Tutorial on Client-Server Communications

EE368/CS232 Digital Image Processing, Spring 2015

Version for Your Personal Computer

Introduction

In this tutorial, we will learn how to set up client-server communication for running an image processing application on a server from an Android device. Since many image processing algorithms require high complexity cost, running these algorithms on a mobile client with responsive interactions is often infeasible. One remedy to this problem is to offload the work to a high-performance server over the network.

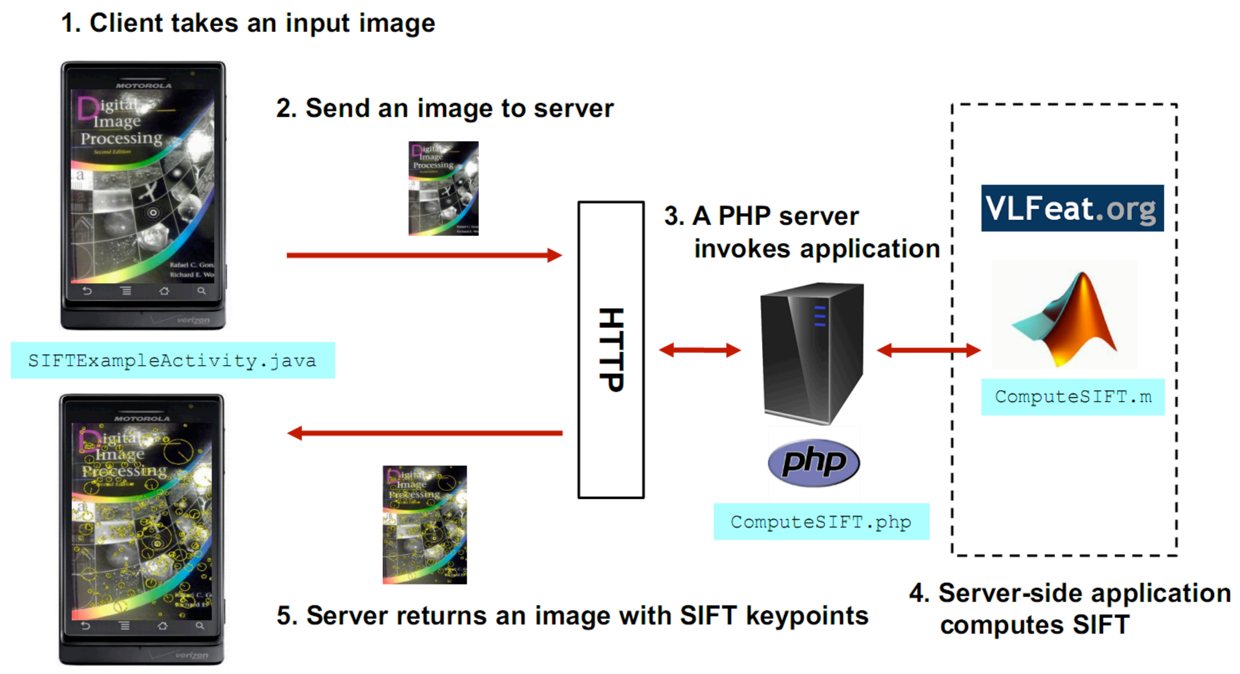
1. Client takes an input image
2. Send an image to server
3. A PHP server invokes application
HTTP
VLFeat.org
SIFTExampleActivity.java
php
ComputeSIFT.php
ComputeSIFT.m
4. Server-side application computes SIFT
5. Server returns an image with SIFT keypoints

Figure 1 An example of a client-server image processing system

As an illustrating example, suppose you would like to offload the computation of SIFT[1] from an Android device to a server in your class project. Figure 1 depicts a possible scheme for a client-server image processing system. The user will first capture an input image using an Android client and send the image to a server via HTTP. A PHP script on the server then invokes the server-side application to compute SIFT on the image. After the computation is completed, the server will send the result back to the Android device for display.

