Floc App, Fall 2016

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

December 10, 2016

Abstract

Turbidity measurements provide the primary source of performance monitoring at many water treatment plants. Turbidity provides an excellent way to measure overall plant performance, but it does not provide insight into why the water treatment plants are performing well or poorly. The floc size and count app team's aim is to create an easy-to-use desktop application that would measure floc size distribution and potentially provide insight as to why a plant is performing well or not. The app will be written in LabVIEW and made available for easy use as a stand alone executable application. The aim for this semester is for the team members to finish the LabVIEW program , integrate the program with a camera system and have other teams test the camera system and software. By the end of this semester, we hope to have a functional prototype that can be tested by other AguaClara subteams in their experiments.

Introduction

Water treatment operators and AguaClara researchers can learn a significant amount about plant functionality by observing particle aggregates (flocs) formed by plant processes at any given stage of the treatment process. Receiving data regarding average floc size and count helps researchers visualize effects of different designs in the plant and also allows on-site workers to determine whether all plant components are functioning smoothly, by comparing new data to previously retrieved data. Currently, turbidity measurements (cloudiness of a liquid measured by nephelometric turbidity units (NTU)) are the only tool that plant operators can rely on to gain information about plant efficiency. The World Health Organization (WHO) recommends a maximum turbidity of 5 NTU for drinking water, so this method of checking water quality is quite simple. However, the turbidity measurements do not provide any information regarding individual plant processes. Furthermore, these measurements indicate the efficiency of the plant without giving any indication as to why it is performing in a certain way. For example, if there is a clog in one part of the plant, an application that determines average size and count of flocs could ~~display~~help identify this issue on a computer screen and determine where in the plant the clog is, while turbidity measurements would simply tell researchers that there is a problem.

The goal of the floc app team was to create a program that would collect and analyze images at key locations of the plant (for example, after the flocculator

or sedimentation tank) and return the max Feret diameter and area of all flocs identified as well as the number of flocs. Currently, the program at hand is impractical since it analyzes only a single image. This semester, the team aims to upgrade the program to display data in a more organized manner and be continuous. The team also aims making the preexisting program from the last semester that only took one photo and analyzed that individual photo to become a continuous process, capturing and analyzing data in real-time without the need to constantly press a button. This would make the program much more user-friendly and practical.

Literature Review

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 (Siwei Sun, 2016, p. 24, 40).

Of particular importance, Sun describes the hardware of the imaging system consisting of an LED light source, flow cell, and a Flea3 FL3-GE-13S2M monochrome GigE camera (Siwei Sun, 2016, p. 25). The light was incorporated as a means of providing enough illumination for the short exposure time needed. The exposure time needed to be kept short to ensure a floc did not move significantly while the image was being taken. This provided contrast to the images, distinguishing the flocs from the surrounding water. 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 (Siwei 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 2.6μm) were often too difficult to analyze and could not be clearly identified. The optimal shutter speed for image contrast was determined to be 3μs or 10 times longer than the minimum shutter speed (Siwei 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 (Siwei 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 (Siwei 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 (Siwei Sun, 2016, p. 32).Flocs that are too small are also problematic because they cannot be measured accurately because of the airy disk phenomenon 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 (Siwei Sun, 2016, p. 37).

