ELEKTROTEHNIŠKI VESTNIK 78(1-2): 18–23, 2011
ENGLISH EDITION

Transforming computer-vision applications into useful web services

Bojan Kverh

University of Ljubljana, Faculty of Computer and Information Science, Tržaška 25, Ljubljana
E-mail: bojan.kverh@fri.uni-lj.si

Abstract. Web services are applications, offering their service to different clients on the web. In this paper we will explain how to transform a standalone desktop computer vision application into a web service so that minimal, if any, changes in the application will be required. In this way we completely avoid all the trouble, arising from trying to distribute the application to different operating systems with different configurations. As computer-vision applications tend to be computationally expensive, they are usually not written in managed programming languages, like Java or C#. Our implementation runs on the Linux operating system with the Apache web server and uses the PHP scripting language and the NuSoap library. The latter allows for an easy implementation of web-service functions and a client, using web-service functionality.

Key words: web services, web applications, computer vision

1 INTRODUCTION

Web services [3] are internet applications, that offer their services to different users. They are used in a variety of different areas [7] and enable using their services in different dynamic web pages and other internet applications. In this paper, we will address the possibility of using web services for transformation of computer-vision algorithms into internet applications. We will focus on transforming the existing desktop applications in a way requiring as few changes in the existing source code as possible. Furthermore, the services of the existing destop computer vision applications will be made available for being used in different web pages.

Computer-vision applications are usually created as standalone desktop applications, that can be run on the operating system, they have been created for. The main reason for this is that these applications are computationally expensive and executing them in virtual machines (like Java for example) would even further slow them down. If we want to offer these applications to other users, we may encounter the problem of distributing them. The application can be offered in the form of an executable program, but this is limited to the operating system and the environment (libraries) where the application was compiled. It is possible to compile the application for a variety of different systems and offer a compiled version for each of them however, this requires a lot of effort in both distribution and mantainance. Distributing the application source code is much easier, but it can also be much more frustrating to the user, for being left by himself with the challenge to compile the application on his particular system. The trouble can be avoided by using the approach developed by Skočaj [10]. He extends the application with the CGI functionality over which the application communicates with an HTTP server. To do this, it is necessary to adapt the application for receiving the data, commands and presentation of the obtained results. Also, this kind of application is only available on servers, with the application itself physically present. Since the time Skočaj presented this approach, internet has evolved drastically; new technologies have been made available allowing transforming standalone desktop applications into web applications with less effort. With the evolution of computer hardware over the years, it is also more feasible to use different scripting languages presenting the basis for different internet applications, dynamic web pages and web services.

The web service technologies developed in the last few years present the opportunity to offer their services to anyone interested in using them. These services allow us to use the functionality implemented on one particular server on any other web page in the way we desire. The rest of this paper will describe a particular transformation of a standalone application into the web-service with all the details needed to take care of when implementing the transformation.

Received 12 October 2010
Accepted 15 November 2010

2 WEB SERVICE AND ITS CLIENTS

From the dynamic web-page developer point of view, the web service is just a set of functions performing certain tasks or providing certain information. These functions can take parameters defining our requests in more detail and returning some results. Clients and web services, running on a server usually communicate via the SOAP protocol [6], which is based on the XML language.

In our particular case, after receiving a request from a client, our web service will execute a standalone application on the server which will perform the required operations and save the results. The results will be sent back to the client after the standalone application is terminated. In our opinion, the easiest way to achieve this kind of behavior is to adapt the application to react to a set of different parameters passed to it from the shell command. By using these parameters the operations to be executed by the application must be specified. Some applications, specially those with a graphical user interface, cannot be controlled by the command shell parameters. However, adapting the application to this kind of control, is usually fairly simple. Next, the application needs to be able to save the obtained results in a particular form, but most of the applications already have this kind of functionality implemented.

The client using the web service can be any dynamic web page or internet application. It has to make sure that it calls the right web-service function with suitable parameters and that it allows enough time for the service to come up with the final results. If the latter requirement is not fulfilled, we could face the “Server did not respond in time” error.

