Wilhelm Burger, Mark Burge (eds.)

# Pattern Recognition 1997

Proceedings of the 21st Workshop of the
*Austrian Association for Pattern Recognition*
(ÖAGM/AAPR)
Hallstatt, Upper Austria, May 26-27, 1997

R.Oldenbourg Wien München 1997

## Applications

# Testing Computer Vision Algorithms Over World Wide Web*

Danijel Skočaj, Aleš Jaklič, Aleš Leonardis, Franc Solina

Computer Vision Laboratory, Faculty of Computer and Information Science, University of Ljubljana
Tržaška 25, 1001 Ljubljana, Slovenia, tel: +386 61 1768 381, fax: +386 61 1264 647
e-mail: danijels@razor.fer.uni-lj.si

**Abstract:** *In this paper we explore different possibilities of using the Internet for making algorithms publicly available. We describe how to build an interactive client/server application which uses World Wide Web for communication. The client program is a Java applet. The server program works on the server as a CGI program which is started by the HTTP server on the demand of the client. The data transfered between the client and the server program passes also through the HTTP server as the HTTP protocol is used for data transfer. A stand-alone program for image segmentation was transformed into the Java-client/CGI-server application, which can now be used as a service on the World Wide Web.*

## 1 Introduction

Researchers in the field of computer vision are working on different computers using various operating systems. Therefore it is very difficult to make the software available to the others in a convenient way because of different platforms being used.

Fortunately, we have the Internet, which enables to communicate between different platforms. Researchers can exchange messages, discuss problems in discussion groups, and exchange even programs and program code. But there still remains a problem - how to execute these programs on different platforms. Compiling a source code is just a partial solution. We need a platform-independent code to solve this problem completely.

Internet service World Wide Web offers a solution – Java programming language. A Java byte code is completely platform independent and portable, and it can be interpreted by the web browsers on all the main operating systems. We can write our algorithms in Java or we can write in Java only the interface to our program which is executed on our computer. In this way we make possible to test our algorithms remotly from different platforms.

In this paper we explore different possibilities of using the Internet for making the researcher's work publicly available. We describe how to build an interactive client/server application, where the client is a Java applet and the sever is a CGI (Common Gateway Interface [1]) program. This technology is implemented on an example of a program for image segmentation, which can now be used as a service on the World Wide Web.

## 2 Computer vision software on the Internet

We can make our algorithms available on the Internet in several ways [2].

* This work was supported by the Ministry of Science and Technology of Republic of Slovenia (Projects J2-6187, J2-8829, L2-8721), European Union Copernicus Program (Grant 1068 RECCAD), and by U.S. - Slovene Joint Board (Project #95-158).

#### Distribution of the source code

Distributing the source code by putting it on an anonymous FTP server is the most common use of the Internet for making algorithms publicly available. Everyone can download this source code, compile it, run it, and test it. But while compiling the source code, there often appear problems with paths, libraries, and platform specific functions. It is possible to achieve a high level of code portability with careful programming but the program might still not run on every platform and on some platforms may run only after significant extra efforts of the users.

#### Algorithms in Java

If we want to have a fully platform-independent program, we must write it in the platform-independent and portable programming language, such as Java [3]. A user can obtain and use such program in very easy way. His web browser downloads a Java applet together with the web page and starts it. There is no need for installation.

But Java has some disadvantages such as being slow and having restrictions due to security reasons. Therefore at the moment, Java is a good choice for implementing short, computationally inexpensive algorithms, which do not need an access to the file system.

#### Sending requests and waiting for the response

A remote user can over the Internet just post requests and parameters (images, commands) to us. We then pass this data to the program which processes them. We inform the user to download the results when the work is done. This method is not very practical to use, thus we should try to automate the entire process.

#### Sending requests and getting results with forms

World Wide Web can also be used for sending requests and getting results. First, a user fills a form with the data. These data are then submitted to the program on the host, which processes the data, creates the results in HTML format and sends them back to the user's computer, where they are displayed in the web browser.

This method is most suitable for algorithms which require input data only once, process them and return the results also only once. In other words, it is suitable for the algorithms that do not require higher level of interactivity.

#### Interactive client/server applications

We can use Java programming language to create a client/server application with full interactivity [4]. In this case, we write the client part of the application as a Java applet. This applet serves as an interface between a user and the computer program which is executed as a sever program on the host machine. This method requires more efforts for implementation, but as a result we get a client/server application with a two-way communication.

In the next section we describe an implementation of this method.

## 3 Interactive client/server applications on the WWW

The main characteristic of a client/server application is that a part of the application (client) requests the execution of some job, while the other part of the application (server) executes this job [5].