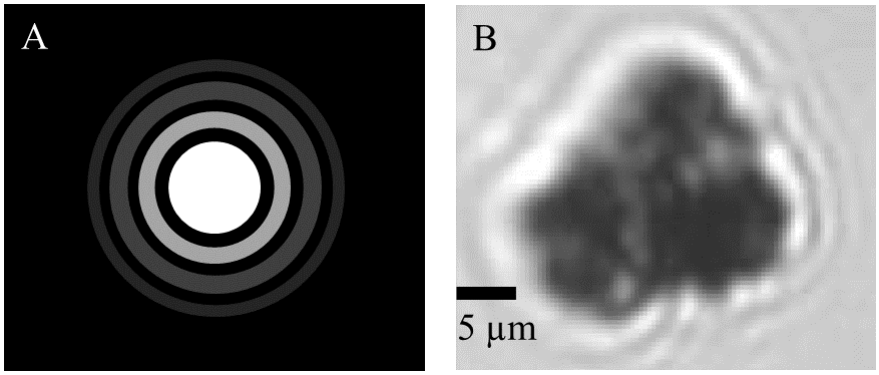

Figure 1: Effect of Airy Disks

Sun obtained a linear relationship between turbidity and the number of clay particles (Siwei Sun, 2016, p. 44). The slope of the line was determined to be $1.9 * 10^6 \pm 2.2 * 10^5 \frac{\text{particles}}{\text{L*NTU}}$ (Siwei Sun, 2016, p. 45). In addition, a logarithmic decay plot was created by plotting number of flocs versus PACI dose (Siwei Sun, 2016, p. 50). Thus, with increased coagulant doses, the concentration of flocs decreased.

Previous Work

During the Spring 2016 semester, the team worked to understand how LabVIEW worked in general and how the Vision extension package was incorporated in the source code. The team also tried to understand how a potential final product could be distributed to AguaClara lab teams in Cornell and plants across the world without requiring any licensing or complex installments. This required knowledge on building an executable as part of the research. The team unfortunately could not create an executable since the program was finalized at the very end of the semester.

The program was divided into two main parts. The first phase involved the acquisition of photos through a USB-connected camera, later saving specific images in a specified directory. The directory to use was obtained by the use of a pop-up menu.

The second phase dealt with analyzing the acquired photographs. In order to do this, the program made several modifications to the original photo to perfect the process. It began by applying a blur to the original photo in order to dull edges, making the borders less complicated. A user-selected thresholding scheme (back ground correction by default) was then used to separate flocs from the background, with small holes within flocs being filled. After making these modifications to the original picture, the program would then count the number of flocs and keep track of several important statistics, including the positions and the maximum feret diameters of each floc.

The program showed the final image, which marked the identified flocs and a histogram showing the size distribution of the flocs in the image. [Does the histogram show the distribution of flocs in a single image, or cumulatively for all images taken since the start of the program?]

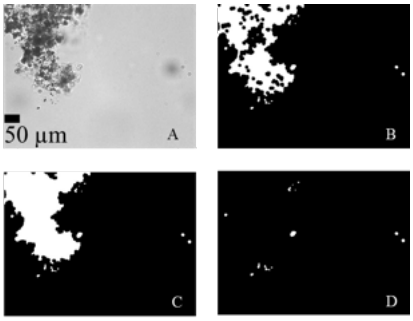

Figure 2: Processes of Image Analysis

After these two sections were completed, the team combined them in a single main VI. The program was usable at the end of the semester.

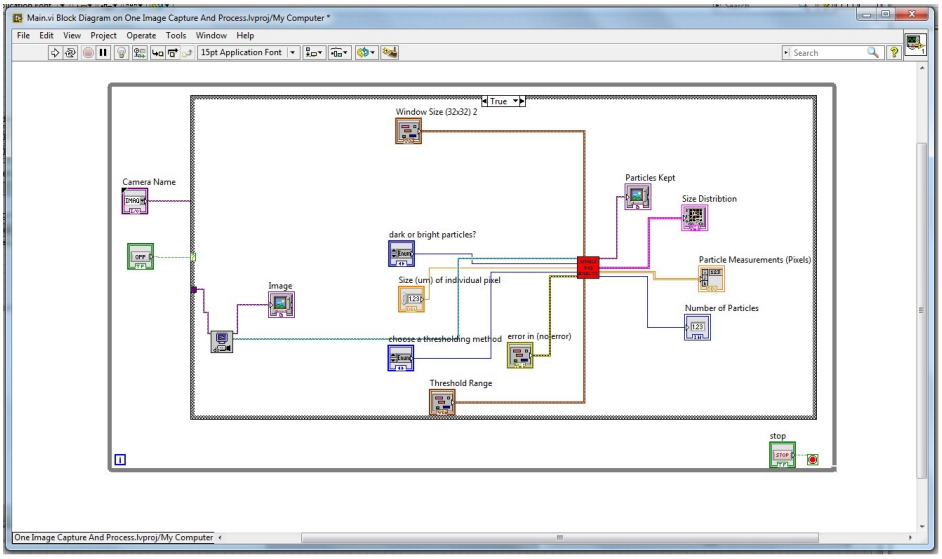

Figure 3: Final Main Code from Spring 2016

Initial Improvements and GUI Design

For the first three weeks of this semester,the team expanded upon last semester's work. This primarily consisted of furthering the team's understanding of the code, by meticulously commenting code previously made and looking up better ways to implement it. In addition, there were many instances of unnecessary code, which was removed as a means of condensing the extensive program. For example, near the beginning of the program, there was a large array that did

