Project in Network Systems

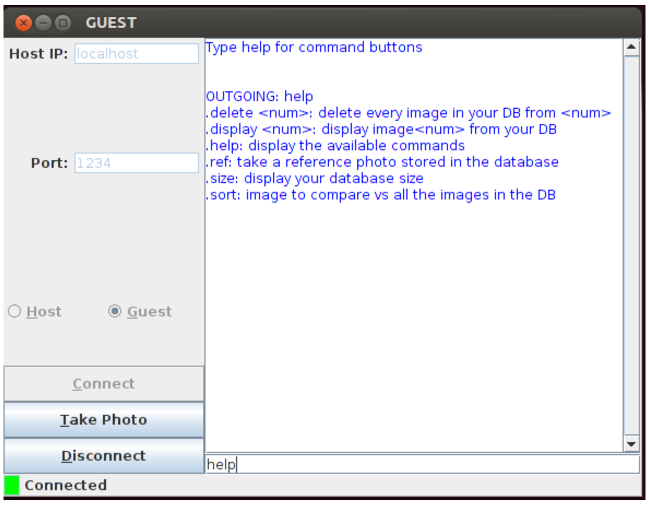
Pierre LAURENT
plaurent@kth.se

February 4, 2014

Abstract

This report explains the work that has been done in the Surf Project. It explains all the fundamental that have been understood till now and gives some guidelines to improve it.

Contents



```
Type help for command buttons


OUTGOING: help
.delete <num>: delete every image in your DB from <num>
.display <num>: display image<num> from your DB
.help: display the available commands
.ref: take a reference photo stored in the database
.size: display your database size
.sort: image to compare vs all the images in the DB
```

```
help
```

Figure 1: Example of the interface on the guest side - all functionalities

1 Introduction

The main goal of this project is to take a picture from a remote computer and to compare it with samples that are stored in a database. The results of this comparison have to be presented on a local computer. A given application is for example to figure out which traffic light color a driver has overpassed. The figure 2 shows the basic architecture of the project.

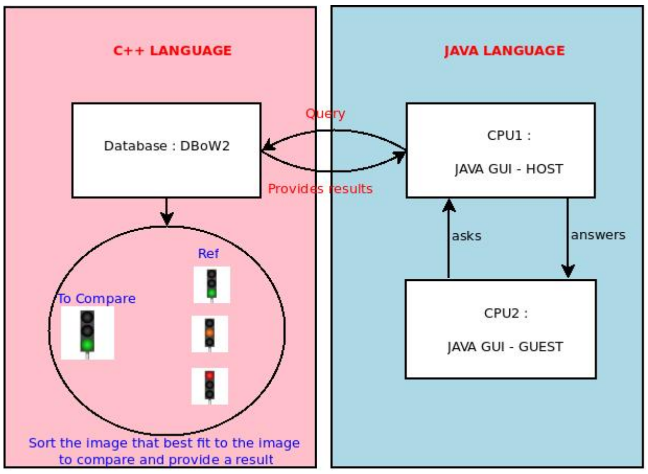


Figure 2: Basic architecture of the project

In this project, we have 3 major parts :

1. The JAVA GUI used by the user to connect to the remote computer and to ask/receive the results.
2. The database logic which has to analyze the image that are stored (in C++).
3. The connection between those 2 interfaces.

2 Code developpment and explanations

2.1 Java interface

The JAVA GUI is the client-oriented interface. To start the application, you just have to run the javainterface/src/application/Main.java class. In this class, you have the processing paths (database path and application link) so just modify it if you want to treat your data using other implementations. (see section Bug report for making it work using different OS).
To use it, you have to make 2 connections:
- one as a host (located on a remote computer)
- one as a guest (on your own computer – where you will ask the host for data).
In this part, GUI and commands have been separated in order to make it easier to modify the code if needed (ex: you only want to implement new commands etc . . .).

Nota: all the tests have been done using only one computer (with 64bits Linux OS) and running on localhost. To make it work on few computers, you just have to type the IP @ of the remote computer in the guest's Java GUI

2.1.1 Command functions

In the guest side, you have access to the remote options using the chat box. Here are the functionalities implemented (type it in the chat box or press the buttons):

- Connect/Disconnect (Button): first connect the Host and then the Guest. Be careful that you can only connect one guest to a host. At every disconnection you will have to reconnect both – this solution has been used because we want only one PC to be connected to the guest. Otherwise, you can create a thread on the host side that waits from guest connections.
- Take Photo (Button): Press it when you want to take a photo (every time on the host side). When pressing Take Photo, you will store it in the host's database (path images/ on Linux) as "image0.png". This photo will be taken using the webcam and will be the photo to compare, i.e. it will be the photo that will be taken as a reference when processing the database.
- other functionalities: type help in the chat box to see all functionalities that have been implemented. You can therefore fill the image database, display/delete images, find the database size, and process the database using the sort function (see next section).

