Enhancing the Taverna Workflow System for Executing and Analyzing the Performance of Image Processing Algorithms

Sureerat Kaewkeeree[1] and Pichaya Tandayya[2]
Department of Computer Engineering,
Faculty of Engineering, Prince of Songkla University,
Hat Yai, Songkhla, Thailand
5210120104@email.psu.ac.th[1], pichaya@coe.psu.ac.th[2]

***Abstract*—In image processing, different groups of sub-algorithms give similar outputs. Therefore, there are more than one solution to process an image for a specified result. This research uses Taverna Workbench, a workflow technology, to help the developer in designing and checking the performance of algorithm sets more easily. We propose our integrated web service operation working with the Taverna workflow editor. Currently, the tool supports the image processing function based on our case study, PSU's cell image analyzer. Experiments show that using the workflow tool can help reducing the design time and the user can easily obtain the results and make decision about which algorithm sets to be used.**

Keywords-component; workflow; web service; image processing;

I. INTRODUCTION

Traditionally, in most scientific processes, to obtain the satisfactory results, the researchers had to perform an experiment through repeated cycles. Also, in image processing analysis, proper results can be achieved in various ways as there are many different algorithms to be exploited in image processing processes. For example, in the PSU cancer cell analyzer, which is our case study, there are various solutions for segmenting the cancer cells that give similar outputs. It is time and resource consuming to tediously and manually compare which is the best solution.

A workflow system can help programmers to quickly design and re-design the processing steps. In this work, we explore and enhance the Taverna Workbench [1], a workflow software for bioinformatics, to assist programmers for analyzing image processing algorithms visually and codelessly. Even non-expert programmers can create a practical workflow. The programmers can graphically control input, output and simulation variables via the data link. They do not need to be familiar with the details of the back-end system. Another key value of using Taverna is that it allows the users to add their own components which can include our provided image processing services for being processed on its workflow.

Harpia [2], a project that demonstrates a similar idea of visual interface, provides a stand-alone graphic environment that helps developing, prototyping and learning of image processing programming using OpenCV functions. Its interface is simple and user-friendly. There are many basic OpenCV functions provided for the user to assemble block diagrams. However, Harpia does not allow users to add or manage its functions. Due to the lack of new features updated, the support for complex operations is also not presented according to the latest version, Harpia 1.1.

Using the Taverna workflow system that enables the integration of distributed web services and allows addable and adaptable web services, we can provide a more flexible and scalable environment at some cost of web service communication. Our system will help the programmers to develop optimized image processing workflow diagrams and improve the designed system performance. Our case study in this paper is Cell Image Analyzer (CIA) that has been developed since 2008 [3].

II. TAVERNA WORKBENCH

With the rapid growth of e-Science technologies and applications, scientific workflow systems have been used in many specific domains. There are many examples of workflow systems. Most of them are visual programming interfaces that enable scientists to orchestrate the sequence of tasks in order to simulate their experiments and to obtain the result in real-time, e.g. Taverna Workbench, Triana [4], Kepler [5].

Triana's graphic user interface has more powerful editing capabilities for workflow composition. It came from the wave analysis environment at first and now large lengths of applications have been added. By the way, Triana is a data-flow system. The control links such as looping and conditional behaviors are not included. Kepler provides more flexibility in workflow composition. Having some incompatibility in sharing resources with others, Kepler was not widely used. Finally, we have decided to choose Taverna because of its up-to-date features, strong user and developer communities and piles of research documents and publications [6].

Taverna Workbench is an open source scientific workflow environment includes the graphic user interface based workflow composer and a workflow executive engine. The workbench consists of a set of processors which representing various kinds of software components such as web services, java classes and local scripts. The editor allows the user to

978-1-4673-1921-8/12/$31.00 ©2012 IEEE

easily create a workflow by graphically dragging elements into the workflow editor. In order to control the data flow, the processors are connected through the data links and exchange data between processors. With the advantage of visualization that easily describes working processes, mostly Internet-based services and more reusable components, Taverna has changed the way of doing scientific research.

