# Web interface for image processing algorithms

S. Chastel, G. Schwab, and D. Paulus
Active Vision
Institut für Computervisualistik – Universität Koblenz-Landau
Universitätsstr. 1 – 56070 KOBLENZ – Germany
agas@uni-koblenz.de

## ABSTRACT

In this contribution we present an interface for image processing algorithms that has been made recently available on the Internet (http://nibbler.uni-koblenz.de). First, we show its usefulness compared to some other existing products. After a description of its architecture, its main features are then presented: the particularity of the user management, its image database, its interface, and its original quarantine system. We finally present the result of an evaluation performed by students in image processing.

**Keywords:** Programming environment, interface, Internet

## 1. INTRODUCTION

The use of multimedia techniques is nowadays a reality. Concerning its use in the field in image processing, it can be however stated that only very few free systems are proposed over the World Wide Web. This paper will present one of these tools which has been recently made available: a Web interface for PUMA. PUMA is a library consisting of C++ image processing classes. It stands for the german acronym meaning Programming Environment for Pattern Analysis.

There exist some products that allow the use of image processing algorithms. Most of the time they are offered as complete software products as Khoros, or Matlab for instance. However their access requires a costly registration or license. They often provide limited or restricted access to the source code of their functionalities. Lastly, they do not provide any other interface but their own and use specific data formats: communications between different entities is made difficult and in some cases impossible.

Considering that fact, we thought about a possible solution consisting of a library having the following properties: its core is written in a non-proprietary language (namely C++); its source code is open; its source code is free; it does not rely on any particular operating system; it does not rely on any proprietary library, although its configuration allows the inclusion of commonly used packages, such as ImageMagick.[1]

Over the years PUMA has been developped by many researchers and students in Erlangen University and in Koblenz-Landau University. Its components group any image processing algorithms with potential interest. Filters or segmentation tools along with dedicated color images tools, edge detection algorithms and comparison algorithms. It is based groundly on the intensive use of C++ library. It is now provided with different interfaces: a classic command line one, a graphical one using Tcl/Tk and more recently an interface which that can be used through Internet connection.

The aim of such an interface is multiple. The first of its purposes is undoubtly educational: any student becomes able to see the effect of an image processing algorithm integrated to the system in quasi real time. For the same reasons the illustrations of lectures becomes also easier: focusing on advantages or drawbacks of a method becomes obvious. The second aim of such a tool might be its implication in the use of image processing methods in industrial processes. Our idea is not to provide a full operating system instantaneously able to allow the take of decisions concerning quality control for instance, but to give the possibility at a very low cost (that is for free) for image processing engineers to test some existing and high-end methods on the problem that they are concerned with. The proper tuning of control parameters and optimization of these methods is after that a question of industrial concern.

This paper is divided into four parts. First we give a short overview of the tools that are available over the Internet and that offer a web interface. The second part will mainly concern the internal mechanisms of PUMA

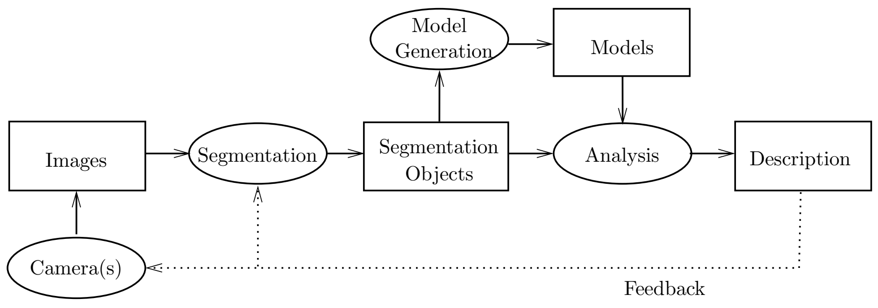


**Figure 1.** Data flow in an image analysis system.[3]

and more precisely the definition of the interface which is required in order to add external components to the database of algorithms. In the third part we will show the practical use of the interface we propose. We will finally conclude in the fourth by a discussion on the usefulness of that tool and some of its possible extensions.

## 2. ARCHITECTURE

The general problem of image analysis is to find an appropriate symbolic description of the input images. The description needs to be useful in the current problem domain. Sometimes this means that the most precise description has to be found, in some other cases a less detailed result which can computed quicker will be sufficient. Image analysis may be divided into several sub–problems. After an initial preprocessing stage, images are usually segmented into meaningful geometric parts. The resulting data structure can be summarized in a so-called segmentation object.[2]

Segmentation objects are matched to models in a knowledge base that contains expectations of the possible scenes of the problem domain.