3 IMPLEMENTATION

Our web-service was implemented using the PHP scripting language and the NuSOAP library [4] on the Linux operating system, running the Apache web server. Linux and Apache were chosen for their reliability at serving web pages and their leading role in the world of servers. PHP is a very popular scripting language for dynamic web-page development, while the NuSOAP library simplifies creating web services and their clients.

3.1 Web services

Web service implementation using the NuSOAP library must act on a particular request from clients. It has to recognize the function that the client wants to be executed and transform the request into a shell command which will then execute a standalone application. The application must read the command-line parameters, execute proper operations, save the results to a hard disk and terminate. After its termination, the web service reads the saved results from the disk and passes them as the result of a function call to the client. The entire system is shown in Fig 1

The NuSOAP library allows easy creation of web-services, as illustrated in the PHP code below:

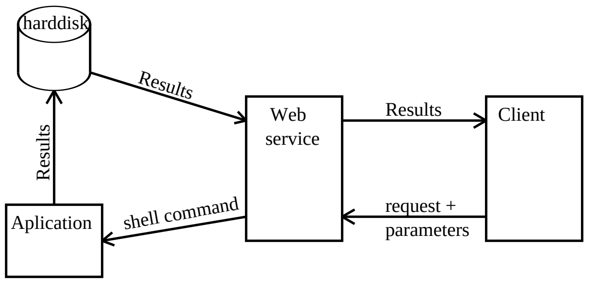


Figure 1. Standalone application as web service.

```
require_once(``lib/nusoap.php'');
$server = new nusoap_server;
$server->configureWSDL(``segmentWS'');
$server->register("segment",
  array(``x'' => ``xsd:int'',
  ``y'' => ``xsd:int''),
  array(``result'' => ``xsd:string''));
$server->service($HTTP_RAW_POST_DATA);
```

This piece of code first makes sure that proper library files are included and made available for the rest of the code to use their classes. Then it creates an instance of class `nusoap_server`. The third line of the code enables the clients to receive the description of all the functions and their parameters on request in a special WSDL language (Web Service Description Language). The fourth line of the code contains registration of the web-service function made available to the clients. The function name, parameters and the form of the result to be returned to the client have to be specified. In the last line of the code, there is a call of the function that receives and recognizes the client requests and executes proper functions. Besides, we have to implement a function (in our case it is the function “segment”), with all the parameters specified at registerating it as one of the web service functions:

```
function segment($x, $y)
{ ...
  }
```

This function has to be implemented in the way enabling it to execute a standalone application that will perform the requested operations. The NuSOAP library makes sure that the returned results are all transformed into the SOAP form and sent back to the client.

There is a special issue when executing applications with a graphical user interface or showing the results in a graphical form. Namely, in a Linux environment, the web applications and services are usually executed under special system user `apache` which, however, is not permitted to execute graphical user interface applications for security reasons. We can resolve this issue by using the VNC server [2]. This can be executed by the `apache` user and attach it to one of the virtual screens of the graphical desktop. In the web service function, we have to change the line:

```
exec($command);
```

which executes the standalone application, into

```
exec(``export DISPLAY=''.$dsp.``; ''
.$command);
```

Before doing this, `$dsp` variable is assigned the number of virtual screen used by the VNC server. In this case, the application is executed in a particular virtual screen instead in a usual graphical desktop screen. Variable `$command` has to consist of the path to the application, application executable name and eventually also the parameters, controling execution of operations. Thus, the application must be able to read the command-line parameters and execute accordingly. This is also the only part where we may need to upgrade the original standalone application when transforming it to the web service.

4 Client

Implementation of clients using the web services, is also very simple by using the NuSOAP library. An example is presented in the code below:

```
require_once(``lib/nusoap.php'');
set_time_limit(1200);
$client = new nusoap_client($url.'?wsdl',
  true, false, false, false, false,
  1200, 1200);
$result = $client->call("segment",
  array(``x'' => ``9'', ``y" => ``6''));
echo '<br/><img src=``'.$result.''' />';
```

