# Floc Size and Count App, Spring 2016

Christian Rodriguez(cer95), Anthony Verghese(akv26), Deniz Yılmazer(dy223)

May 20, 2016

**Abstract**

Turbidity measurements are the primary sources of performance data in water treatment plants. However, turbidity readings of flocculated suspensions do not provide any insight into the performance of subsequent treatment processes. The Floc Size and Count App Team's aim is to create an easy-to-use personal computer application that could measure floc size distribution using a digital camera with appropriate magnification. The app will be written in LabVIEW. The aim for this semester is to develop coding expertise in the LabVIEW environment and start working on the app that will be used in the AguaClara labs and eventually in AguaClara drinking water treatment plants.

## Introduction

The aim of this subteam was to develop an application that could measure the size and number of flocs from an image taken of water flowing at a particular point in an AguaClara treatment plants. This application will be used in the AguaClara project lab as a means of testing different designs for certain aspects of the plant and visualizing how the floc size distribution is affected by the changes. Optimal designs and specifications of the plant, thus, could be made with the aid of the floc size and count application. In addition, by utilizing the software and hardware developed, the operators of the AguaClara plants in Honduras and India would be able to test the efficiency of their respective plant and make sure everything is functioning smoothly by simply checking the application on their computers. It would work as a way to capture images from the flocculator and count the number of particles in each frame. After this, the program would analyze the physical properties of these floc collections. The floc sizes and masses are important pieces of data to understand the effectiveness of the flocculator since it would show how much of the particles bonded together. The flocculator efficiency translates to sedimentation efficiency since larger particles settle more easily than smaller ones. Therefore, an app that measures floc size and mass would be a way for the operators to learn more about the functionality of the plant and also perform further research.

There are two main components of this project. The first and initial part was to develop the software that would analyze the data. This software was constructed using LabVIEW. The second element is the development of hardware that will be used to collect the data on floc size and distribution, which will make use of cameras. However, this part will probably not be implemented this semester because fixing the code takes a significant amount of time and making sure that the software is running smoothly is crucial. The plan of the team for this semester is to master the LabVIEW environment and implement image analysis.

## Literature Review

Siwei Sun's thesis, concerning the characterization of flocs and floc size distributions using image analysis, was reviewed, and the content was used to better understand the functionality of the floc application (2016). She used LabVIEW and its Vision Builder Toolkit to explore particle size distribution changes during flocculation and sedimentation. The thesis goes into great detail describing how she tested her floc-measuring device. Different concentrations of polyaluminum chloride as well as clay, specifically kaolinite, were added to water to change turbidity and make sure the image analysis was applicable over a range of conditions. A detailed flow chart was included, displaying the steps of the experiment and where the hardware was placed (Sewei Sun, 2016, p. 24, 40). This information regarding her tests of the experiment was not necessarily useful at this point in time because experimentation will only take place

next semester; it may be more helpful in the future, when editing of the current draft of the software is completed.

More useful was the information describing the hardware and software used. Sun describes how the hardware of the imaging system consisted of an LED light source and a Flea3 FL3-GE-13S2M monochrome GigE camera (Sewei Sun, 2016, p. 25). The light was incorporated as a means of making sure the image was visible when uploaded onto the computer. With a clearer image, flocs can be distinguished from the surrounding water with greater ease. Another topic that Sun explained was the effect of airy disks. If the edge of a particle at a length scale is close to the wavelength of light, then airy disks often show up (Sewei Sun, 2016, p. 27). Errors occurred if these airy disks were not taken into account because airy disks cause the computed size of the particle to be inaccurate. Particles that were too small (less than 2.6$\mu$m) were often too difficult to analyze and could not be clearly identified. The optimal shutter speed for image contrast was determined to be 330$\mu$s or 10 times longer than the minimum shutter speed (Sewei Sun, 2016, p. 28).