### 2.1. Data Flow

Various data types during image segmentation, like images, lines, regions, etc., provide data abstraction of the problem. Segmentation may be seen as a data flow problem relating these various representations. The main components of an image analysis system, as they are present in many implementations, are shown in Figure 1; data is acquired with an input device, e.g. a camera, and transformed to a description. Processing tasks are shown in oval boxes; data is depicted as rectangles.

The problem of finding an optimal match and the best segmentation can be seen as an optimization problem and is formulated as such.[4] Optimization may search for the best possible match as well as include efficiency considerations which are crucial in real–time image processing.

Knowledge-based vision uses a model of the scene for image interpretation. The scene may be decomposed into object–models which can be represented using their structural properties and relations or as statistical object models.[5]

The system architecture in Figure 1 implies various interactions between modules. Information has to be exchanged by means of well defined interfaces. Modules and data structures with well-defined interfaces are naturally implemented as classes in object-oriented programming.

The segmentation object is a central idea for the data representation independent from the algorithms used for image segmentation, and can be used as an interface between data-driven segmentation and knowledge-based analysis. Models can be described in a similar formalism.[6]

## 2.2. Image Analysis Systems

If image processing is seen as a data flow problem , the command interpreter of the operating can be used to compose operations from single processes. This strategy has been applied in the system HIPS[7] which uses Unix–pipes to create sequences of operations. The khoros system also provides a command line interface; the automatically created sequences of program invocations use intermediate files as interfaces.

The TABS system[8] uses the Tcl/Tk language and interface to compose image understanding algorithms. The components may be written in C or C++.

An interesting effort has also been made by the image community with the Open Source Computer Vision Library (OpenCV).[9] This library is mainly aimed at real time computer vision. It provide many different tools from basic image functions to statistical estimators. According to its authors, its aim is to "aid commercial uses of computer vision [...] by providing a free and open infrastructure where the distributed efforts of the vision community can be consolidated and performance optimized". The components of that library are written in C or C++.

## 3. WEB INTERFACE

We made some PUMA image processing algorithms available online. In that part, we present the features of that interface that are offered to users. We explain these features in the same order as the user will meet them.

### 3.1. User management

Users are divided into two groups: unregistered ones and contributors. We must also mention the existence of administrators that have special privileges compared to other classical users. All users have the possibility to use any algorithm which is made available. By users, we mean students, scientists and even simply curious people who want to test an image processing algorithm, to evaluate its advantages or its drawbacks.

Unregistered users have limited possibilities compared to contributors. We will detail the other features that are offered to contributors in section 3.4. They can access an image database that we make available or upload their own images in order to run an algorithm on it. That upload is realized with a *multipart / form-date* HTML form. The uploaded image is then checked and if it is recognized as a supported format file, the image temporarily added into a database as a so called *session image* as it can be seen in Figure 2.

