Gamera: A Structured Document Recognition Application Development Environment

Karl MacMillan, Michael Droettboom, and Ichiro Fujinaga

Peabody Conservatory of Music
Johns Hopkins University
1 East Mount Vernon Place, Baltimore MD 21202
email: {karlmac,mdboom,ich}@peabody.jhu.edu

ABSTRACT

This paper presents a new toolkit for the creation of customized structured document recognition applications by expert users. This open-source system, called Gamera, allows a user, with particular knowledge of the documents to be recognized, to combine image processing and recognition tools in an easy to use, interactive, graphical scripting environment. Additionally, the system can be extended through a C++ module system.

1. INTRODUCTION

This paper[1] presents a new toolkit for the creation of domain-specific structured document recognition applications by expert users. This system, called Gamera, allows a knowledgeable user to combine image processing and recognition tools in an easy to use, interactive, graphical scripting environment. The applications created by the user are suitable for use in a large-scale digitization project; they can be run in a batch processing mode and easily integrated into a digitization framework. Additionally, a module (plug-in) system allows experienced programmers to extend the system. This paper will give an overview of Gamera, describe the user environment, and briefly discuss the plug-in system.

2. MOTIVATION AND GOALS

Gamera is being created as part of the Lester S. Levy Sheet Music Project (Phase Two). The Levy collection represents one of the largest collections of sheet music available online.[2] The Collection, part of the Special Collections of the Milton S. Eisenhower Library at the Johns Hopkins University, comprises nearly 30,000 pieces of music which correspond to nearly 130,000 sheets of music and associated cover art. It provides a rich, multi-faceted view of life in late 19th and early 20th century America, featuring famous songs such as “The Star-Spangled Banner”, “Hail Columbia”, and “Yankee Doodle Dandy” along with engravings, lithographs, and many forms of early photo reproduction on song covers.

The goal of the Levy Project (Phase Two) is to create an efficient workflow management system to reduce the cost and complexity of converting large, existing collections to digital form. From the beginning of the project, optical music recognition (OMR) software was a key component of the workflow system. The creation of a flexible OMR tool is necessary because of the historical nature of the Levy collection; existing OMR systems are not designed to handle the wide range of music notation found in the collection or deal with the potentially degraded documents. OMR alone is not sufficient for the complete recognition of the scores in the Levy collection as they are not comprised only of musical symbols. Text is also present as lyrics, score markings, and metadata. It was hoped, however, that an existing optical character recognition (OCR) system would be able to process such text. Early trials of existing systems revealed there are many problems with the current generation of OCR software within this context.

To address the need for OCR in the Levy project the Gamera system was created. Gamera is an extension of the existing OMR system to a general symbol recognition system. By creating a general symbol recognition system it is possible to use the same technology that allows the OMR system to perform well on the musical portions of the Levy collection to recognize the text. In addition to serving the needs of the Levy project, we hope that the system may be used in the future for the recognition of historical documents and any other structured documents that current recognition systems do not adequately address.

In addition to generalizing the system, a graphical programming environment has been added to ease the adaptation of the system by users with expert knowledge of the documents to be recognized. This environment provides an easy-to-learn scripting language coupled with a graphical user interface. The goal is to allow the user to experiment easily with algorithms and recognition strategies during the creation of custom scripts for the recognition process. This will allow users to leverage their knowledge of the documents to customize the recognition process. It is hoped that users without extensive computer experience can effectively use this environment with a small amount of training. The scripting environment does contain, however, a complete, modern programming language that will allow advanced users considerable flexibility and power.

[1] Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page.