Regarding the software used by Sun, the program began by applying a Gaussian filtering function to smooth the edges of the particles, making them more useful for analysis (Sewei Sun, 2016, p. 29). The program then applied a process known as "local thresholding" to the image. This effectively separates the particles from their surroundings by applying a dark color to the background and a light one to the particles (Sewei Sun, 2016, p. 29-30). A closing objects function was then made as a means of filling small holes and smoothing the boundaries of flocs captured in the images. The program removes flocs that were out of focus. Out of focus flocs were identified based on a weak light intensity gradient at their borders (Sewei Sun, 2016, p. 32). Flocs that are too small are also problematic because they cannot be measured accurately. The application is also unable to be sure of the dimensions of flocs whose edges cross the borders of the picture frame. Because of this, flocs that cross the image border were not analyzed. The remaining particles were then analyzed and their sizes were recorded in a comma separated value (csv) file (Sewei Sun, 2016, 37).

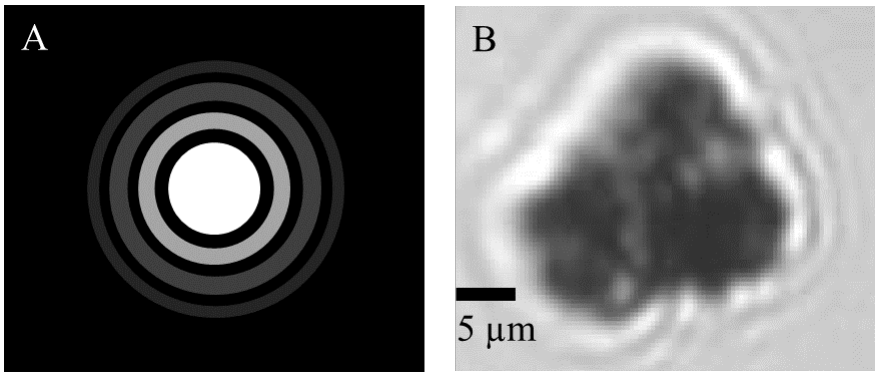


Figure 1: Effect of Airy Disks

Sun obtained a linear relationship between turbidity and the number of clay particles (44). The slope of the line was determined to be $1.9 * 10^6 \pm 2.2 * 10^5 \frac{\text{particles}}{\text{L*NTU}}$ (Sewei Sun, 2016, p. 45). In addition, a logarithmic decay plot was created by plotting number of flocs versus PACI dose (Sewei Sun, 2016, p. 50). Thus, with increased coagulant doses, the concentration of flocs decreased. Some of the topics brought up by Sun in the paper proved to be too advanced and confusing. For instance, the concepts of thresholding and Gaussian filtering functions were quite difficult to comprehend without previous exposure to them. Without much context, the reader was left without a solid understanding of what Sun was trying to describe. The thesis, however, provided brief descriptions of other aspects of her research on numerous occasions, maximizing the reader's understanding.

## Methods and Discussion

None of the members of the team had previous experience coding or creating software applications in LabVIEW. Therefore, the team focused on learning the tools available. The team was mostly focused on learning the basic data structures in LabVIEW, executables, shift registers, state machines, using configuration files, sub-VIs, type definitions and graphical analysis. In this respect, various teaching resources have been used by the team.

The main source for the team in this process were informative tutorials by National Instruments, the firm that developed LabVIEW. These tutorials were used for understanding loops, data types, graphical representations, signal processing and debugging (National Instruments, 2016). After watching these basic tutorials, the team members took varying approaches in learning how to use LabVIEW.

Some members of the team continued watching videos from YouTube by groups affiliated with National Instruments. One such video discussed configuration files (National Instruments Eastern Europe, 2014). Another discussed the construction of an executable using LabVIEW (EnableEngVideo, 2012a). After watching these videos, the team members proceeded to put what they had learned into practice in various ways.