Recently, Taverna gains popularity fast. For example, the Cancer Grid (caGrid) [7], the platform that enables sharing of the cancer research resources integrates with Taverna environment to help for analyzing information from different sources. The medical image processing on the Enable Grids for E-sciencE (EGEE) Grid [8] uses Taverna as a workflow enactment tool. In addition, there is an online community for users sharing scientific workflows and computational resources called myExperiment [9]. Since Taverna supports major bioinformatic services, the workbench is widely used in bioinformatic researches and data intensive applications.

In this work, we employ the workbench as a workflow editor in order to optimize image processing workflows. The workbench access the web service endpoint interface via Web Service Definition Language (WSDL) files which describe the image processing service descriptions.

III. PSU'S CELL IMAGE ANALYZER, THE CASE STUDY

Our case study, PSU's cell image analyzer [3], is a computer-aided system for analyzing microscopic images developed by Prince of Songkla University's research team. The system automatically counts for the number of nuclear stained breast cancer cells and works as an assistant system in pathological analysis of breast cancer. Various ways have been proposed by researchers in order to perform the segmentation of breast cancer cell images. In 2007, they presented the use of color based segmentation using back-propagation neural networks. This method shows how to separate positive and negative cancer cells from the cell image background. In 2008, a new method has been proposed for cell segmentation using K-Mean Clustering which can be used to improve the performance of previous researches. A new approach on segmenting breast cancer cells has been proposed to perform color space transformation, global thresholding method, and morphological operations in 2009. The newest method presented in 2010 used the extracted feature technique to classify the cell type [10, 11].

The above references indicate that those all proposed methods give similar results but their performances were slightly different. Analyzing these processes and comparing their performances manually take quite a long time. In addition, the breast cancer cell images used in this study are rather large and need more computing resources to produce feedbacks in real-time.

Our system provides many components of image processing algorithms based on our case study. The programmers can rearrange the block diagrams to perform the image processing procedure as they want.

IV. METHODS AND SOLUTION

A. *System overview and implementation*

Our system is implemented as a web service application. There are two representative sides which are the server side and client side. The system characteristic is described in Fig. 1.

First is the image processing services, the server role. The web server can be on the same machine as the client or a remote server that can provide better performance. We have implemented a web service server using the open source gSOAP toolkit [12] running on a 6-core CPU with 8 GB of RAM. Each of image processing service description is performed by a Common Gateway Interface (CGI) program which is deployed using the Simple Object Access Protocol (SOAP) 1.1 protocol over an HTTP web server. All of Web service interfaces are defined by WSDL documents.

Taverna Workbench, the client role, is a web service client that supports workflow design for the operational sequences and structures of processors containing image processing algorithms. The processor input and output ports correspond to the WSDL service interface which is provided by the web service server.

The WSDL file is a document describing how the image processing web service is connected to the workbench. Taverna implements each processor by a single web service operation related to the WSDL file.

As shown in Fig. 1, a programmer can arrange the processors by dragging them to construct a workflow. The data link dependently connects the output of one processor to the input of another. The planed workflow is continuously executed by the Taverna workflow execution engine.

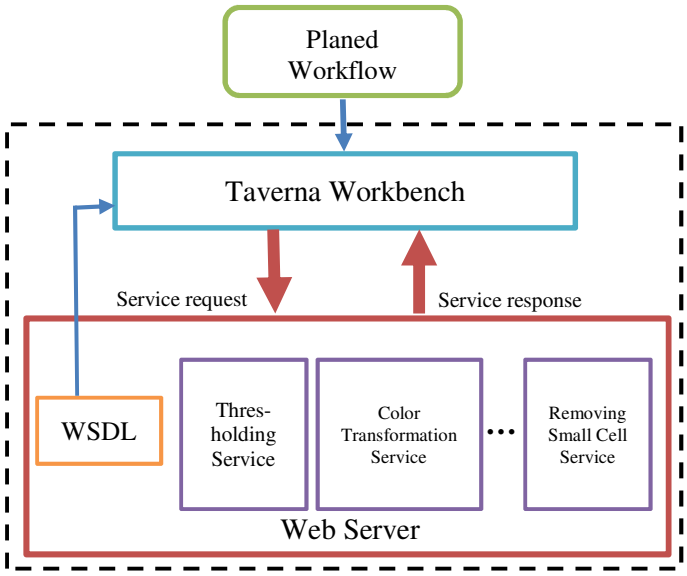

Figure 1. The system overview

Each step of the workflow will be executed at the web service server in sequence. The next processor will start executing after the previous processor has successfully completed the execution and returned response message to Taverna.

