*, B. Jähne, H. Haussecker, and P. Geißler, Eds. San Diego, CA, USA: Academic Press, 1999, vol. 3: Systems and Applications, pp. 103–132.

[17] R. Levillain, Th. Géraud, and L. Najman, "Une approche générique du logiciel pour le traitement d'images préservant les performances," in *Proceedings of the 23rd Symposium on Signal and Image Processing (GRETSI)*, Bordeaux, France, Sep. 2011, in French, to appear.

[18] J. Sauvola and M. Pietikäinen, "Adaptive document image binarization," *Pattern Recognition*, vol. 33, no. 2, pp. 225–236, 2000.

[19] M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation," *Journal of Electronic Imaging*, vol. 13, no. 1, pp. 146–168, 2004.

[20] L. A. Fernandes and M. M. Oliveira, "Real-time line detection through an improved Hough transform voting scheme," *Pattern Recognition*, vol. 41, no. 1, pp. 299–314, 2008.

[21] P. Salembier and M. Wilkinson, "Connected operators: A review of region-based morphological image processing techniques," *IEEE Signal Processing Magazine*, vol. 26, no. 6, pp. 136–157, November 2009.

[22] R. Smith, "Hybrid page layout analysis via tab-stop detection," in *Proceedings of the 10th International Conference on Document Analysis and Recognition, 2009 (ICDAR)*, Barcelona, Spain, Jul. 2009, pp. 241–245.

[23] KDE, "Nepomuk, the social semantic desktop," http://nepomuk.kde.org.