### 3.2. The image database

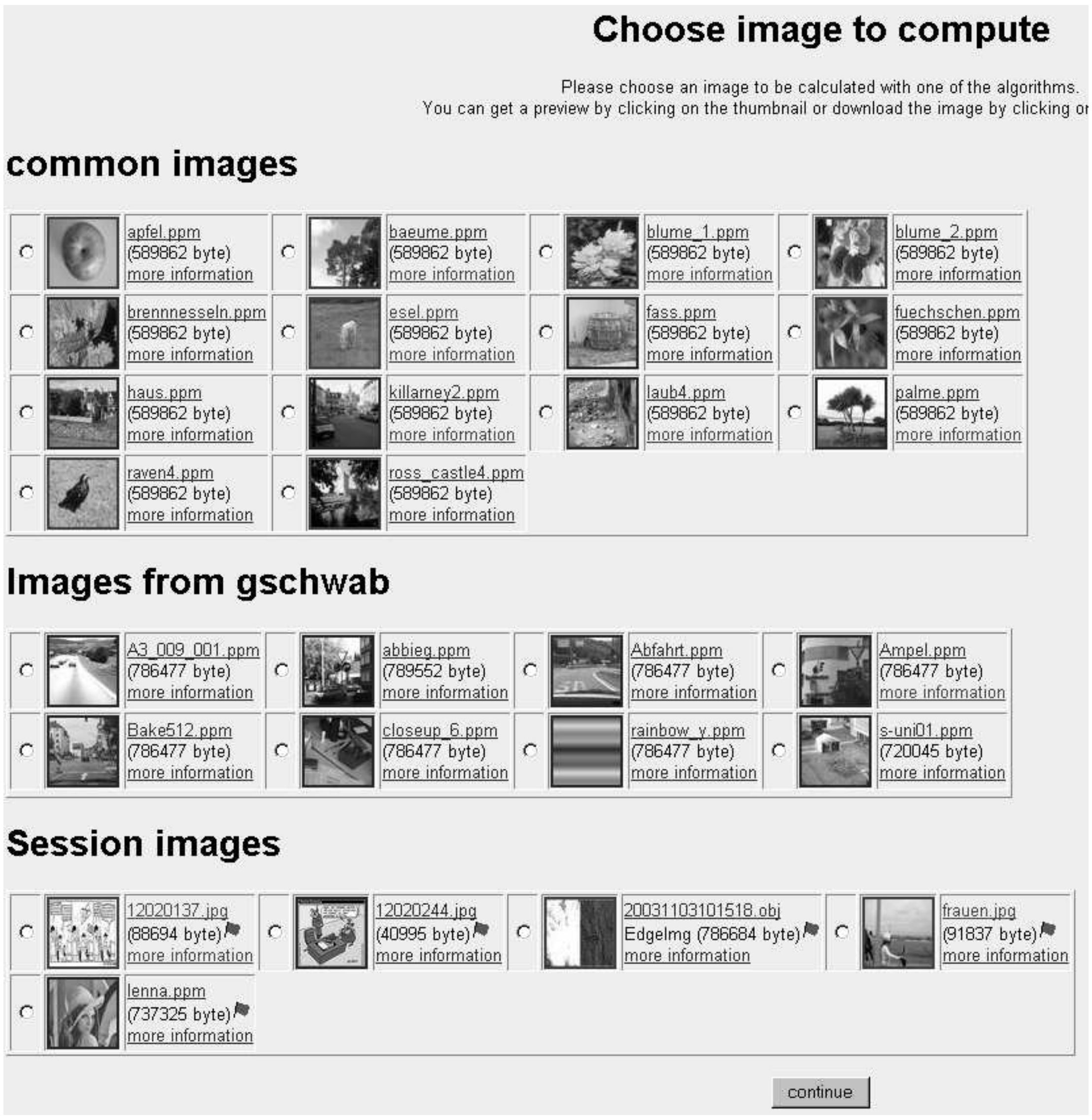
The result of the process is temporarily stored in the own user database and is also referred as being a session image. The composition of different methods is therefore made possible for any user. The complete data flow from upload to storage in the session database is described in figure 3. Those images are erased after the user disconnected from the application.

Concerning images that are in the database, they are available in its original format, but a description of the image is also given: the name of its author, its size, and possibly text information concerning its contents or the way it has been acquired for instance. We offer a preview of images using $51 \times 51$ thumbnails. For the moment, we do not provide any means to search for a special image. We hope to provide soon a first generation image retrieval system.[10] A more evolved retrieval system is not foreseen yet but it might be interesting to integrate queries involving more perceptual features. Note that this is not our primary aim here though.

### 3.3. The interface

Once an image has been chosen, the user has to select the algorithm thanks to a simple radio button 4. For each proposed algorithms currently in our database, a short description of its goal is shown, in order to lead unexperienced users in their choice. The list of algorithms that are available in our system as well as their names and description are automatically generated.

After that choice of an algorithm, the next interface consists of the different parameters that have to be tuned. That interface, shown in Figure 4, is also built automatically from the binary program that effectively realizes the image process. More precisely, it is extracted from the help provided with any PUMA program when it is used in command line.