These methods included constructing original programs, analyzing those already provided by LabVIEW, and making some under the instructions of "White Papers" provided by National Instruments. Additionally, a calculator was constructed using event structures, and many sub VIs were created as a means of making future programs simpler to make. Together, this work helped foster understanding of LabVIEW and help the members of the team generate ideas as to how the floc size and count application should be made.

The team refined image analysis code written by Siwei Sun and Casey Garland. The code was analyzed to understand how it worked and what exactly it did. This code was found to be functional but it was disorganized and required scrolling across a large block diagram to read. The team created subVIs to make the code more readable. The team is also working on the user interface design. By doing this, even those who are not skilled with LabVIEW would be able to work more easily with the code, thus making it more effective and useful in a real-life situation.

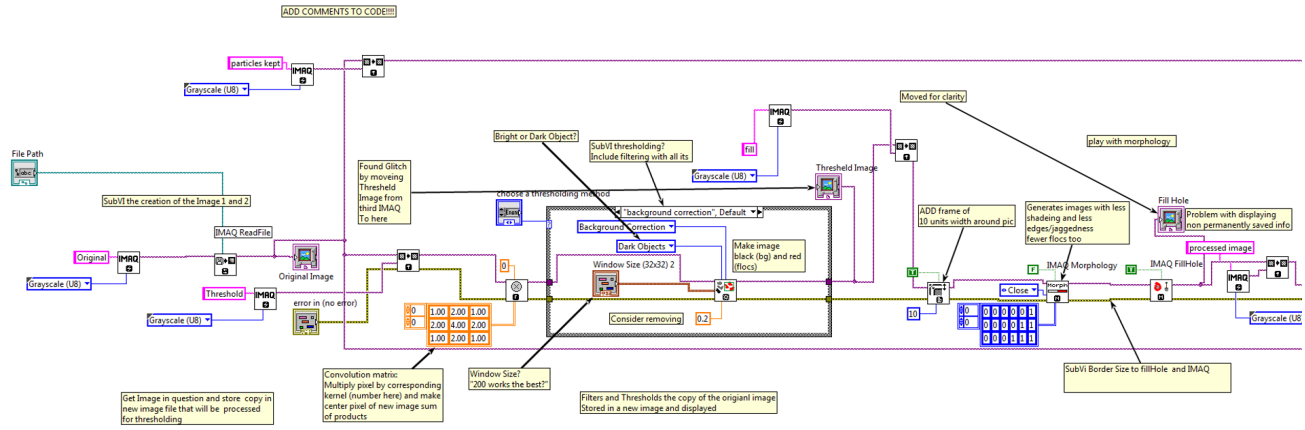


Figure 2: The above is only a portion of the original code for single image analysis. The majority of the code is not visible on a single 15 inch monitor.

Two parts of the process, capturing and analyzing data, were combined into one program that will take a photo and then analyze it according to user selected criteria. The primary goal in making this app is to help with research in the AguaClara project lab. Several teams are working with and researching flocs, so having this app will be very beneficial. In addition, this code will be the final product that will be given to the plant operators for their use. However, the code should be made into an executable app first for it to be used in all of the plants with ease. Therefore, the team learned how to create an executable. One problem with this approach is that the users will need a license for the Vision toolkit for LabVIEW that is used in the program.

The team split into three different "sub-sub-groups" after the initial LabVIEW training and code analysis was completed for greater efficiency. Thus, creation of subVIs, learning more about executables and combining data capturing and analysis parts of the program progressed in parallel, which increased the speed of the team.

To create the subVIs, small parts Garland's new VI; thus, a copy of Garland's code was essentially made that was more user friendly. To allow images to be transferred straight from a camera to the

program, the file path aspect at the beginning of the program was taken out and replaced with an image control. After this step was completed, focus was put towards the getting the executable prepared, which was a much harder task.

To save previous and updated versions of the subVI code, image processing code, and executable made by the team, sourceforge was used. Sourceforge allowed each member of the team to access the same VIs and edit them; although this resembled a google drive account shared amongst other people, sourceforge was a much better choice because a folder was installed on the lab computers' desktops so, rather than having to log onto gmail, code was accessed from the files on the desktop of each computer. This allowed for more efficient work on the code and sped up the process of working on the floc app.

