Future Outlook Document

April 1

2013

Image Correlation Spectroscopy (ICS) is an application for performing analysis on images of bio-membranes taken from microscopes. It is capable of completing correlations between different image channels, which are then used to output graphs and values. The application consists of two parts: a standalone executable for Windows and Linux, and a web interface which can be set up on a web server and accessed remotely.

Developed for Dr. Nils Petersen at the University of Alberta as a project for CMPUT 401 in Winter 2013.

ICS:
Image
Correlation
Spectroscopy

Table of Contents

Introduction

This document provides an outline of potential future work regarding the BioMembranes project.

It includes insight into strategies for avoiding pitfalls encountered during the first iteration of this project, as well as an overview of possible future improvements.

Dropped/Deferred Features

1. Automatic Parameter Detection (#7)
   Parameters for the image processing algorithm can be determined from the input.
   This would remove the need for users to manually enter this information.
2. Microscope Metadata (#8)
   The program can determine parameters using metadata from the microscope.
   This would remove the need for users to manually enter this information.

Some other features were deferred within the project to future milestones, but by the end of the project, all requirements other than those listed above had been implemented to an acceptable standard.

Architecture

PyQT

PyQT was used to create the Local GUI for the project. Using this module ended up being a great decision for the following reasons:

- It is cross-platform, which was a requirement of our final product
- It is written in Python, which made integration with other components easier
- It comes with an IDE for GUI development, PyQT Designer, which made creation and modification of the interface easier
- It is well-maintained

Some drawbacks of using PyQT included:

- A lack of available documentation and examples online; while the PyQT Class Reference proved invaluable, it was somewhat lacking in completeness and clarity
- Difficulty with dependency installation

Our team recommends continuing the use of PyQT in future iterations.

Scipy & Numpy

The scipy and numpy Python modules proved very useful for the image analysis and computations needed for our product. As such, our team recommends continuing the use of these modules.

Django

Django was used to develop the Web GUI. It provided us with a good foundation on which to construct the scripts needed to run the Web GUI.

Our team has determined that Django adequately met our needs, and as such we recommend continuing the use of this module.

ImageMagick and PythonMagick

The ImageMagick and PythonMagick modules were used extensively in our project for image processing. They provided all of the functionality needed to create the final product. However, we ran into a few glitches in limitations which increased the complexity of our product, as well as the time and effort required to develop it.

Our team has determined that these modules meet the requirements of our product and would be a good choice for future iterations. However, it is worth doing some research first into finding alternative modules and determining if they would be easier to use.

Lessons Learned

Major Advances

Towards the beginning of the project, majority of the backend was completed which finished the computation-heavy component of the project.

We were able to find several excellent libraries to do the computations we needed, alleviating any concerns that our lack of knowledge in the area of image analysis would interfere with productivity.

We had excellent communication with our client, and were able to finalize requirements and interface design early in the project, allowing us to spend most of the project implementing things.

At the end of March, everyone met in the Computing Science Center for an entire Saturday afternoon. This was the first chance we really got to sit down together and work through everything we needed to get done, and was great for team motivation and productivity. After this, we had finished the bulk of the remaining workload.

Because we allowed for a lot of slack time in the project schedule, the setbacks we encountered did not prevent us from completing the project to a high degree of quality.

During the second sprint, we started utilizing Github extensively for issue tracking, which greatly contributed to project organization, task assignments, and project assessment. This saved us a lot of management time which we could instead spend on development.

Finally, most members ended up having more available time to devote to the project than was originally expected, which enabled us to spend adequate time designing, developing, testing, and repairing the product.

Major Setbacks

Members of our group were responsible for only one or two of the components of the product—backend, midend, local GUI, or web GUI—which meant there was difficulty with integration and a lack of consistency between different components. As an example, the Web GUI and Local GUI look quite different even after attempts to make them consistent. As another example, the midend was found to be necessary partway through the project to allow interfacing between the backend and frontend.

Towards the beginning of the project, there was discrepancy regarding the specific interface layouts as well as the nature of inputs and outputs. This led to a lot of overhead creating and revising the interface design to ensure it met the client's needs, which took away from development time.

During and after Reading Week, productivity declined and deadlines and sprints were pushed back. This was likely due to other coursework, members going on vacation, and a lack of classes and meetings. This happened during the same time we were having difficulties with calling the backend and generating graphs, which led to limited functionality in the local GUI for our halfway prototype, and no web GUI to show. Additionally, this led to the cancellation of the Profiling Sprint. This sprint had been intended to optimize and refactor the existing code to make it more efficient. Refactoring was limited, and items in this sprint had to be moved to the Completion Sprint. However, this did not prevent us from finishing the project because of the following reasons:

- Much of the computation profiling was completed during earlier sprints
- Most of the refactoring needed was with regards to code clarity but not efficiency
- We had intentionally left the Completion Sprint and Docs & Demo Sprint with a smaller backlog than earlier sprints in case something went wrong
- Many backlog items which were part of the Completion Sprint were not actually due until the Docs & Demo Sprint, and the Docs & Demo Sprint had a very small initial backlog, so items could safely be deferred to the next sprint without affecting deadlines.