B. Image processing service operations

The image processing services in this work are developed using OpenCV library [13], a free and powerful open source computer vision library that provides a various number of real time image processing, computer vision and machine learning methods.

Our work provides some frequently used OpenCV functions for building a large range of image processing workflow, including basic matrix arithmetic, low-level image processing and image analysis. These routines are divided into two main classes: an original set of image processing functions in OpenCV and the group of the cancer cell image analysis functions based on our case study.

The set of image operating functions includes the often used image processing functions such as basic structure and its operations, image filtering, structural analysis, image transformation, and neural networks. The groups of cancer cell image analysis functions have been converted from our case study, breast cancer cell analysis application. There are many different groups of analysis operation such as removing small cells, scanning cells, separating two cells with one wall, and counting the number of positive cancer cells. Different types of processors mentioned above can be put together within a single image processing workflow.

To build image processing services, a web server must be configured. A set of web services have been designed and implemented by using the gSOAP toolkit which automatically provides WSDL files for integration with Taverna. Each service performs an OpenCV functions. Some of the image processing apparatus may need more than one OpenCV function, e.g., contour drawing, machine learning or neural network training, depending on the complexity of the algorithm. We have implemented an image loading service that converts image data into a binary format and keep it in a local storage. Therefore, the newly added service can use the same image data storage as previously used in the former service for its processing. Once the service is called, the image data will be loaded from the local memory storage, and processed by the OpenCV functions in that service and keep the image result into the local memory storage once again.

As common image processing routines are well suited for data parallelism, we have therefore implemented some parallel image processing services using the OpenMP directives [14]. In order to design the parallel execution services of image processing, we split the image data into one smaller piece per thread. The data is separately processed by each thread and then gathered into an image of the same size as it was.

This will help the programmer to design for better performance in processing the image data. The programmer can mix serial and parallel service operations together into a diagram as they want. Also, OpenCV also officially supports Intel® Threading Building Blocks (Intel TBB) which enables more parallelizable and provides better performance.

C. Building a new web service by the user

The interactive image processing service in this study is implemented as a CGI program installed on the web server, performing the image processing execution and communicating with the Taverna engine. Some of basic image processing operations are provided. In addition, the advanced user can also design new image processing services for other proposes.

In order to build an image processing service, a processing method must be defined as a function in a CGI program. The input and output data types are required for the interaction as well. In this case, using CGI-based is quite easy as it is a simple and well known mechanism. The HTTP web server will manage the requests and responses for the web services.

The programmer can use a normal image processing function by adding some relevant parameters which are used to communicate with the web server. Most of our provided services interact with the binary image data, thus, the image data transformation services and the local memory storage are provided as described in Section E. Input and output of a service operation should be image data filenames which will inform the web service engine where its data is being held.