[1] SIFT: Scale-invariant feature transform. For reference, see EE368 *"Scale-Space Image Processing"* and *"Feature-based Methods for Image Matching"* lecture slides or Lowe, David G. (2004) *"Distinctive Image Features from Scale-Invariant Keypoints"*.

We will be using the above example throughout our tutorial. The tutorial is divided into two parts. The first part of the tutorial will focus on how to set up the server application. We will implement a PHP script to facilitate client-server communication and build a server-side application to compute SIFT using Matlab and VLFeat library. In the second part, we will explain how to handle client-server communication on an Android client.

Disclaimer: This tutorial is for educational purpose only. It is intended for students who would like to build a fast-working prototype for their class projects. Please be aware that the tutorial does not consider practical system issues, such as security and reliability, in a real-world setting.

Pre-requisite:

- Completed Android tutorial #1 of EE368, "Tutorial on Using Android for Image Processing Projects".
- Prior experience with PHP and running a web server.
- A server with Matlab, HTTP software (Apache, IIS, etc) and PHP installed

Part I: Server

Server Setup

This tutorial assumes you have a web server with HTTP software (Apache, IIS, etc) and PHP installed. Otherwise, please refer to the following links on how to setup a web server:

`http://www.wampserver.com/en/`	`(Windows)`
`http://www.mamp.info/en/index.html`	`(Mac)`
`https://help.ubuntu.com/community/ApacheMySQLPHP`	`(Linux/Ubuntu)`

Download Server Source Code

In your server, download the server source code of the tutorial from:
`http://www.stanford.edu/class/ee368/Android/Tutorial3/EE368_Android_Tutorial3_Server.zip`

Extract `EE368_Android_Tutorial3_Server.zip` to your web hosting directory. Example: `/var/www/` or `/home/<username>/public_html` in Linux/Ubuntu.

VLFeat Library

In this tutorial, we will use VLFeat library (`http://www.vlfeat.org`) to compute SIFT of an image on the server. VLFeat is an open-source software written in C with interface to Matlab for ease of use and experimentation. The library implements many common computer vision algorithms including SIFT, MSER and k-means. Some of you may find this library very useful for your class project.

To use VLFeat, download and unpack the binary package from:
`http://www.vlfeat.org/download.html`

Alternatively, you can use VLFeat package (`./vlfeat-0.9.14`) that is already provided in the tutorial files.

To use and interface VLFeat with Matlab, add this line to your `startup.m` file:

```
run('<path_to_your_VLFeat_Library>/toolbox/vl_setup')
```

Please refer to this link for creating or modifying `startup.m`:
http://www.mathworks.com/help/techdoc/ref/startup.html

Matlab Script : `computeSIFT.m`

Inside the tutorial files, we have already written a Matlab script, `computeSIFT.m`, to compute SIFT of an image for you. It takes an input image from the `upload` folder and converts it to a grayscale image with single precision:

```
InputImg = imread(input_img_path) ;
GrayImg = single(rgb2gray(InputImg)) ;
```

After obtaining a grayscale image, we simply use `vl_sift` from VLFeat to compute SIFT,

```
[f,d] = vl_sift(GrayImg) ;
```

- Each column of `f` is a SIFT keypoint with format [X; Y; S; TH], X,Y is the center of the keypoint, S is the scale and TH is the orientation (in radians).
- Each column of `d` is the 128-dimensional SIFT descriptor.

After SIFT is computed, we plot the relevant keypoints using `vl_plotframe` on the colored image and save the result to the `output` folder.

PHP script : `computeSIFT.php`

A PHP script called `computeSIFT.php` is also included in the tutorial files. Please have a quick look. This PHP script allows a client to upload a captured image and returns the difference-of-Gaussian (DoG) keypoints of the image from the computation of SIFT. Below is a code snippet from `computeSIFT.php` for taking an input image uploaded from a client:

```
#declare target path for storing photo uploads on the server
$photo_upload_path = "./upload/";
$photo_upload_path = $photo_upload_path. basename(
$_FILES['uploadedfile']['name']);

# copy temporary upload file to target path that stores the photo upload
if(copy($_FILES['uploadedfile']['tmp_name'], $photo_upload_path)) {
 #perform something on the image ….
}
```