[2] The Lester S. Levy Collection of Sheet Music, <http://levysheetmusic.mse.jhu.edu>

3. ARCHITECTURE OVERVIEW

Gamera is primarily a toolkit for the creation of applications by knowledgeable users. It is composed of modules (plug-ins), written in a combination of C++ and Python, that are combined in a very high-level scripting environment to form an application. The overall design is inspired by systems like MathWorks Matlab, CVIP tools (Umbaugh 1998), or spreadsheet macros. In Gamera, modules perform one of five document recognition tasks:

1. Pre-processing
2. Document segmentation and analysis
3. Symbol segmentation and classification
4. Syntactical or semantic analysis
5. Output

Each of these tasks can be arbitrarily complex, involve multiple strategies or modules, or be removed entirely depending on the specific recognition problem. Additionally, keeping with the toolbox philosophy of Gamera, the user of the system has access to a range of tools that fall within the general category of these tasks. The actual steps that make up the recognition process are completely controlled by the user.

In addition to flexibility Gamera also has several other goals that are important to the Levy project and to large-scale digitization projects in general. These are:

1. A batch processing mode to allow many documents to be recognized without user intervention.
2. Open-source so that the software can be customized to interact well with the other parts of the digitization framework.
3. The system designed to run on a variety of operating systems including Unix, Microsoft Windows, and Apple MacOS.
4. Recognition confidence output so that collection managers can easily target documents that need correction or different recognition strategies.

The first three goals have been achieved while the last goal is currently being developed.

3.1 Pre-processing

Pre-processing can involve almost any standard image-processing operation including noise removal, blurring, de-skewing, contrast adjustment, sharpening, binarization, or morphology. Any number of operations may be necessary to take a raw input image and prepare it for recognition, but the output of this step must be a binary image for the rest of the recognition process.

Many documents, particularly historical documents like those in the Levy collection, will depend on this part of the recognition process to ensure overall good performance of the system. Discoloration of the documents makes binarization difficult and often requires locally-adaptive algorithms (Trier and Torfinn 1995). Additionally, broken lines often cause problems in the segmentation of symbols. Experiments suggest that simple blurring or morphology (Serra 1982) may help with these difficulties.

3.2 Document segmentation and analysis

Before the symbols of a document can be classified, an analysis of the overall structure of the document is often necessary. The document segmentation and analysis process is designed to anaylse the overall structure of the document, segment it into sections, and perhaps remove elements (Haralick 1994; Yanikoglu 1998). For example, in the case of music recognition, it is necessary to identify and remove the staff lines in order to be able to properly separate the individual symbols. The proper identification of the staff lines and the grouping of the lines into staves and systems is essential to the classification of symbols and later to the interpretation of those symbols. Similarly, text documents may require the identification of columns, paragraphs, lines, or tables.

3.3 Symbol segmentation and classification

The segmentation and classification of symbols is the core of the Gamera system. The current implementation provides tools for the creation of simple heuristic classifiers, template-based image matching, and a learning classifier using the k-nearest neighbor algorithm enhanced with a genetic algorithm. Other possible classification algorithms include neural-nets, decision trees, or hidden markov models. The use of both learning and heuristic classifiers allows for the balancing of flexibility, training time, and recognition speed.

3.4 Syntactical or Semantic analysis

This process reconstructs a document into a semantic representation from the individual symbols. Examples include combining stems, flags, and noteheads into musical notes, or grouping words and numbers into a table. Obviously this process is entirely dependent on the type of document being processed and is a likely place for large customizations by knowledgeable users.

3.5 Output

Output converts either the raw symbols or the post structural interpretation data into a suitable format for storage.

4. USER ENVIRONMENT

The goal of the user environment is to leverage the knowledge and skills of the user about the documents being recognized. This is accomplished by creating a dynamic scripting environment and graphical user interface where users can experiment with various Gamera modules.

4.1 Scripting Environment

Gamera includes a complete scripting environment that a user can use to create custom recognition systems quickly and easily. The

scripting environment tries to be easy to use, flexible, and extensible.

4.1.1 Ease of Use

Perhaps the most important aspect of the Gamera scripting environment is ease of use by users with limited computer programming experience. As previously stated, the targeted user is a person with expert knowledge of the documents to be recognized that may or may not have computer programming experience. In order to meet this goal Python was chosen as the foundation and extensions were written that are as easy to use as possible.