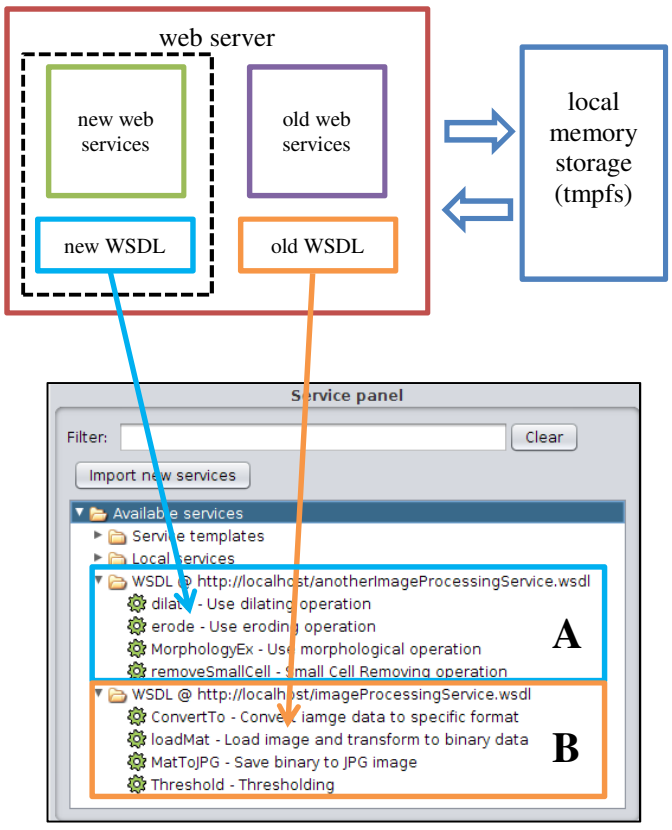

Figure 2. Adding new image processing web services

Using gSOAP to build a web service, the service definition, service namespace, location endpoint of the executable program, and other parameters must be defined in the header file. The program file contains the service operations which are implemented as functions. This procedure may be slightly different if using other tool to develop the web service.

Fig. 2 shows the overview of how to add a new image processing service to the system. Once the CGI program is deployed, the toolkit also generates a WSDL file. The programmer needs to provide the URL pointed to the WSDL for Taverna. The service operation will be available after adding the WSDL to Taverna Service Panel. Box A. in Fig.2 shows the newly added service processor which can be used together with the old service processors shown in Box B.

More details about how to build a web service application using gSOAP are described in [12]. If the programmer is not familiar with CGI, Taverna also supports other types of web services, such as REST-style web services, local java services, and external command tools.

D. The example workflow

Fig. 3 shows an example of a designed workflow following the cancer cell image processing steps based on our case study. The image processing web service operations appeared as green blocks are connected consecutively in order to form an execution flow.

The workflow input is a breast cancer tissue stained image which is used in the case study experiments. The digital image was saved in a color JPEG format with the resolution of 2,560 ×3,600. The data links show relationships amongst processors.

Once image data is sent to be processed at the web service server, the first operation will prepare the data to be used in the next operation by converting a JPG image into a matrix data which is one of the employed image structures that can be used as an input of any service operation that processes images.

The workflow designer can add new parameters to any processor. Default parameters can also be provided when the user does not set the parameters. We also validate input parameters to ensure the correctness due to the high complexity and restriction in coding with OpenCV in order to enable that sequentially linked components can well communicate with each other.

From the example workflow in Fig. 3, by default, the engine invokes the services in parallel if there are no data dependencies between the services. The result is collected to the server after completing each service.

According to the case study, researchers had tediously spent much of experiment time to investigate each part of cell processing steps finding which are proper techniques for the cell counting system. Our listed component group of image processing functions enables that researchers can easily arrange the processing step. It clearly reduces the whole design time for the case study project.

E. Image data storage problem

This kind of storage problem occurs when implementing the image processing algorithms as web services which are considered as complete programs by the operating system. The common approach of image processing application uses the existing image data to perform the execution of each different function in a pipeline. When the processing steps are like that it looks contradictory to the characteristic of web services as each of them are independent applications. After a service is completely executed, the program exits and all of its allocate memory is completely deleted. It is the problem of unnecessary repeating memory relocation; loading a program, allocate its memory, release the memory and then reallocate it again every time the web service is called. Therefore, the data management problem is prominent.