not seem to have a purpose, so time was spent understanding what its function was, and the team realized that it was not particularly useful for the Process Data SubVI, so it was removed. Also, there were many variables that were initialized at the beginning of the program and displayed at the end, but the data was not essential for the Floc Application, so they were removed, making the code look more visually appealing.

The team's work then shifted towards developing a graphical user interface (GUI) that was both user-friendly and aesthetically appealing. The controls and program-generated data were separated into two separate tabs. These tabs are further divided into different pages to further organize the information. A timer was also added to the program for the user to examine the duration of the test.

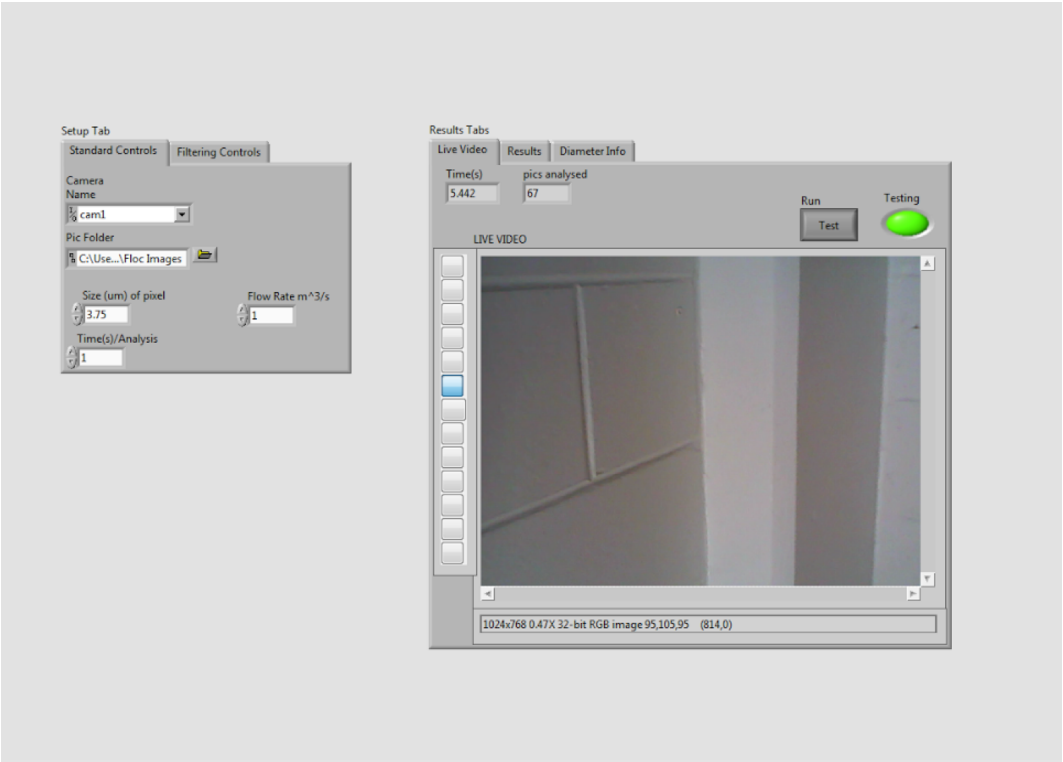

Figure 4: GUI of the program

Continuity

Throughout the semester, several attempts were made at creating a fully functional prototype. The first of these was inadequate because it did not show live-video of the flocs while simultaneously sampling certain pictures. In order to solve the issue of live-video, the team researched, modified, and incorporated a program made available on a community page hosted by National Instruments (LVbum42, 2013).