The basic idea is that we write the client program as a Java applet and we modify our computer vision program into CGI program. This program works as a server program and performs all the computationally expensive operations. The execution of Java applet is not very fast, but since the client program is small and it does not contain computationally expensive functions, its execution is fast enough.

We use WWW and its HTTP protocol [6] to distribute the client program (Java applet) to the user. The server program is started by the HTTP server on the demand of the client. HTTP server on the host is used as an interface between the server and the client program. Figure 1 depicts the course of communication.

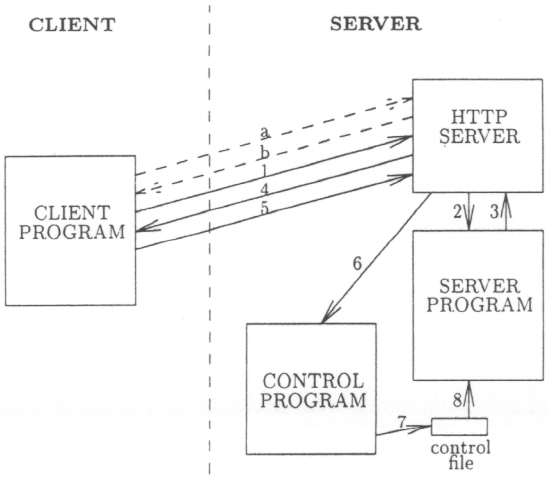



Figure 1: Java-client/CGI-server communication model.

At the beginning the user requests a web page with client program from the host (a) and he gets it (b). The client program, run by the browser, then sends the request to the HTTP server (1) to start a CGI program which is a server program of our application. The HTTP server executes the request (2). The server program begins by informing the HTTP server that it will be sending an answer in multiple blocks using the Server push method (3) [7]. The HTTP server then informs the client that the server program is ready for further communication (4). This connection remains active as a one-way communication channel from the server to the client. The user can then through the HTTP server (5) start the parallel CGI program and send it a request. When this CGI program receives the request (6), writes it into a special control file (7) from where the server program reads it (8). This is a one-way communication channel from the client to the server. The server program than executes the request and sends the answers through the HTTP server (3) to the client (4). In this way the two-way communication channel between the client and the server is established.

While transforming a stand-alone program to a client/server application, the input and output of the program have to be moved to the client program. A schematic data-flow in a client/server application is depicted in Figure 2. A user enters the requests into the client program. The client program then sends these requests to the server according to the communication protocol. These requests are the input to the server program. The server program processes the data, creates the results and sends them to the client. The client program shows them to the user as the final output of the client/server application.

All that has to be done to transform a stand-alone program into a CGI server program is to modify the input and output of the program. The CGI server program reads the input data from the control file instead of from the keyboard or some other input device, so the stand-alone program must be modified to support this. The same must be done for the output. All output commands have to be replaced with the new ones which send the output data to the client in the corresponding format.

We collected all the new commands in C++ class *wwwInterface*. This is an interface between a CGI server program and an HTTP server. *wwwInterface* takes care of the communication in the server program. It has a corresponding partner in the client program which understands its language - *WWWinterface* class in Java.

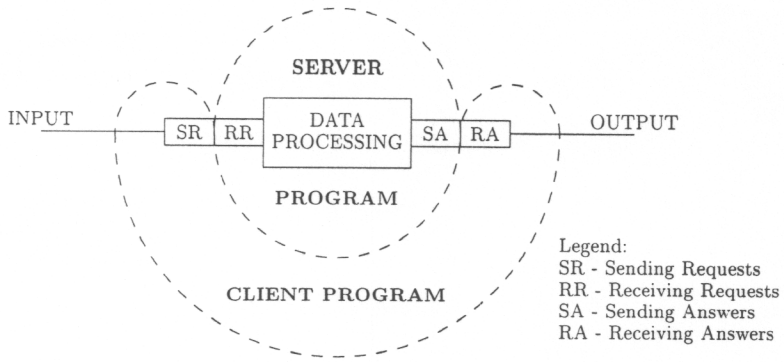



Figure 2: Data-flow in a client/server application

Writing a Java-client/CGI-server application is much easier if we use a pair of *wwwinterface* interfaces. They handle the communication between both parts of the application. We only have to include *wwwInterface* in a stand-alone program and move the input and output functions to the Java client program.

## 4 On-line Segmentor

On-line Segmentor is a Java-client/CGI-server application for image segmentation over the World Wide Web. It is made from the stand-alone program Segmentor, which is an object-oriented framework for image segmentation based on recover-and-select paradigm [8]. A case of range image segmentation with superquadrics is implemented.