In the web service technology, there are many interesting ways to transfer binaries amongst web services, for example, using Base64 encoding to exchange binary contents which has a great interoperability with Extensible Markup Language (XML) standard. However, data encoded in Base64 is not efficient because the Base64 encoding roughly increases file sizes by 33% larger than their binary format. Another method is embedding Message Transmission Optimization Mechanism (MTOM), the binary data in the SOAP envelope. This method uses a smaller size of the data exchange than does the Base64 encoding method since data is pure binary. The other method uses the File Transfer Protocol (FTP) server or shared location and keeps only the reference to the binary data. Since the image size is quite large, we choose to keep the image data at the server and exchange only the file reference. In order to view the result image at the workbench, the MTOM method has been implemented.

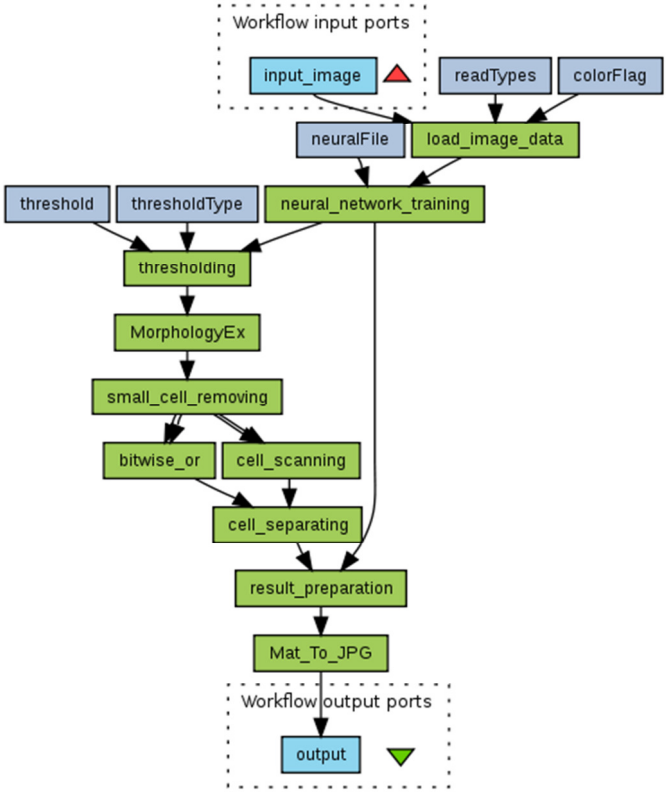

Figure 3. The example of cancer cell analysis workflow

According to reduce the execution time and storage resources when processing the workflow of image processes, the Temporary File System (tmpfs) [15] is used as an image data storage area. This is because reading and writing data in the memory is faster than to do so on storage disks. The tmpfs allowed the user to store all files in virtual memory. File access in this case only causes a memory-to-memory copy of data; no files will be created on the hard drive. However, as the files are stored in RAM, they will be completely deleted if the user unmounted the partition or shut the device down.

In addition, the image's binary data is saved to tmpfs instead of an image file format stored in the file system in order to avoid losing the image quality and reduce the access time to the I/O. Depending on different system behaviors, developers can choose either the tmpfs option or the MTOM standard method which are also provided.

V. RESULTS AND DISCUSSION

In this work, we provide various image processing methods as available web services to be added in a Taverna workflow as shown in Table 1.

Our work provides two types of image processing services which can be used to compose workflows in Taverna as described in Section B. The first group is a set of original functions in the OpenCV library which are shown in the first column in Table 1. Another column shows a group of functions for the cancer cell image processing based on the case study. Each type has different characteristics which are suitable for different purposes. The pro of the first group is that the developers can easily create workflows from the OpenCV library. It provides more flexibility but the developers must have a know-how of the OpenCV library. In addition, due to the complexity of the OpenCV algorithms and the web service characteristics, deep adaptive control of the code may not be possible.

On the other hand, using the second group of web service operations will be easier because there is no need to worry about the functions within the service. This is useful for the developers who do not deeply know the OpenCV library.