After mandatory inclusion of the NuSOAP classes and setting up the maximum allowed time of execution in the first two lines of the code, there is a creation of an instance of the `nusoap_client` class. We must pass the URL address of the web server (specified in variable `$url`) as the first parameter to the constructor of this class. With the second parameter set to `true`, we request the description of the service and its functions in the WSDL language. This is followed by a few more or less important parameters, whose roles can be found in [4], while the 7th and 8th parameter specify the maximum allowed connection time and maximum allowed time for the web service to respond. In our particular example they are both set to 1200 seconds. This can be changed in order to accomodate the service function to take more time to execute. In the next line of the code there is a call to the web service function by invoking the "call" method of the `nusoap_client` class. This method takes two parameters. The first has to name the web service function we want to execute. The second should be an associative array containing all the parameter names and their values needed for the specified web service function. The "call" method returns the result it received from the web service function which can be used to create a dynamic web page. In our particular example, we assume that the service returns the URL address of the image containing the final results, to show this image on the web page using the HTML tag `img`. However, this is not the only possible way of presenting the results. The web service can generate any other kind of data as a result of the processing. In this case, we can offer the user a "Download" button in the web page to allow him to download the generated data by clicking on it. Another positive side of this kind of implementation is that we can tailor the usage of web services to particular user needs. If the web services offers a large amount of parameters to configure, which is not unusual in the computer vision field, the dynamic page programmers can make a decision as to which parameters are better to be configured already in the code and which can be left to the users for experimentation. The user interface of the client application can be much more user-friendly with this approach.

5 File transfer

Applications are not particularly useful if they do not offer their users the possibility to process their own data, images, files, etc. The same is true for web services. Processing data being in an appropriate format by using some application enables us to compare its results with other applications of the same type and figure out which one is the best in terms of quality or even the processing time.

Transfering a file to a web server is usually done with special forms on web pages, in which the user specifies the file at his local disk and transfers the file by clicking on the submittion button. We suggest the same kind of approach even in the case of using web services to process a particular user file. However, this will only upload the file onto a server where the client application, using the web service, is executed. So, when using web services, we have to transfer the file also from the client to the web service, which can be running on a different server than the client application. One of the possible ways to achieve this is that the web-service function is implemented so that it receives the file in one of its parameters, while the client application reads the file content into a variable and passes it to the web-service function as a parameter. The web service then saves the file content to its local disk and executes the application which processes the saved data. At both the client and the web service we have to set up file write permissions in a way enabling the files to be saved in particular places. When passing the file content as a parameter to a web service function, it is advised to use the "xsd:string" type for text files and "xsd::hexBinary" or "xsd::base64Binary" for binary files.

5.1 Concurrent usage of web services

Before the application that has been turned into a web service is offered for general usage, we have to resolve two minor issues. If the user reloads the page, when it is waiting for the web service to finish the processing, the page will again request the same processing from the web service. Two instances of the standalone application will be running as a result of the reloading, with one of them not being needed anymore. Thus we have to provide a mechanism to remove the not needed instance of the application as it still uses computer resources. When the web service is in a general use, there can be two instances of the application running, but each of them started by a different user. In this case, however, no instance should be removed. A possible solution to this issue is that the web service requires unique authentication of qither the IP address or the username and keeps a list of who has executed particular instances of application via web service. Before the web service executes an instance of application on a request from a particular user, it has to remove all the instances, initiated by the same user in the past, if they are still running. On the Linux systems, every executed instance of the application is assigned a unique number, i.e. PID, which stands for the process identifier. If we want to find out the PID of the application, we have just started, we need to execute the following code:

```
$com = ``export DISPLAY=:''.$dsp.``;
nasUkaz > /dev/null 2>&1 & echo $!'';
exec($com, $output);
$pid = (int)$output[0];
```

Instead of `ourCommand`, we have to input the actual shell command which starts the standalone application with appropriate parameters. The following part outputs the PID of the initiated process which is then read into variable `$pid`.However, the command `exec` returns the control to the operating system immediately, and not after the application finished with its processing. This allows the web service function to terminate before the standalone application provides the final results. Thus, we have ti check periodically whether the application has indeed finished by using the shell command `ps`. We can do this as follows:

```
$running = 1;
while ($running > 0)
{ $cmd = ``ps ''.$pid;
  exec($cmd, $output);
  if (count($output) >= 2) sleep(1);
    else $running = 0;
  unset($output);
  }
```

Application PID is passed as a parameter to shell command `ps`. If execution of this command outputs at least two lines to the standard output written in the variable `$output`, it means that the application is still running. When the script detects that the application is not running anymore, the `while` loop can be terminated. User identifications together with PID of the application instance that was initiated by the user are saved into a specific database table. Before this, the web service has to find any PID from the same table which is tied to the current user identification and remove the application instances with the following command:

```
exec(``kill -9 ''.$pid);
```

The second issue it the file preservation. The uploaded files and files with the obtained results belonging to one user must not overwrite those that belong to another user. This could happen if two files from different users had the same names. However, this issue can also be resolved by requiring authentication from the user invoking the web servies and by renaming the files so that their names include the user identification.