## Choose image to compute

Please choose an image to be calculated with one of the algorithms.
You can get a preview by clicking on the thumbnail or download the image by clicking o

### common images

| | | | |
|---|---|---|---|
| apfel.ppm (589862 byte) more information | baeume.ppm (589862 byte) more information | blume_1.ppm (589862 byte) more information | blume_2.ppm (589862 byte) more information |
| brennnesseln.ppm (589862 byte) more information | esel.ppm (589862 byte) more information | fass.ppm (589862 byte) more information | fuechschen.ppm (589862 byte) more information |
| haus.ppm (589862 byte) more information | killarney2.ppm (589862 byte) more information | laub4.ppm (589862 byte) more information | palme.ppm (589862 byte) more information |
| raven4.ppm (589862 byte) more information | ross_castle4.ppm (589862 byte) more information | | |

### Images from gschwab

| | | | |
|---|---|---|---|
| A3_009_001.ppm (786477 byte) more information | abbieg.ppm (789552 byte) more information | Abfahrt.ppm (786477 byte) more information | Ampel.ppm (786477 byte) more information |
| Bake512.ppm (786477 byte) more information | closeup_6.ppm (786477 byte) more information | rainbow_y.ppm (786477 byte) more information | s-uni01.ppm (720045 byte) more information |

### Session images

| | | | |
|---|---|---|---|
| 12020137.jpg (88694 byte)🚩 more information | 12020244.jpg (40995 byte)🚩 more information | 20031103101518.obj EdgeImg (786684 byte)🚩 more information | frauen.jpg (91837 byte)🚩 more information |
| lenna.ppm (737325 byte)🚩 more information | | | |

continue

**Figure 2.** Screenshot of (an excerpt of) the different image databases

## 3.4. Contributors. Quarantine system

Contributors have two privileges. The first of them is that they have persistent sessions, that is, they can recover images that have been generated during image sessions or their own images. Moreover, they can submit these images to the group of administrators in order to make them available in the image database. When such an operation is performed, the contributor has to fill in required fields such as the name of the author of the image, its origin and can also provide more information on the picture itself. It is stored in a quarantine system where images stay until administrators take a decision about it. Administrators have then to judge about of the relevance of the images provided by the contributors. As soon as an image is considered being relevant, it is permanently added to the common image database with the informations provided by the contributor and is removed from the quarantine system. Images that are currently in quarantine are shown with a small (red) flag (as it can be seen in the lower part of the Figure 2).

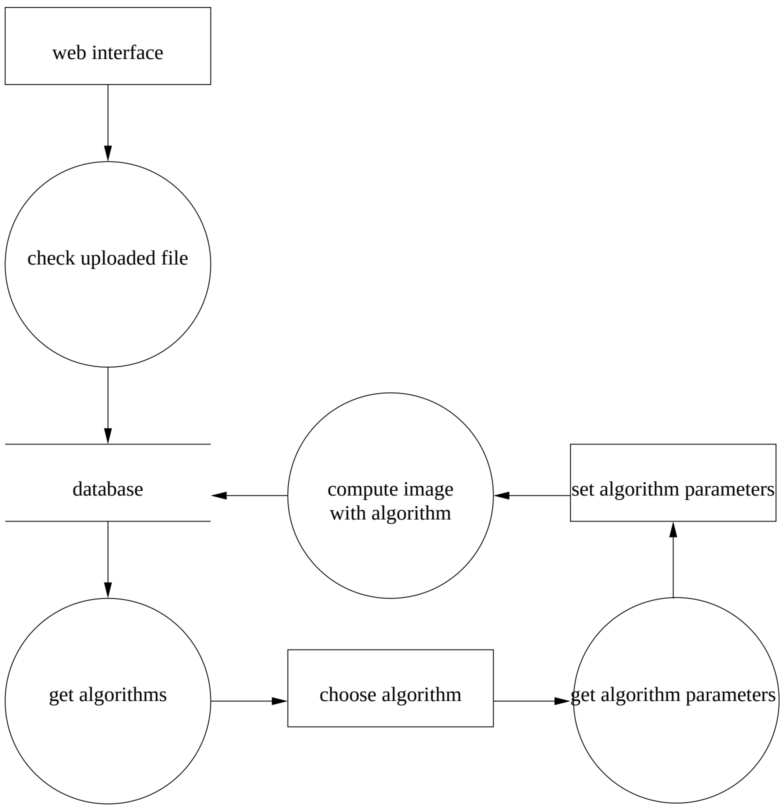


**Figure 3.** Data flow for image upload

## 3.5. Available Algorithms

The algorithm that we currently provide can roughly divided into three main categories according to the type of the result they provide. Some of them (`Mean`, `Gauss`...) give as a result an image which is of the same type (and we provide the same format) as the input, that is a real mono-component (gray level) image or multi-component (color) image. The second category is made of edge images. They are generated by edge detectors such as `sobel` (implementing the well-known Sobel operator), `robertsX` (Roberts cross edge detector[11]) or `robinson` (Robinson edge detector[12]) for instance. Such edge images consist of an array of edge direction and edge strength for every position in the input image. Finally segmentation algorithms belong to the third class currently provided. These algorithms as `Threshold` or `CSC`[13] provide a labeled image that can consist of one or more classes that build up a partition of the set of pixels.