## Executable

In order for the program to run outside of the LabVIEW development environment, an executable and an installer for the project must be built. Several team members prepared for this task by watching two YouTube videos created by EngableEngVideo; one discussed the creation of executables while the other discussed the creation of installers. The team then proceeded to practice what had been taught in the video. These videos went into great detail explaining the numerous setting options that must be decided on when "building" both files. In terms of the executable, the programmer can decide the name of the program, where it is stored, whether it includes the source code, the executable's icons, how it starts running (automatically or with the push of a button), and display features. In terms of installers, the videos explained how to decide where the executable is stored and how to add extra installers to the installer for the team's executable (EnableEngVideo, 2012b). Since the Floc Size and Count App makes use of the Vision Builder Toolkit, this section was particularly useful.

The team proceeded to create an executable and an installer for a previously created LabVIEW project that uses and edits a configuration file to store setting information. This exercise was tremendously useful. It not only provided basic training in the creating these to files, but it also exposed the team to numerous problems that may occur in an executable. While running the test-executable, it was discovered that the configuration file generated by the program was not being used. The cause of this defect, as was later discovered, was that the program stored the file inside of a private directory. Only when the program was run under administrative settings was the program able to successfully read and edit its configuration file. To solve this issue, the program was modified so that the configuration file would be saved in the *Public Application Data* directory. This experience provided great insight into the considerations that need to be taken when creating the executable and installer for the Floc Size and Count App.

## Storing Work

Once the team began to work with Cacey's original LabVIEW code, work was originally stored in AguaClara's Google Drive account. In order to edit the code, team members downloaded the code from the cloud storage system, make appropriate changes, delete the old files from Google Drive account, and upload the new ones. This eventually proved to be inconvenient and the team investigated alternative methods.

It was later discovered that many of the AguaClara subteams whose work consists of developing software store their work on TeamForge. In this website, users create repositories that store the source code of a project. A repository for this team had already been developed in AguaClara's TeamForge account with the name *FloatingFloc*.

Files are originally uploaded from a computer to a repository. In order to conveniently access and edit the repository, the team utilized the TortoiseSVN program that was installed in AguaClara's lab computers. For each team member, directories were created that downloaded the repository's files on command. The repository's URL was required in order to accomplish this (1). This was done following the instructions from AguaClara's wiki page (Carrin Moxley, Rachel J., 2014).

$$https://forge.cornell.edu/svn/repos/floating\_floc \tag{1}$$

There are many advantages that the team has enjoyed with TeamForge that were not present with Google Drive. Once this was done, editing became much more natural. The processes of downloading and uploading the team's work each requires only a click of a button using TeamForge as opposed to logging into a website, traversing a myriad of directories, and selecting the appropriate files. When "committing" (uploading) to the online repository, TortoiseSVN only updates the files that have been modified by the

user. So, as long as two users working on the same directory edit different files, both edits will be saved after each commits his or her edits to the repository. In addition to storing the current versions of a file, TeamForge repositories also store prior versions. This proved to be a particularly useful feature when one of the team members accidentally saved all of the work to a later version of LabVIEW, making it incompatible with specific packages used in the code. Another team members simply downloaded a previous version of the code and committed it soon after.

# Results and Analysis

## Modifying the Code

The primary objective of the Floc Size and Count App Team was spent on remodeling the original floc analysing software developed by Casey Garland and Siwei Sun. As shown in Figure 2, the LabVIEW code behind the analysis of each picture was both complicated and long, too long to fit on a single screen. The large size of the code made it difficult to see the key steps that were used to analyze each image.