TABLE I. OUR AVAILABLE IMAGE PROCESSING WEB SERVICE OPERATIONS

Operations from OpenCV Functions	**Operations for the case study**
- Arithmetic and logical operations such as sum, subtraction, multiplication, division, bitwise not, bitwise or, bitwise and bitwise xor. - General functionality such as loading, saving and displaying images. - Image transformation such as color conversion, thresholding and watersheding. - Image filtering operations such as dilation, erosion and morphology. - Structural analysis such as contour detection and contour area calculation. - Feature detection such as canny operation. - Machine learning such as neural networks.	- Small cell removing - Cell scanning - Cell separation - Result preparation

However, they may get low flexibility in designing the workflows because the algorithms are fixed within the web service operations. The results have been validated as our web service applications gave the same results as non-web service applications from the case study. Our work provides some basic of cell image processing operations. However, new operations can later be easily added.

Fig. 4 displays the integration of our image web service operations to the Taverna Workbench.The left panel (A) shows the image processing service provided by adding the WSDL file to the workbench. Programmers can visually drag and drop services to the graph layout view (B) in order to design a workflow. After running the workflow, the results will be shown in the invocation window (C). Taverna also shows the execution time of the whole workflow. The result of the workflow shows an image result with different colors of positive and negative cancer cells. In this case, red indicates positive cells and green indicates negative cells.

The major advantage of our image processing workflow builder is the ability to select the available image processing processors, thus the workflow designer can selectively address, move, delete or control the flow of the workflow. The results can be obtained in any part of a processor. Therefore, with this method, programmers can more easily compare the results than the old method. Thus, using this workflow system can obviously enable the programmer to quickly design the processing steps of image processing; especially in this case study, the cancer cell image analysis.

As we mentioned above, using the workflow techniques can assist the programmer to reduce the design time and complexity. However, to analyze which is the proper workflow for the application is the task of the programmer because the result satisfaction depends on the background knowledge of the workflow designer and the case specification.

Work is ongoing to provide more service operations to support the image processing workflow environment since there are still more valuable functions from the OpenCV library. Moreover, this workflow technology enables sharing services with other applications across multidiscipline. Therefore, it is possible to adopt this work for more general and wider usage.

VI. CONCLUSION

Applying the workflow technology, we present an easy solution to design image processing steps illustrated by using the cancer cell analysis as a case study. We have successfully provided a web service server consists of image processing algorithms which are sets of image operating functions in OpenCV library and groups of image processing function depending on specific needs based on PSU's Cancer Cells Analyzer application. The image processing services allow the programmer to arrange them in order to perform the desired workflow of image processing in the workflow editor. The programmer can more easily choose, combine and reorganize any of the provided image service operations. This can help them to quickly design and tune up the image processing flow. Then, the programmer can decide which workflow gives better performance. In addition, local data storage has been used to

improve the data exchange problem between the server side and the client side. Also, the programmer can easily and quickly add new processing services on the web server following our guideline. New web services can be easily added. The workflow can include distributed and parallel web services for performance enhancement and system scalability. Memory management options such as tmpfs and MTOM are also provided for faster execution on a standalone machine.

ACKNOWLEDGMENT

This work has been funded by Department of Computer Engineering, Faculty of Engineering, Prince of Songkla University. The authors are grateful for Dr. Somchai Limsiroratana's advices and contribution on the image processing source code and data in the case study.

REFERENCES

[1] T. Oinn et al., "Taverna: Lessons in Creating a Workflow Environment for the Life Sciences," *Concurrency and Computation: Practice and Experience*, vol. 18, no. 10, 2006, pp. 1067-1100.

[2] S2i - Industrial Intelligent Systems. (2009, July). *Harpia Project* [Online]. Available : http://s2i.das.ufsc.br/harpia/en/home.html

[3] P. Phukpattaranont et al., "Computer-Aided System for Microscopic Images: Application to Breast Cancer Nuclei Counting" in *Int. Nat. Applied Biomedical Engineering,* vol. 2, no. 1, 2009, pp. 69-74.

[4] I. Altintas et al., "Kepler: an extensible system for design and execution of scientific workflows," in *Proc. 16th Int. Conf. Scientific and Statistical Database Management*, 2004, pp. 423-424

[5] S. Majithia et al., "Triana: a graphical Web service composition and execution toolkit," in *Proc. IEEE Int. Conf. Web Services*, 2004, pp. 514-521.