Python is a popular, general purpose scripting language often praised for it simplicity and elegance (Gauld 2000). Additionally, Python has been used as a teaching language with considerable success. For this reason, we believe that Python is a good choice for the basis of the scripting environment. The existence of books and tutorials about the language also means that there is more help available to users than with a custom scripting language.

In order to transform Python from a general purpose scripting language to scripting environment tailored to the needs of Gamera users, a set extensions were written in a combination of Python and C++. Example 1 shows a script for OMR. This script gives a good indication of the high-level of Gamera scripts.

```
# load an image
image = Image()
image.load_image('example.tiff')
# Convert to binary using the Otsu thresholding
image.otsu_threshold()
# Remove staves and store information about them
staves = image.remove_staves()
# Perform recognition on the image - this is a
# two step process. First the image is
# segmented and then the K-nn classifier is
# used on the individual symbols
symbols = image.connected_components()
classified_symbols = knn_classify(symbols, 'knn-
database.knn')
# Interpret the symbols with the Optical Music
# Interpretation object
omi = OMI()
omi.interpret(image, staves, classified_symbols)
# Output GUIDO and the MIDI
omi.save('example.gmn', 'guido')
omi.save('example.mid', 'midi')
```

4.1.2 Flexibility

Flexibility is the second most important goal for the scripting environment. Again, this aspect of the scripting environment is facilitated by the choice of Python. Because Python is a general-purpose programming language, a large portion of the system can be implemented directly in standard Python. In general, only those algorithms that need direct access to image pixels are written in C++. This allows users to customize existing modules written in Python, combine the low-level building blocks into new modules, or write modules from scratch.

4.1.3 Extensibility

Despite the flexibility of the scripting environment, not all algorithms can be suitably implemented in Python. For this reason, a C++ module system for use by experienced programmers has been developed. Some of the features of this system are:

1. Automatic binding of C++ code to Python.
2. Runtime addition of C++ modules as methods to Python classes.
3. Abstraction of the data storage format of image data using C++ templates to allow convenient access to compressed images.
4. Flexible programming interface allows the easy conversion of existing C and C++ code that uses a variety access methods to image data.

4.2 Graphical Interface

In addition to the scripting environment Gamera includes a graphical user interface that allows the interactive display and manipulation of images using the scripting environment. This can be as simple as displaying the results of a pre-processing algorithm or as complex as complete interface for training. Figure 1 shows a sample Gamera editing session. Again, like the scripting environment, the graphical interface is created with standard tools entirely in Python allowing users to extend and modify the system.

5. CONCLUSION

A graphical programming environment for the creation of document recognition applications was described. This system, called Gamera, is designed to be used by people with expert knowledge of the documents to be processed. These users are not required to have extensive computer experience; the system can be effectively used with a small amount of training. Users with considerable programming experience can also create custom modules in Python or C++ to extend the system. The applications created by this system are suitable for large-scale digitization projects because they can be run in batch mode and integrated into the digitization framework.

6. ACKNOWLEDGEMENTS

The second phase of the Levy Project is funded through the NSF's DLI-2 initiative (Award #9817430), an IMLS National Leadership Grant, and support from the Levy Family.

7. REFERENCES

Gauld, A. 2000. *Learn to program using Python.* Boston: Addison-Wesley.

Haralick, R. M. 1994. Document image understanding: Geometric and logical layout. In *Proceedings of IEEE Computer Society Computer Vision and Pattern Recognition.* 385–90.

Serra, J. P. 1982. *Image analysis and mathematical morphology.* London: Academic Press.

Trier, O. D., and T. Torfinn. 1995. Evaluation of binarization methods for document images. In *Proceedings of IEEE Computer Society Pattern Analysis and Machine Intelligence.* 312–5.

Yanikoglu, B.A., and L. Vincent. 1998. Pink panther: A complete environment for ground-truthing and benchmarking document page segmentation. *Pattern Recognition* 31(9): 1191–204.

Umbaugh, S. E. 1998. *Computer vision and image processing: A practical approach using CVIPtools.* Upper Saddle River, NJ: Prentice Hall.