Factors including time constraints, lack of relevant knowledge, and the separation of concerns between team members resulted in many task reassignments—that is, many backlog items were completed by someone other than the person to whom the item was originally assigned. This sometimes resulted in an uneven workload distribution between members and some items being delayed by one sprint or more.

Because of the lack of an actual server on which to set up and test the Web GUI, the ultimate fate and necessity of the Web GUI was in question. This had a moderate impact on team motivation and morale, and made it difficult to accurately represent the usefulness of a Web GUI for our client.

Several internal glitches within the ImageMagick dependency and some other dependencies meant that time was taken away from development in order to find and resolve these glitches. Development was occasionally put on hold for short periods because of this.

Before the Completion Sprint, there was discrepancy between team members regarding the program inputs and outputs and the purpose of each. This meant that we had to make several new GUI iterations during the project.

The requirement for our product to be cross-platform posed an issue. In particular, we were doing our development on UNIX operating systems, which meant that the build process for Windows was long and complex, and we ran into several unexpected bugs which occurred only on Windows. Because of the need to rebuild everything whenever a new code iteration was released, finding and fixing these bugs took a significant amount of time and effort.

Prior to the end of March, almost all work had been done individually and almost all communication between team members had been through email or Github, with the exception of our weekly team meetings and bi-weekly client meetings. This increased development time as the likelihood of implementing something incorrectly and having to repair it was increased.

Because of our setbacks, we did not fulfill the two requirements regarding automatic parameter detection. However, this is not a significant issue as it is not difficult for the user to enter these parameters, and these requirements were not high-priority.

Despite setbacks, our team was able to recover from the delayed sprints and backlog items and return to our original project plan by the end of the Completion Sprint. Our team feels confident that we have produced a high-quality product which fulfills our client's requirements.

Risk Retrospective

This section details which of the risks outlined in the project plan we ended up encountering, and what effect they had on the project.

High-Impact Risks Encountered

- *"Some group members are not familiar with Python or Matlab, so may require support from other group members who are knowledgeable on the subjects."*
    - Our team encountered this at many stages of our project. It resulted in different parts of the project being understood only by one or two members rather than the whole team, and was the root cause of our integration issues towards the end of the project.
- *"A member of the group may be too busy with other courses, or unwilling to dedicate the time necessary to adequately complete the project."*

    - At various points throughout the project, some members were unable to complete their assigned project components. However, we were able to still finish the project by shifting responsibilities and workloads.
- *"The server hosting the web component of our project may experience failure."*
    - While not a failure with a server itself, we were not able to secure a web server which means the web component of our project was only ever hosted on local machines.

Low-Impact Risks Encountered

- *"Github may become temporarily available."*
    - Github did go down, but it did not significantly affect our project as it did not happen during a time when we were pushing a lot of changes.
- *"A member of the group may become ill or otherwise unable to fulfill their duties."*
    - A member of the group did become ill, but it was not for an extended period of time so it didn't have a significant effect on our project.
- *"It may be difficult to schedule meetings with all group members, the client, and our TA are all able to attend."*
    - Attendance was good enough at the majority of the meetings, so this did not significantly impact our project.
- *"There may be difficulties in communication between the group members and client."*
    - The client was out of the country towards the end of the project, making communication difficult. However, by that point we had already communicated enough regarding requirements that it didn't affect our project.
- *"The undergraduate lab servers may become unavailable."*
    - The undergraduate lab servers went down multiple times, but since members did most of the work on their personal computers this did not significantly affect us.

Risks Not Encountered

- *"There may not be libraries available for multidimensional arrays and visualizations."*
- *"Numerical functions may be difficult for our team to code due to lack of background knowledge and/or experience."*
- *"A member of the group may drop CMPUT 401 or withdraw from university."*
- *"A member of the group may lack the background knowledge and require training such that they hinder the group rather than helping."*

Future Requirements

This section details possible requirements for future iterations of this project.

1. Support for OSX.
2. A fully set-up web server for running remote correlations.
3. Improved batch mode, with less stringent naming conventions, support for subdirectories, and so on.
4. Support for additional file formats.
5. Support for non-square images.
6. Support for batch mode archive files other than ZIP, such as RAR, TAR, and 7ZIP.
7. A dynamic zoom feature, possibly utilizing a scroll bar or the mouse wheel.
8. Force new windows (such as batch mode, the help dialog, or the zoom window) to show up in the same position as the parent window.
9. Enhancements to Web GUI upload interface.
10. User registration captcha for Web GUI.
11. Detect input parameters automatically, possibly through the use of microscope metadata.
12. Extension of Web GUI progress bar—support for single-image mode and display of simulation time remaining.
13. Addition of a message box to the Web GUI, similar to that of the local GUI.
14. Object-oriented refactoring of local GUI.