It is worth noting that while creating subVIs, a minor glitch in the original program was discovered. The display that was supposed to represent the image only after the thresholding was incorrectly connected to the memory location containing the image after the holes were filled. This error was corrected. The thresheld image after this correction had many more holes inside of the flocs and more closely resembled the original image than the previous picture generated by the original code. This proved to the team that the modification was justified.

The new code is much more compact and logically organized. The codes division into subVIs was guided primarily by the image-processing steps outlined in Siwei Sun's Thesis. The filtering and thresholding processes were combined into a single subVI. The Hole filling, removal of unfocused flocs, and the analysis of the focused flocs were each turned into individual subVIs (Figure 3). The final code returned results that perfectly matched the original code's output (minus the improvement mentioned above).

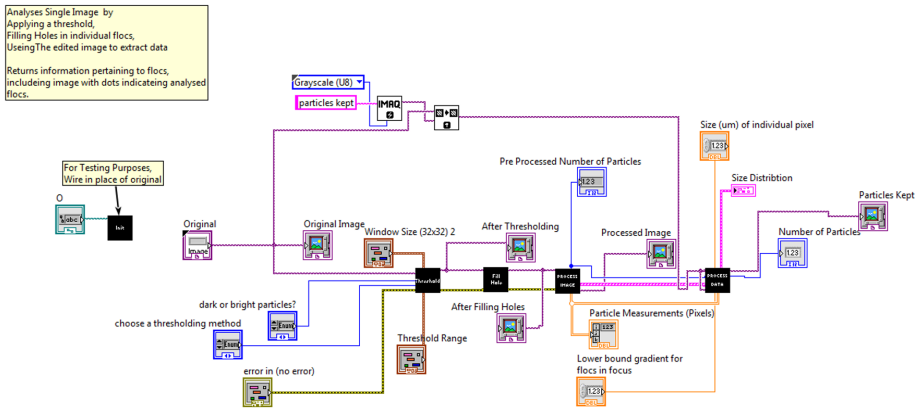


Figure 3: Notice that the whole program fits on one screen. Its organization strongly correlates to the steps of image analysis as described in Siwei Sun's Thesis.

## Executable

Using the information acquired from the works of EnableEngVideo, the team was able to create its first executable for a program that analyzes a single image (Figure 4). Since the program requires image inputs from a camera, it could not be tested. An installer for this executable was also built after the runtime engines from the Vision LabVIEW package were installed to the appropriate lab computer. Since the Vision package is not standard, computers would need other runtime engines to be installed to their computer in addition to the standard LabVIEW runtime engine. Before the installation, an installer that did not include all the recommended additional installers was generated. In the future, the project team should test both the executable and installer on a computer that does not have LabVIEW or its installers preinstalled; this would be the situation that most computers used in AguaClara plants would face.