[6] E. Deelman et al., "Workflows and e-Science: An overview of workflow system features and capabilities," *Future Generation Computer Systems*, vol. 25, no. 5, pp. 528–540, 2009.

[7] W. Tan et al., "CaGrid Workflow Toolkit: A taverna based workflow tool for cancer grid," *BM*, vol. 11, p. 542, 2010.

[8] K. Maheshwari, P. Missier, C. Goble, and J. Montagnat, "Medical Image Processing Workflow Support on the EGEE Grid with Taverna," in *Intl Symp. Computer Based Medical Systems*, Albuquerque, New Mexico, USA, 2009.

[9] D. D. Roure, C. Goble, and R. Stevens, "The design and realisation of the Virtual Research Environment for social sharing of workflows," *Future Generation Computer Systems*, vol. 25, no. 5, pp. 561-567, 2009.

[10] P. Phukpattaranont and P. Boonyaphiphat, "Color based segmentation of nuclear stained breast cancer cell images," in *Proc. ECTI Transaction on Electrical Eng*., vol. 5, no. 2, 2007, pp. 158-164.

[11] P. Phukpattaranont and P. Boonyaphiphat, "Computer-aided analysis of nuclear stained breast cancer cell images," in *Proc. ECTI Transaction on Electrical Eng*., vol. 1, 2008, pp. 485 –488.

[12] R. A. van Engelen and K. Gallivan, "The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks", in *2nd IEEE Int. Symp. Cluster Computing and the Grid*, Berlin, Germany, 2002, pp. 128-135.

[13] G. Bradski and A. Kaeller, Learning OpenCV—Computer Vision with the OpenCV Library, O' Reilley Media, 2008.

[14] L. Dagum and R. Menon, "OpenMP: An industry-standard API for shared-memory programming", *IEEE Computational Science and Engineering*, vol. 5, 1998, pp. 46-55.

[15] P. Snyder, "tmpfs: A virtual memory file system," In *Proc. EUUG Conf.*, 1990, pp. 241–248.

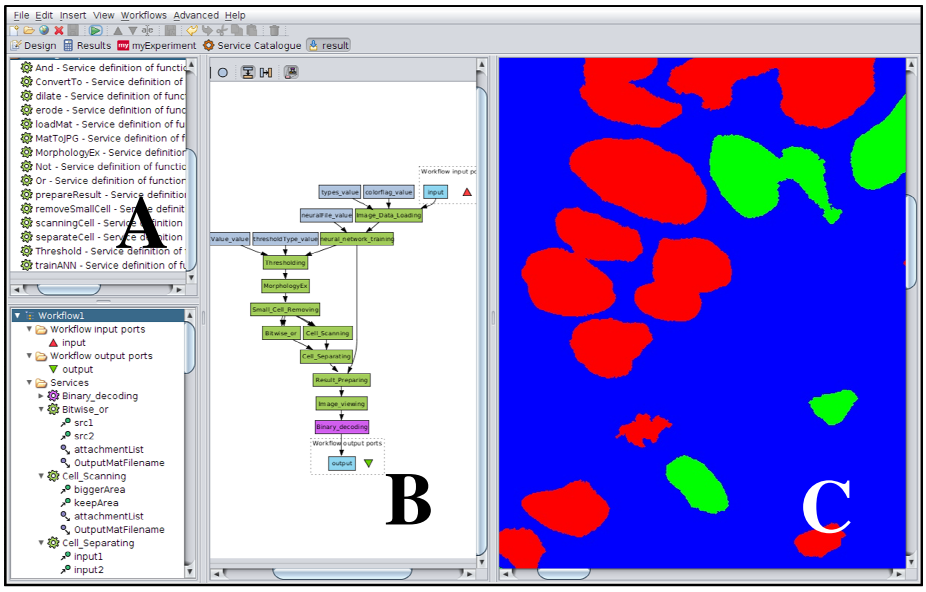

Figure 4. The Taverna workflow editor with a list of our available image processing service operations