EECS 22L: Project 2

Prepared by: Che-Wei Chang, Weiwei Chen, and Prof. Rainer Dömer

February 14, 2013

1 Optical Character Recognition (OCR)

This is the second team programming project of this course.

The goal of this programming exercise is to develop an OCR (Optical Character Recognition) program in which a user can convert scanned images of handwritten, typewritten or printed text into a digitized text file. OCR is a common method of digitizing printed texts so that these texts can be stored or processed in a more efficient way.

This project is designed to be an interesting exercise where you can practice all elements of software programming and engineering and work in teams. In particular, you will practice specifying and documenting the software program, designing data structures and algorithms, designing software modules, writing original source code, testing and debugging the software program, and collaborating and communicating effectively in a team.

Now, let's go!

1.1 Story

Mr. Nob Ackup runs a small software company that implements small-scale projects for its clients.One day, a thunder storm strikes. Lightning creates a sudden power surge followed by an outage.After power is restored, Mr. Nob Ackup finds his backup systems damaged and can only partially recover the source code of his projects from old tapes. Luckily, his software engineers find stacks of papers with printed hard copies of most missing source files that can be scanned into digital images.

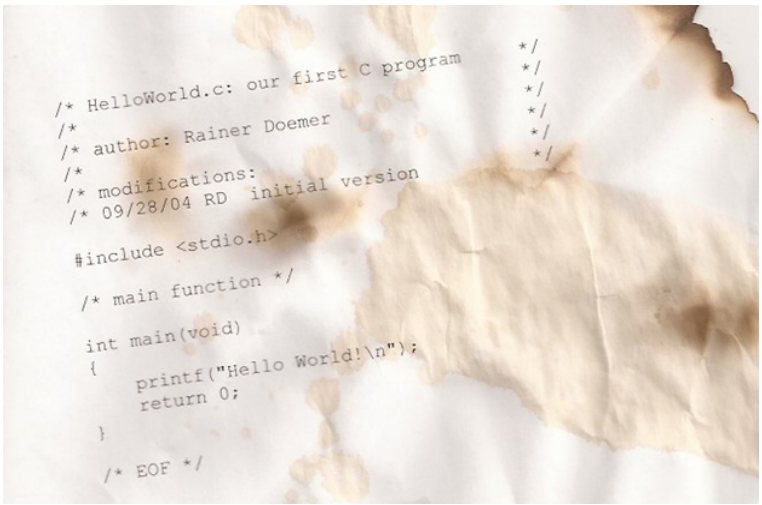

Figure 1: Example of the "recovered" source code

1.2 Tasks

Design a software package for Mr. Nob Ackup's company to efficiently convert scanned images of printed source code back into *compilable files*.

1.3 Considerations

- **Basic Requirement** Your OCR program should be able to read a clear image (fixed image format, clean paper, clear character, standard size, and perfectly levelled) and convert the characters in the image into a source code file.
- **Multiple Formats** The scanned text could be provided in different format, such as .jpg, .bmp, or .ppm. It would be helpful if your OCR program can support multiple image formats for input.
- **Multiple Files** The hard copies of the source code can be printed on several papers and scanned into multiple images. It would be helpful if your OCR can read multiple files and combine the contents of those images into one source file.
- **Rotated Images** In reality, the scanned text might not be perfectly levelled, and it might decrease the recognition rate of your OCR program (depending on how you implement your OCR program). It would be helpful if your OCR program can handle a scanned image which is not perfectly levelled.
- **Different Fonts** The hard copies of the source code can be printed in many kinds of fonts, such as Courier or Helvetica. It would be helpful if your OCR program can recognize the font and adjust the recognition algorithm according to the font to improve the performance.
- **Different Size** The characters in the hard copies of the source code can be printed in different sizes. It would be helpful if your OCR program can recognize the size of the characters and resize the image to improve the performance.
- **Stained Images** The hard copies of the source code could be stained (e.g. with coffee). It would be helpful if your OCR program can handle an image from a stained paper.
- **Wrinkled Images** The hard copies of the source code could be wrinkled. It would be helpful if your OCR program can handle an image from a wrinkled paper.
- **Handwritten Document** The easiest way to put information on the paper is through handwriting. It would be outstanding if your OCR program can recognize the handwritten characters and convert them into digitized text.
- **Human Interface** Your OCR program does not have to do all the work by itself, as long as you provide a user interface in your OCR program so that user can input important parameters, such as font, type, or rotating degree, to improve the recognition rate.
- **Dictionary Support** For some characters and digits like "1" and "l", "0" and "O", "C" and "G", it is very difficult for OCR program to distinguish those letters correctly. If your OCR program has a dictionary, it can help to make the right decision.

For a perfect OCR program, the features listed above are all desirable. However, in this project, you do not have to cover all features listed above, and you have the freedom to choose or come up with further advanced features to make your OCR smarter. Of course, the mores features your OCR program covers, the better it will perform in different kind of situations.