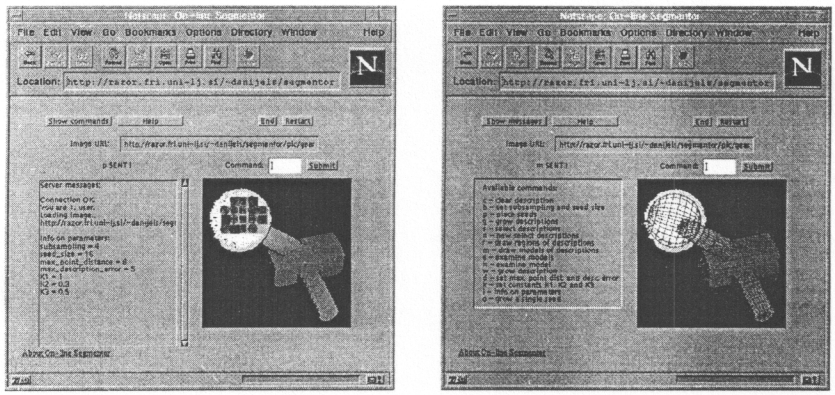
*wwwInterface* class was upgraded into *wwwSegmentor* class. It inherits all the properties of *wwwInterface* class and has additional methods which are specific to the Segmentor.

In the client program *WWWinterface* class is used. It establishes a connection and takes care of the data transport between the client and the server. Instead of pixels, only the parameters of superquadrics and vertices of polygons are transfered to achieve a better response time. Superquadrics and polygons are constructed from these data on the client. User's data input and the display of results are performed in the client program, too.

To be more specific, a user loads on his computer the web page from the URL *http://razor.fri.uni-lj.si/~danijels/segmentor/segmentor.html*. This page contains the client program which is started by the browser. Then the user enters an URL of a range image. This image is loaded on the server and then it is shown to the user in the client program.

The user can control the course of segmentation with available commands (place seeds, grow descriptions, select descriptions, etc.). The server program informs the user about the processes going on at the server during all time. The superquadrics are drawn simultaneously one by one after each superquadric is computed. Data input and sending requests are performed in parallel with receiving the answers and showing the results. Therefore a user can enter new commands while observing the results.

While modifying a stand-alone Segmentor into a client/sever application we were trying to achieve three main objectives: similarity to the original program, short response time, and small modifications of the stand-alone program. All this objectives have been achieved. The work with On-line Segmentor is very similar to the work with the original Segmentor, the response time has not increased significantly, and a great part of the code which upgrades the stand-alone Segmentor into the CGI server program is included in the *wwwSegmentor* class, so the code of Segmentor has not modified significantly.



(a) (b)

Figure 3: On-line segmentor (a) during the placment of the seeds, and (b) at the end – segmented image.

## 5 Conclusion

It is very important to make the work publicly available over the Internet; e.g. for comparison tests. In this paper we described a construction of a Java-client/CGI-server application. The implementation with the HTTP protocol using a HTTP sever was used. The stand-alone program Segmentor was transformed into a client/server application which can be used as a service on the World Wide Web.

Our main conclusion is that implementing the Java-client/CGI-server application for testing computer vision algorithms requires some modifications of the stand-alone program and it also requires creating the client program, however, it makes possible to test these algorithms on various computers and operating systems.

## References


[1] D. R. T. Robinson, The WWW Common Gateway Interface Version 1.1, University of Cambridge, February 1996. http://www.ics.uci.edu/pub/ietf/http/related/draft-robinson-www-interface-01.txt

[2] Computer Vision Home Page, Computer Vision Demos.
http://www.cs.cmu.edu/afs/cs/project/cil/ftp/html/v-demos.html

[3] A. Newman, et al., *Special Edition Using Java*, Que Corporation, 1996.

[4] D. Skočaj, *Execution of a remote program on the World Wide Web* (in Slovenian), Diploma thesis, Faculty of Computer and Information Science, University of Ljubljana, 1996.

[5] J. Bloomer, *Power programming with RPC*, O'Reilly & Associates, Inc., 1992.

[6] T. Berners - Lee, F. Fielding, U. C. Irvine, H. Frystyk, Hypertext Transfer Protocol – HTTP/1.0, Informational, May 1996.

[7] An Exploration of Dynamic Document in Netscape 1.1, Netscape Communications Corporation, 1995.
http://www.netscape.com/assist/net_sites/dynamic_docs.html

[8] A. Jaklič, A. Leonardis, F. Solina, *Segmentor: An Object – Oriented Framework for Image Segmentation*, Speach and Image Understanding, Proceedings of 3rd Slovenian – German and 2nd SDRV Workshop, pages 331–340, April 1996.