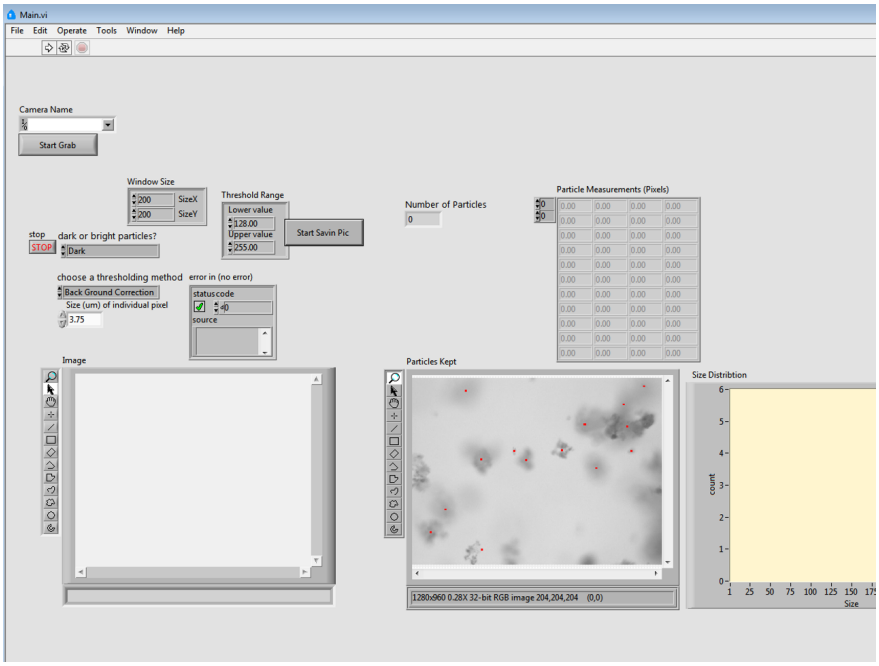


Figure 4: An executable that takes one photo from a camera and analyses it.

### Image Acquisition

The images that the program analyzes needed to be acquired by a camera from the flocculator. To be able to achieve this, the team created an image acquisition VI. The function of the program was limited because of late access to a camera to test the program functionality. However, the VI was in working condition at the end of the semester, able to take images with the camera, save the images to the computer, and return the image itself, which was successfully tested with a simple camera found in the lab.

### Final Product

The final product will be acquiring the data (photos of the flocculator in this case) and analyzing it in a single program. The team finished working on both data analysis and capture parts of the functionality for the product. However, these were two separate VIs at the end of the semester. The attempts to combine them together had not been successful. The image taken from the image acquisition VI was not compatible with the image requirements for the analysis VI. Using the file path for the saved image could be a way to fix this problem.

## Conclusions

Despite having difficulty with learning LabVIEW, a complex language to get accustomed to when used to languages such as java and matlab, for the first half of the semester, the first semester of the Floc App team was quite successful. Being one of the few teams using LabVIEW was quite tough because there were few people to aid the team when complications occurred. In addition, Casey and Siwei's code was quite complex, so trying to make sense of the given code that was necessary for editing was a tough process.

However, after getting used to the LabVIEW platform, working on the code became slightly easier. The subVI program was created successfully and is much more user-friendly, a great benefit for other teams that will use the floc application for their own research. In addition, the executable and image processing part of the code were almost completely finished, thus displaying how the application can soon be used by other subteams and possibly workers at the Agua Clara plants in Honduras.

## Future Work

Completing the image processing aspect of the code is the next step for the Floc App team. Currently, there are a few bugs in this part of the program, but fixing them should only take a few weeks in the 2016 Fall semester.

After the image processing aspect of the code functions properly, the data acquisition and analysis will be combined into one continuous program. To do this, the team needs to decide on the best way to implement this (use event structure, state machine, etc). The team also aims to make the program able to analyze an array of images instead of a single image. In a real situation, a camera will be taking many pictures at once, so having a program that can analyze many pictures continuously is very important. The team will additionally continue to develop the user interface of the program, making it easier to use.

Finally, the hardware components of the project will be added to be able to acquire new images. Members of the team will conduct research in order to find components that balance quality and price. The hardware will then be tested for compatibility and performance with the LabVIEW software in the AguaClara lab.