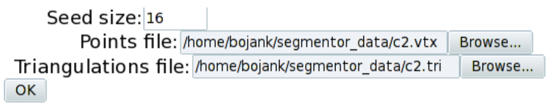
In case of a concurrent usage of the web servie from several users, it is necessary to monitor the system resources usage so that the processor or processors are not too busy executing the web-service application. If this happens, accessing the services would become very time consuming or even impossible. With specific measurements, we can determine what is the maximum reasonable number of application instances running on the system. When the number of application instances, which we can determine from the database table describe before, the web service can refuse to do the required processing. Instead, it should return a message informing the user that the maximum capacity of the system has been reached and that he should try to use the service later.

6 SEGMENTOR

Segmentor [9] is an application segmenting 3D range images into consistent regions using the Recover-and-Select paradigm [13], [12]. The main feature of this algorithm is that every region is described with a parametric model. Region growing depends on the distances of points in the region neighbourhood from the parametric model. The points close enough tothe model are included into the region and new model parameters are calculated from the points from the extended region. After the predescribed number of the region growing steps, selections are performed, retaining only the regions necessary to cover the entire 3D range image. Since the regions can overlap with each other after growing, selections tend to reject the unnecessary ones in order to speed up the growing process. Selections are based on the MDL principle [14] ensuring the description of the range image in terms of the paramteric models and point residuals to be as short as possible. In our case, the

Segmentor uses superquadrics [9], [11] as parametric models, but other models like planes, spheres, cylinders, cones and tori are also available. For each used model there has to be an algorithm to calculate the the optimal model parameter values from a given set of 3D points. Optimal in this case means that the sum of squared point residuals is minimal.

We chose this application as an example of turning it into the web service. We created a web page with a form where the user can enter three parameters, two of them being files representing the range image in the form of triangulated data [8]. When we fill the form with data as shown in 2, we click on the "OK" button and start the segmentation process. The selected files



Seed size:	16	
Points file:	/home/bojank/segmentor_data/c2.vtx	Browse...
Triangulations file:	/home/bojank/segmentor_data/c2.tri	Browse...
OK		

Figure 2. Web form to fill in parameters and filenames

are uploaded from the user's hard drive onto the client's one and the client web application reads the files content into variables and invokes the web service function using these variables and the form parameter of the initial seed size as parameters. The web service saves the files content on its hard drive and starts the Segmentor application with adequate parameters. The application performs segmentation and saves the original 3D image and final segmentation results on the hard drive and the web service returns the client the URL address where these two images can be accessed. The client shows the images on the web page as shown in 3.

As a standalone application, the Segmentor allows the user to set up a large number of parameters. To set them up correctly, it is necessary to know the process of segmentation. Administrators of web pages using the Segmentor as the web service can make a decision about which parameters will have a predefined value and which ones could be adjusted by users. Using this approach, the users are not faced with the vast complexity of standalone application and testing it via the web service can be much easier. On the negative side, advanced users knowing the standalone application details may be unable to experiment with lots of parameters as they may be used to them by using the standalone application.