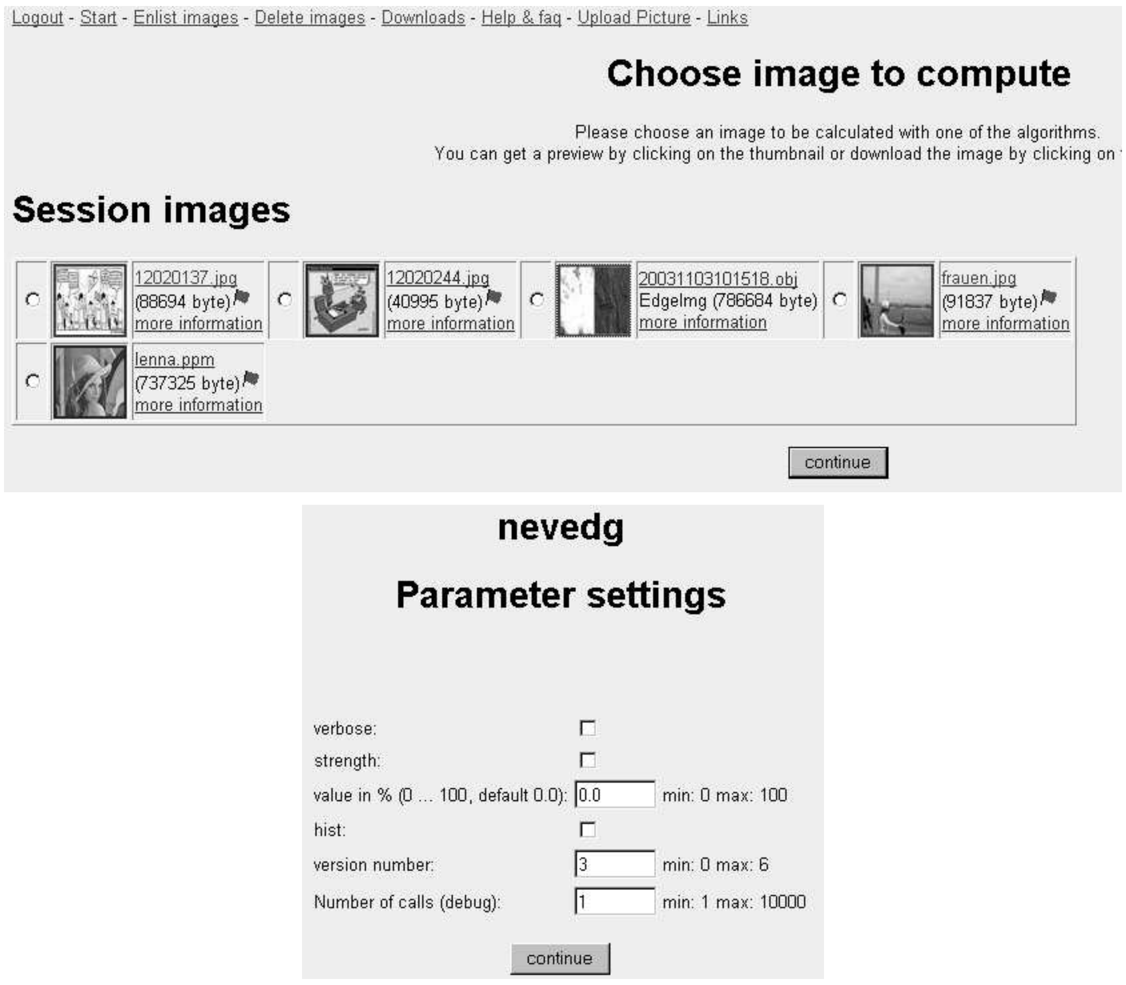
## 3.6. Data Type Management

Input images stemming from a web upload (Figure 3) are checked for a valid format type. ImageMagick[1] provides an easy solution for this. As any PUMA program can read all formats known to ImageMagick, these files can be stored in their original format in the database.

An application of a program listed in section 3.5 will create an object which may be either an image or some initial symbolic description, i.e. an edge image or a segmentation image. As Internet browser cannot support these symbolic descriptions, we provide preview for them JPEG images.