On the client side, we will post an image upload to the server by calling `computeSIFT.php`. The image file is uploaded to a temporary storage area on the server. To save our uploaded file, we need to refer to the associative array `$_FILES`, which stores all the information about the file posted. There are three elements of this array you should know for our purpose:

- `uploadedfile` - the reference we will assign when we post a file from a client. We will need this to tell the `$_FILES` array which file we want to process.
- `$_FILES['uploadedfile']['name']` - `name` contains the original path of the client's uploaded file.
- `$_FILES['uploadedfile']['tmp_name']` - `tmp_name` contains the path to the temporary file that resides on the server.

After the temporary file path is known, we can copy the uploaded file to our `upload` folder for processing. To invoke our Matlab script `computeSIFT.m` in PHP, we write (see `computeSIFT.php`, line #59-60):

```
$command = "matlab -nojvm -nodesktop -nodisplay -r
\"computeSIFT('$photo_upload_path','$processed_photo_output_path');exit\"";
exec($command);
```

- `$command` variable defines the command line needed to execute `computeSIFT.m` as if you are running from a command line prompt. For the Matlab command, we specify the options "`-nojvm -nodesktop –nodisplay`" to disable Matlab GUI and display. We then call our Matlab function using "`computeSIFT('$photo_upload_path','$processed_photo_output_path')`". '`$photo_upload_path`' and '`$processed_photo_output_path`' specify the input image path and the output image path respectively.
- `exec` simply executes the command defined.

After the Matlab script is executed, `computeSIFT.php` needs to automatically push the result back to the Android client. We have implemented a PHP function called `streamFile` (see line #9) to do this for you easily:

```
streamFile( $location, $filename, $mimeType)
```

- `$location` is the file path of the source file.
- `$filename` is the filename that the client will use to save file.
- `$mimeType` is the character set type. You may use '`application/octet-stream`' as the MIME type in this example.

Testing Server Code

To test your server code, we provide `test.html` in the tutorial files. You can load `test.html` using an Internet browser and upload a test image in JPEG format. If your server scripts are setup successfully, an output image with SIFT keypoints annotation will be downloaded to your browser within a few seconds. Figure 2 shows an example of a test image and its corresponding result.

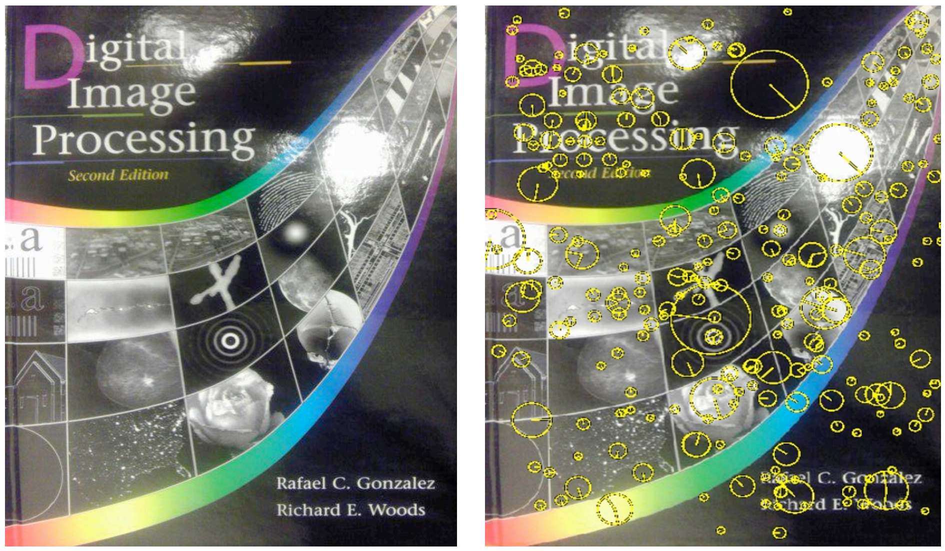

Figure 2 (Left) Input test image (Right) Output result with SIFT keypoints

When Matlab is not available
If you do not have Matlab available in your server, you can easily refer to VLFeat library and write your own C program for computing SIFT. Instead of invoking Matlab command, you can replace the `$command` (line #59) with your binary execution command in `computeSIFT.php`:

```
$command = "<your_program_path>/<your_program_name> <arguments>”
```

Part II: Android Client

Download Client Source Code
In your server, download and unzip the client source code from:
http://www.stanford.edu/class/ee368/Android/Tutorial3/EE368_Android_Tutorial3_Client.zip

Setting up Android Client Application
Open Eclipse, and select File → New → Android Project.