7 Conclusion

In this paper we presented a simple way to turn standalone applications into web services in the way enabling their functionality to be used in different dynamic web pages. The implementation, which is based on Linux, Apache, PHP and NuSOAP library is easy and

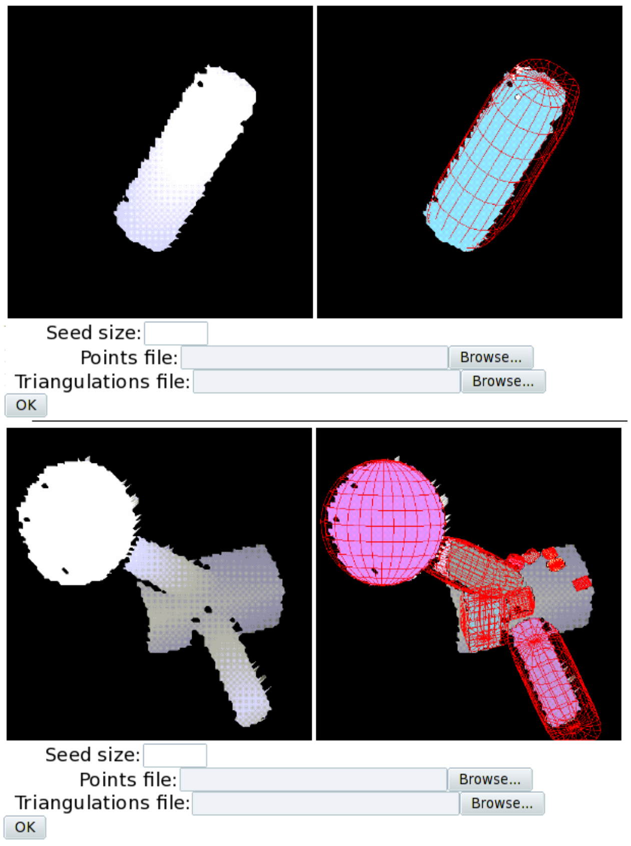


Figure 3. Two examples of results obtained with the standalone application turned into web service

elegant. The only requirement when turning a standalone application into a web service is that the application execution can be driven by shell parameters at the start up. In such case, the application can be executed from a PHP script in which the web service is implemented. Ideas presented in this paper were use to offer the functionality of the standalone application Segmentor as a web service, with which we can perform segmentation of the 3D range images in the form of triangulation. Web page [1] uses this serviceand allows us to upload our own range images, thus making it much more useful. The web services offer an elegant way to use standalone applications on systems for which they have never been planned to be used.

References

[1] *http://amanda.fri.uni-lj.si/segmentorWS/segClient/client.php*.
[2] *http://en.wikipedia.org/wiki/Virtual_Network_Computing*.
[3] *http://en.wikipedia.org/wiki/Web_service*.
[4] *http://sourceforge.net/projects/nusoap/*.
[5] *http://www.w3.org/TR/wsdl*.
[6] *http://www.w3schools.com/soap/default.asp*.
[7] *http://www.webservicelist.com/*.
[8] A. Leonardis B. Kverh, A. Jaklič and F. Solina. Using recover-and-select paradigm on triangulated data. In B. Zajc, editor,

Zbornik šeste Elektrotehniške in računalniške konference ERK '97, 1997.

[9] A. Jaklič. Segmentor: An object-oriented framework for image segmentation. Technical report, Computer Vision Laboratory, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 1996.

[10] A. Leonardis D. Skočaj, A. Jaklič and F. Solina. Testing computer vision algorithms over world wide web. In *Proceedings of the 21st Workshop of the Austrian Association for Pattern Recognition (OAGM/AAPR)*, 1997.

[11] A. Jaklič, A. Leonardis, and F. Solina. *Segmentation and Recovery of Superquadrics*, volume 20 of *Computational imaging and vision*. Kluwer, Dordrecth, 2000. ISBN 0-7923-6601-8.

[12] B. Kverh. Izboljšava klasičnega postopka gradnje in izbire za segmentacijo podatkov. *Elektrotehniški vestnik*, 2003.

[13] A. Leonardis. *Image Analysis Using Parametric Models*. PhD thesis, Fakulteta za elektrotehniko in računalništvo, Univerza v Ljubljani, 1993.

[14] J. Rissanen. Modelling by shortest data description. *Automatica*, (14):468–471, 1978.

Bojan Kverh received his B.Sc., M.Sc. and Ph.D. degrees in computer and information science from the University of Ljubljana in 1995, 1998 and 2001, respectively. Since 1995 he has been working at the Computer-Vision Laboratory at the Faculty of Computer and Information Science in Ljubljana. He has participated in several scientific and industrial projects. His research interests include 3D reconstruction of range images, different criteria for model selection, web applications and client-server architecture.