## 4. EVALUATION

To judge about the adequacy of our interface we asked students of the University of Koblenz-Landau to evaluate it. All of them were studying computer science and/or computer vision. They were therefore familiar with the


Logout - Start - Enlist images - Delete images - Downloads - Help & faq - Upload Picture - Links

**Choose image to compute**

Please choose an image to be calculated with one of the algorithms.
You can get a preview by clicking on the thumbnail or download the image by clicking on

**Session images**

| | | |
|---|---|---|
| ○ | | 12020137.jpg (88694 byte) more information |
| ○ | | 12020244.jpg (40995 byte) more information |
| ○ | | 20031103101518.obj EdgeImg (786684 byte) more information |
| ○ | | frauen.jpg (91837 byte) more information |
| ○ | | lenna.ppm (737325 byte) more information |

continue

**nevedg**

**Parameter settings**

| | | |
|---|---|---|
| verbose: | ☐ | |
| strength: | ☐ | |
| value in % (0 ... 100, default 0.0): | 0.0 | min: 0 max: 100 |
| hist: | ☐ | |
| version number: | 3 | min: 0 max: 6 |
| Number of calls (debug): | 1 | min: 1 max: 10000 |

continue

**Figure 4.** Choice of the algorithm to run on the input image and parameter settings interface

use of computers, even if some of them were just starting their studies in that scientific field. Unfortunately only 20 students gave feedbacks to our queries.

All students were initially unregistered users and we did not explain them that the tool that they have to judge was dedicated to on-line testing of image processing algorithms. We asked them simply to connect to our web server and to make a short feedback of their experience. Three main questions were mainly of our concern: how they judge the usefulness of the tool, what they thought about the speed of the system, and the format of images that they preferred to use. We also asked them about the algorithms that they would like to see included in the interface.

For the first question (i.e. How do you judge that service?), three answers were provided: “useful”, “useless”, and “I don’t know”. For the second one (i.e. What do you think about the speed of the system?), there were also three possible answers: “Rather fast”, “rather slow”, and “I don’t know”. Concerning the preferred input format of images, we offered them to fill in a text field. We could have checked the input format of the images that they used to perform the test. We preferred to ask them directly because it could have happened that they used images from our database, as they were just testing the tool, instead of trying with their own images and to see if there would be some unsupported format that we should add. The results can be seen in Table 1.

Concerning the usefulness of the tool, 90% of the students found our interface “useful”, one student did not know what to think about it, and one found it useless. When we asked them, why they found the tool useful,

| Usefulness | | |
|---|---|---|
| Useful | Useless | No opinion |
| 18 | 1 | 1 |

| Speed | | |
|---|---|---|
| Fast | Slow | No opinion |
| 16 | 1 | 3 |

| Preferred format* | | |
|---|---|---|
| JPEG | PGM | Others |
| 18 | 3 | 5 |

**Table 1.** Complete result of feedbacks. *Multiple answers could be given

their answers showed clearly that they thought about the possible use of the interface during their studies. They understood better the effect of such and such parameter of a method and their effect on an image. Our primary aim, that is provide a freely available tool for education, is therefore partly fulfilled. Concerning the two negative ("I don't know" and "useless") comments, it appeared that the students justified their answers by explaining that the number of image processing methods in our interface was too small. We are currently aware of that problem and we will provide more of them regularly.

Concerning the speed, 80% found the interface "fast", 15% did not know and 5% "slow". That question was probably ill-posed. We thought about the required to load the web pages themselves, not the

time that is required by an algorithm to perform on an image, as too many uncontrolled parameters may drastically affect the performance of our system: a very large uncompressed image in bytes size may require a lot of time to upload; a very large image in number of pixels will clearly affect the time required for the computation as soon as the complexity of the algorithm is more than linear. The load of the system is an essential factor too. If the success of our interface increases, we think about distributing the computations over a network. Note that it is possible for algorithms taking a finite number of values, and for images in the common database, the results can be a priori computed and stored on some device that can be accessed when a user gives the appropriate query. Such a system will clearly avoid unnecessary computations and improve the speed of our interface. We must also say that none of them complained about a failure of the interface as it is often the case while using Java-based interfaces. Five different browsers (namely Netscape, mozilla, lynx, konqueror, Internet Explorer) under three different systems (Linux, Microsoft Windows, and MacOS) were used: no complain was given concerning a missing plug-in or an error from the interface.