# Semester Schedule

## Task Map

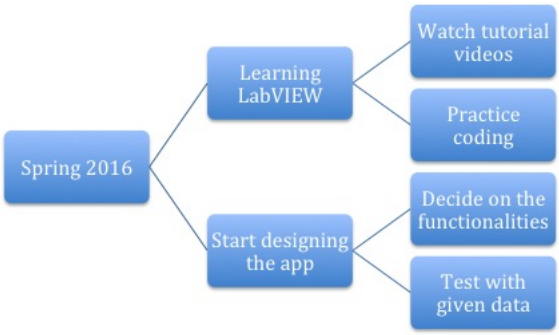


Figure 5: Spring 2016 Task Map

## Task List

1. Watching tutorial videos to learn LabVIEW - 02/24 - all members
2. Practice Coding - 03/23 - all members
   - Loops: Learned about how for and while loops are used in LabVIEW. All group members had previous experiencing with loops, but the unique aspects of LabVIEW made it somewhat confusing. However, tutorial videos as well as practice with LabVIEW helped promote understanding. Finished learning about loops 2/15.
   - Shift Registers: Shift registers were studied because converting output values into input values in loops became a necessity as more complex practice was carried out. Finished learning about and practicing with shift registers 2/15
   - State Machines: State machines allowed the members of the group to essentially create a set of if statements. Certain actions were carried out if certain specifications were met. A mock vending machine was created using a state machine. Finished learning about and practicing with state machines 2/22
   - Sub-VIs: Sub-VIs are essentially functions, and can be utilized in future programs to reduce the amount of code that must be written. This makes the code more user friendly, which is the main goal of this application. Finished learning about and practicing with Sub-VIs 3/3
   - Executables: Youtube videos about executables were watched; however, the topics were quite complex, so actual testing with executables was not conducted. Further research into executables will be done in the upcoming week.
   - Configuration Files: Although the team has looked into Configuration Files, more time needs to be spent on this subject.
   - Type Definition: Further research into the uses of type definitions must be conducted in the upcoming week.
   - Graphical Analysis: Using the graphical representation tools provided by LabVIEW will be further studied this week, because it will be very important as the group begins working on the actual application.
3. Start editing the code given by Casey Garland- 03/15
   - Take specific parts of the code and put them into subfunctions. Also, get rid of unnecessary parts of the original code.
   - Create a new VI with these subfunctions. This new VI is more organized and no longer has the unnecessary parts of the original code.
4. Decide on functionality of the final program - 03/29 - all members

5. Make program be able to process multiple photos continuously- 04/20- all members
    - Use event structures and loops to allow multiple photos to be processed continuously.
    - A VI will be created that completes this action.
6. Create a working version of the post processing image analysis part of the program - 05/10 - all members
7. Order hardware needed to upload photos to program and test the code with continuously generated photos- 05/20- all members

# Glossary

- **development environment**: A program that is used to edit the source code of a program.
- **executable**: A program that can run independent of a programming environment; they provide instructions directly to the computer for execution.
- **floc**: A clump of pollutants bonded together with coagulant.
- **Gaussian Filter**: A way of manipulating the pixels of an image so that their values are closer to local average values.
- **local thresholding**: A process that compares a pixel value to those of neghboring pixels and determines whether the pixel is part of the "background" or not; pixels representing the background are given one color while those that do not are given a different color.
- **installer**: A program that facilitates the installation of an executable, sometimes installing other necessary programs.
- **repository**: A folder stored in the accounts of numerous source code management websites. In practice, a repository stores the source code behind different projects as well as other related files.
- **source code**: The instructions that govern how a program operates.
- **subVI**: A method in a LabVIEW program that is used by another method.
- **TortoiseSVN**: A program that allows for the manipulation and access of SubVersion repositories. To use the program, select a computer directory and use the repository's URL to "sync" the directory with the repository. SubVersion is simply a means of controlling and distributing software. TeamForge uses this system, and so is compatible with TortoiseSVN.
- **VI**: A method in a LabVIEW program.
- **Vision Builder Toolkit**: a collection of objects and methods for LabVIEW that facilitate the manipulation and analysis of images.

# References

Carrin Moxley, Rachel J. (2014). SubVersionf - AguaClara - Dashboard.

EnableEngVideo (2012a). LabVIEW Tutorial 40 - Creating Applications.

EnableEngVideo (2012b). LabVIEW Tutorial 41 - Creating EXE Installers.

National Instruments (2016). Learn LabVIEW.

National Instruments Eastern Europe (2014). LabVIEW Tips&Tricks Episode 4: Reading and parsing INI files.

Sewei Sun (2016). *Characterization of Flocs and Floc Size Distributions Using Image Analysis*. Partial Fulfillment of the Requrements of Degree of Master of Science, Cornell University, Ithaca, New York.