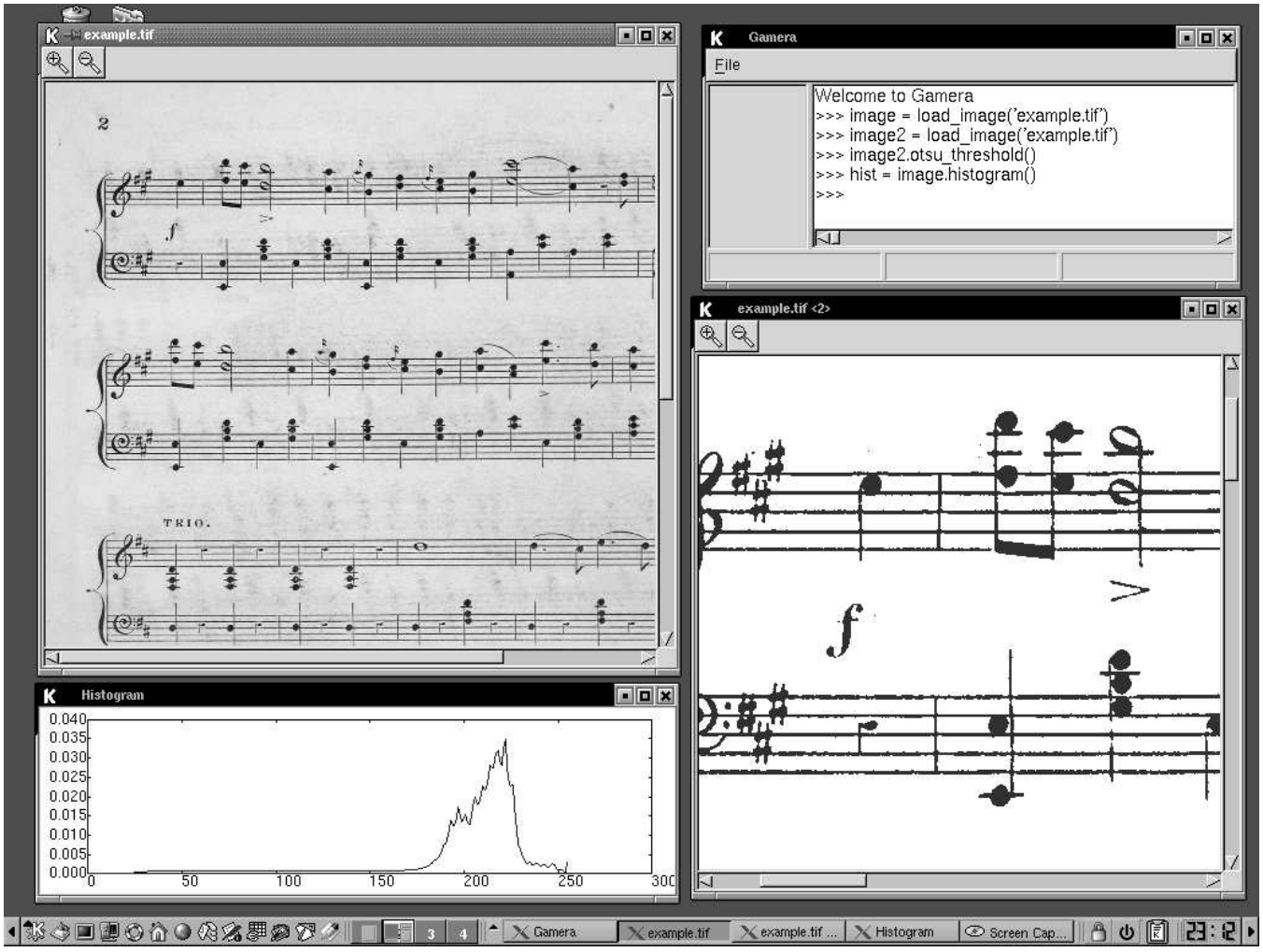

Figure 1. Screenshot of a grey-scale image, its histogram, the script window, and a copy of the image converted to binary format.