About the format of images that the users wanted to see supported, all answers concerned formats that were indeed supported. A lot of users simply tested our algorithms with images from the database. However one of them asked was wondering about fig and Postscript formats. Such a remark is interesting as it made us wonder about vectorial formats and their possible extensions for objects that might be represented as images, but that can be stored as objects that are not images, for instance, regular shapes, and maybe textured areas. It shows the accuracy of our previous remark in paragraph 3.6 concerning the use of unified standards to describe the contents of images.

## 5. CONCLUSION. PROSPECTIVE WORK

An new interface for image processing algorithms has been proposed. It relies on the use of a C++ library dedicated to image processing and image analysis. That library is freely available for download as a tarball or through a CVS repository.[14] Even if the number of feedbacks is not very high, our primary goal, that is provide an educational tool for students in image processing, is partially fulfilled. These students clearly expressed their will to see the number of algorithms increased. However from their queries for tools, it clearly appeared that they want tools dealing more with image analysis as edge detectors or segmentation tools, than image processing tools that is, those that convert an image into another one. Such tools implicitly require the definition of data formats that are not necessarily image ones. PUMA currently support them (both for edges and segmentations) and that is the reason why we require to work with that programming environment.

Lastly, an interesting perspective might be to integrate a feedback concerning a perceptual feedback of the user. Let us take the example of an edge detection algorithm. It might be fruitful to ask the user once its query has been completed if the result of the algorithm, that is the delineated edges in the original image, is satisfying from the perceptual point of view according to what the user was expecting (delineating the main objects in a scene for instance). Maybe such feedbacks, correlated with some other informations extracted from the image or the parts of the image, will help in defining appropriate range for setting the parameters of algorithms or even to define ranges in the image space that will allow to predict which algorithm will behave the best for the input image. Will it be finally possible to extract semantic information from a digital image?

## 6. ACKNOWLEDGMENTS

This work has been partly granted thanks to the Innovatec project funded by the DAAD (German Academic Exchange Service/ Deutscher Akademischer Austausch Dienst).

## REFERENCES

1. ImageMagick, "Image tools to read, write and manipulate an image." `http://www.imagemagick.org/`.
2. D. Paulus and H. Niemann, "Iconic–symbolic interfaces," in *Image Processing and Interchange: Implementation and Systems*, R. B. Arps and W. K. Pratt, eds., pp. 204–214, SPIE, Proceedings 1659, (San Jose, CA), 1992.
3. D. Paulus and J. Hornegger, *Applied pattern recognition: A practical introduction to image and speech processing in C++*, Advanced Studies in Computer Science, Vieweg, Braunschweig, 4 ed., 2003.
4. J. Hornegger, D. Paulus, and H. Niemann, "Probabilistic modeling in computer vision," in *Handbook of Computer Vision and Applications*, B. Jähne, P. Geiler, and H. Hauecker, eds., **2**, pp. 817–854, Academic Press, San Diego, 1999.
5. J. Hornegger and H. Niemann, "A Bayesian approach to learn and classify 3–D objects from intensity images," in *Proceedings of the* $12^{th}$ *International Conference on Pattern Recognition (ICPR)*, pp. 557–559, IEEE Computer Society Press, (Jerusalem), October 1994.
6. H. Niemann, *Pattern Analysis and Understanding*, vol. 4 of *Springer Series in Information Sciences*, Springer, Heidelberg, 1990.
7. M. Landy, "Hips-2 software for image processing: Goals and directions," *SPIE* **1964**, pp. 382–391, 1993.
8. C. Cracknell, A. C. Downton, and L. Du, "TABS, a new software framework for image processing, analysis, and understanding," in *Image Processing and its Applications*, pp. 366–370, (Dublin), 1997.
9. OpenCV, "Open computer vision library." `http://sourceforge.net/projects/opencvlibrary`.
10. A. Del Bimbo, *Visual Information Retrieval*, Morgan Kaufman Publishers, 1999.
11. L. Roberts, "Machine perception of 3-d solids," *Optical and Electro-optical Information Processing, MIT Press* , pp. 159–197, 1965.
12. R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison-Wesley, 1992.
13. L. Priese and V. Rehrmann, "On hierarchical color segmentation and applications," in *CVPR93*, pp. 633–634, 1993.
14. PUMA, "Puma, eine Programmierumgebung für die Musteranalyse." `http://www5.informatik.uni-erlangen.de/puma`.