2.1.2 Implementing new functionalities

To implement new commands, it is really easy as GUI part and logic part have been separated. To do that, you only have to modify the javainterface/src/application/ReceiveData.java class and add a new function in the switch case model. Remember to put a track of the new command in the command.txt doc so that it will be displayed when typing "help" (put a "." before the command so that it won't interfere with the host when calling "help").
Nota: the javainterface/src/application/SendData.java class is only used to implement how the data are sent. It is therefore better not to implement new command on this side as data are sent from the guest to the host, then processing on the host side and received on the guest side.

2.1.3 Code documentation

To make it easier to understand the code in its globality, a JAVADOC has been displayed. You can access it in the DOC folder. You can then have a global look at all the coding side.

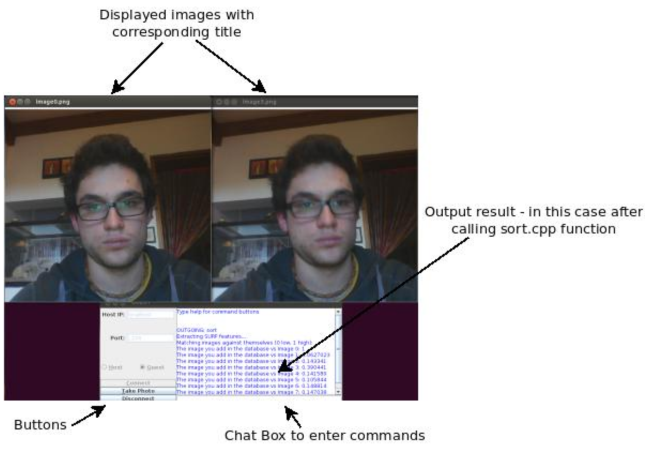
2.2 Database logic and interraction

The database logic uses the DBoW2 project of Dorian Galvez-Lopez.
It works using c++ function (see sort.cpp) and gives as an output the result of the processing between the image to compare and the references images (it is a grade out of 1 - 1 as image are same, 0 they are uncorrelated). To exploit the

C++ function in the java GUI, we provide a java Process on an output file and display the results (JNI has therefore been put apart).
If you want to implement differently the results (encode/decode with ASN etc ...), it is really easy and you just have to modify a bit the sort.cpp function.
To have a look at JNI, just search in the folder jni_guidelines where everything is explained.

3 Interface example

In figure 3 you can see an example of the graphical interface that is displayed on the guest side. In this example, you are given the output results when calling the sort.cpp function from the Java GUI.


Displayed images with corresponding title
Output result - in this case after calling sort.cpp function
Buttons
Chat Box to enter commands
Host IP:
Port:
Host
Guest
Connect
Take Photo
Disconnect

```
Type help for command buttons


OUTGOING: sort
Extracting SURF features...
Matching images against themselves (0 low, 1 high):
The image you add in the database vs image 0: 1
The image you add in the database vs image 1: 0.0627023
The image you add in the database vs image 2: 0.143341
The image you add in the database vs image 3: 0.390441
The image you add in the database vs image 4: 0.141589
The image you add in the database vs image 5: 0.105844
The image you add in the database vs image 6: 0.148814
The image you add in the database vs image 7: 0.147038
```

Figure 3: Example of the interface on the guest side

4 Bug reports

The developpment of such an application is quite complex because it recquires to work with 2 computers and to compile different programming languages. If you have some problem running the application, have a look at the following bug reports, this may help you fixing some bugs:

1. if you are running another OS than Linux (Windows or MAC for example), please report to the document window_mac_compatibility.txt located in the folder report. You will only have to modify the path name in the javainterface/src/application/Main.java class.

2. all the text file have been created using the .txt extension to make it easier for non-Linux OS.

3. if you want to try JNI/ASN encoding (which can create compatibility problems), please report to the folder JNI and use the given makefile to compile. You will then have to implement it in the Java GUI (you will have to locate your jni.h path; in Linux, type locate jni.h in a terminal).

4. to run the makefile using a 64 bits engine, put -lstdc++ -lpthread -lcrypto -lm -lz in the makefile (already done in the given one), otherwise, you will have many problems.