1.4 Design the program

We list several steps here to build a medium-sized programming project.

- *Design the software application specification*: work as a team to decide the functionalities of the program, the *input* and *output* of the program, and other things that describe the *features* of the program for the users.

- *Design the software architecture specification*: work as a team to design the data structure, program modules, application program interface (API) functions between modules, and basic algorithms that will be used to solve the problem.
- *Build the software package*: write the source code and implement the program. Each team member may be in charge of their own modules and work in parallel to implement the program. Use Makefile for rule-based compilation to integrate the modules from different owners.
- *Test and debug the software*: work as a team to decide the testing strategies, write the automatic test program or script, and debug the program if some of the test cases fail. Also, we will provide some samples of recovered images in directory ~eecs22/ocr_scans/ so that you can test the performance of your OCR program. In order to use the disk space on the server efficiently, please don't copy the scanned image files, but use symbolic links to refer to them!
- *Version control and collaboration*: use version control applications to maintain the team project documentation and source code files. Team members can synchronize their own work with the others through the team project repository.
- *Software release*: release the software package with the executable program and documentations, e.g. the README file, user tutorials, etc.

1.4.1 Team work

The programming projects in this course will be performed by student teams. Teams of 6-8 students will be formed at the beginning of this project. Team work is an essential aspect of this class and every student needs to contribute to the team effort. While tasks may be assigned in a team to individual members, all members eventually share the responsibility for project deliverables.

A *team account* will be provided on the `ladera` server for each team to share data among the team members. Since teams will compete in the projects, sharing of data across teams is not permitted.

Every student is expected to show up in both the discussion and lab sessions for team meetings. Attendance at the lecture, discussion and lab sessions is mandatory for the sake of team work.

2 What to turn in

To submit your team's work, you have to be logged in the server `ladera` by using your **team's account**. Also, you need to create a directory named `OCR` in your team account, and put all the deliverables in that directory.
Note that one team only needs to submit one set of deliverables.

Here is a checklist of the files you should have for each week:
We do require these *exact* file names. If you use different file names, we will not see your files for grading.

Now, you should change the current directory to the directory containing the `OCR` directory. Then type the command:

```
% /ecelib/bin/turnin22
```

which will guide you through the submission process.
You will be asked if you want to submit the script file. Type yes or no. If you type "n" or "y" or just plain return, they will be ignored and be taken as a no. You can use the same command to update your submitted files until the submission deadline.
Below is an example of how you would submit your homework:

```
% ls # This step is just to make sure that you are in the correct directory that contains OCR/
OCR/
ladera% /ecelib/bin/turnin22
```

Table 1: The OCR project deliverables

Week	File Name	File Description	Due Date
7	OCR_User_Spec.pdf OCR_SW_Spec.pdf	The application specification The software architecture specification	02/25/13 at 12:00pm
8	OCR_Alpha.tar.gz	The alpha version of the OCR program, including the program source code and documentation	03/04/13 at 12:00pm
9	OCR_Beta.tar.gz	The beta version of the OCR program, including the program source code and documentation especially the software testing specification and status	03/11/13 at 12:00pm
10	OCR.tar.gz OCR_Source.tar.gz	The released software package for the OCR program and the program source code and documentation	03/18/13 at 12:00pm

```
============================================================
EECS 22L Winter 2013:
Project "OCR" submission for weiweic
Due date:  Mon Feb 25 12:00:00 2013
* Looking for files:
* OCR_User_Spec.pdf
* OCR_SW_Spec.pdf
============================================================
Please confirm the following:  *
"I have read the Section on Academic Honesty in the *
UCI Catalogue of Classes (available online at *
http://www.editor.uci.edu/catalogue/appx/appx.2.htm#gen0) *
and submit myoriginal work accordingly." *
Please type YES to confirm.  y
============================================================
Submit OCR_User_Spec.pdf [yes, no]?  y
File OCR_User_Spec.pdf has been submitted
Submit OCR_SW_Spec.pdf [yes, no]?  y
File OCR_SW_Spec.pdf has been submitted
============================================================
Summary:
============================================================
Submitted on Wed Feb 13 20:06:02 2013
You just submitted file(s):
OCR_User_Spec.pdf
OCR_SW_Spec.pdf
% _
```

2.1 Verify your submission

This step is optional, but recommended. If you want to confirm which files you have submitted, call the following command:

```
% /users/grad2/doemer/eecs22/bin/listfiles.py
```

This command lists your submitted files. Don't worry if you submitted too many files. We will only look at the files with defined names (here: OCR_User_Spec.pdf, OCR_SW_Spec.pdf, OCR_Alpha.tar.gz, OCR_Beta.tar.gz, OCR.tar.gz and OCR_Source.tar.gz) and ignore other files.