Choose ‘Create project from existing source’ and enter project name ‘`SIFTExample`’. For the project location, select the path of your unzipped client code.

Select your Android build target. Android 2.2 is recommended.

Go to `SIFTExampleActivity.java` and enter your server URL of `computeSIFT.php` by editing line #51:

```
private final String SERVERURL = "";
```
to
```
private final String SERVERURL = "http://<server-path>/computerSIFT.php";
```

Run the application on your Android device. Please make sure your phone is already connected to the Internet. A camera preview will be displayed. Take an image snapshot by pressing the camera button. Then, the image should be automatically uploaded to your server. If everything is setup correctly, an output image with SIFT keypoints should be displayed on the screen after a few seconds. To take another image, you can press the camera button again.

Congratulations! You have successfully built your own client-server image processing system.

Handling of Client-Sever Communication on Android Client

In `SIFTExample` project, we have three main source files:

- `SIFTExampleActivity.java`: Main activity file. Handles UI and client-server communication.
- `ResultView.java`: A surface view to draw your output image result.
- `Preveiew.java`: A surface view that displays camera preview. Same as what you have seen in Tutorial #1.

Often times, it is a good programing practice to have server-client communication not interfering with the flow of our main program thread. This helps keep our application responsive at all time. Therefore, one solution to this problem is to have an asynchronous task to manage our server-client communication. To implement an asynchronous task, please refer to `AsyncTask` API from Android SDK: `http://developer.android.com/reference/android/os/AsyncTask.html`. You can also refer to `SIFTExampleActivity.java`, line #180 to #338, for the actual implementation in the tutorial.

To offload our image processing to the server, we implemented a function called `processImage` `(`see `SIFTExampleActivity.java`, line #277 to #299`)` that abstracts all handling of server-client communication for you. Below is a code snippet of `processImage`:

```
void processImage (String inputImageFilePath){

        File inputFile = new File(inputImageFilePath);
        try {
                //<1> create file stream for input image
                FileInputStream fileInputStream  = new FileInputStream(inputFile);

                //<2> upload photo
                final HttpURLConnection  conn = uploadPhoto(fileInputStream);

                //<3> get processed photo from server
                if (conn != null){
                        getResultImage(conn);
                }
                fileInputStream.close();

        }
        catch (FileNotFoundException ex){Log.e(TAG, ex.toString()); }
        catch (IOException ex){Log.e(TAG, ex.toString());
        }
}
```

In `processImage`, `<2> uploadPhoto` (see `SIFTExampleActivity.java`, line #191- #257) will help you establish a HTTP connection to your URL and upload your photo. When the HTTP connection is successful, the server will reply your HTTP request with the result data. Using `<3> getResultImage(conn)` (see `SIFTExampleActivity.java`, line #260-273), we can download and display the data. Please take a look at these functions for their implementations. You may want to modify them for your project.

Reference

1. PHP - File Upload. http://www.tizag.com/phpT/fileupload.php
2. Post a File from the Phone to a PHP Server. http://getablogger.blogspot.com/2008/01/android-how-to-post-file-to-php-server.html
3. VLFeat. http://www.vlfeat.org/
4. Android AsyncTask. http://developer.android.com/reference/android/os/AsyncTask.html
5. Android HttpURLConnection. http://developer.android.com/reference/java/net/HttpURLConnection.html
6. EE368 *"Scale-Space Features"* and *"Image Matching"* lecture slides.
7. Lowe, David G. (2004) *"Distinctive Image Features from Scale-Invariant Keypoints"*.