The program was supposed to take a picture after every kth second, which

was defined by the user. Timing was important because the user may want to see how the floc size and count changes over short or long periods of time. For faster flowing water, more pictures can be taken per unit of time. In order to achieve this goal, the team used the fact that each iteration of the loop taking the live video takes $\frac{1}{f}$ seconds, where f is the frame-rate. The team multiplied f by k, which is in seconds, and determined whether the counter's value (i) was a multiple of this reciprocal. The operation can be described with the following Boolean statement:

$$i \bmod^{c0} (k \times f) = 0$$

If the above statement evaluated to true and i were not 0, the image taken at that time was stored for analysis. Thus, the current prototype of the code only functions properly if the camera is capable of taking images at 30 frames per second. Although this is not ideal, the program was successfully made to continuously process images.

Buffer

One of the problems the team encountered when continuity was implemented was that the program failed to analyze some of the images at certain high frequencies. For instance, if we wanted to have the program analyze a picture per second, it would analyze 50 pictures as opposed to 60 pictures in the span of a minute. This was caused because the analysis part of the program ran slower than the entered frequency (determined by the camera's frame rate and the Time/Analysis control) therefore missing the data that came in while it was analyzing a previous image. To solve this, the team decided to use a buffer to store the images that should be analyzed to ensure the given frequency. The team initially used an array to work as a buffer but timing issues arose. The team then used a queue, believing that it would be more efficient. The team revised the code to include a queue that took images at a certain frequency after image capture step and the queue was accessed through a local variable in the analysis loop, which served as an efficient buffer. However, based on testing with various frequencies, the team discovered that it took a few seconds to push an image to the queue, limiting the frequency once more. However, we still improved image handling compared to the program without the buffer.

Making the Program Usable

After creating the source code for the Floc App, the team needed to finalize the product, making it convenient to use and easy to distribute. To achieve these tasks, an executable (also called an application) and an accompanying installer were developed. This is the standard alternative to simply distributing the source code. It allows users to interact with the software without the LabVIEW development environment, increasing program efficiency and providing a quick, direct means of launching the Floc App. In addition, any other program necessary for running the Floc App could be automatically installed onto target computers.

The LabVIEW development environment provides a simple means by which to create applications. One begins by creating a project and including all necessary VIs into the project. In this project their is an option for creating an application. Once this is clicked, a pop-up menu lets the user define several attributes pertaining to the executable. The more important ones are the project name, the startup (main) VI, the "always included" VIs, which are any VIs used by the startup VI or one of its subVIs, and the icon. The team was also able to control settings pertaining to when the program would begin running and how the interface would look.

The installer was made by a similar means. Again, the team was able to control several features pertaining to the installer, such as its name, the text it would display in the start page, and what executable to install. In addition, LabVIEW also had a whole tab dedicated to additional software that the installer would download if not present in the target system. This portion was done automatically and included the latest run-time environments for standard LabVIEW and the libraries for used for image acquisition and analysis.

When the executable was originally created, none of the outputs generated by the program updated themselves: the timer remained at 0s; the live-video screen was blank; and no flocs were ever analyzed. For most of the variables, the problem appeared to be in the way they were used in the code. In order to be properly updated, the output variables needed to be declared outside of the thread in which they were used, with a local variable used where the actual variable was originally placed. This fixed all of the issues with the application except for the fact that no fared diameter of floc count data was collected. The team attempted several procedures for fixing this bug, installing the executable using the installer, activating a few LabVIEW licenses, and restarting the computer. Eventually the application functioned properly, but the team still does not know for sure which of these actions contributed to the program's success.

Hardware Design

The ideal hardware component for the program is very expensive since it requires a high resolution scientific camera to take quality images of flocs. However, the team is considering using cameras that are more affordable to get a wider use of the program for the purposes of getting user feedback about the interface and usability. The team will also use the scientific camera with one chosen subteam to get feedback on the overall use of the program. The setup of the program will be from the AguaClara wiki website. The team has already created a wiki page for the program and it's use, explaining the installation and the parameters of the program for outside users.

Wiki Page

A Wiki page was created on the AguaClara web site to give step-by-step instructions for properly using the floc size and count application. In addition, access to a link to easily download the application is provided. Access to the application through the Wiki page makes it much easier for researchers to utilize the floc size and count application.

Future Work

In this semester, the team has created a functional prototype of the project as well as an executable and installer for the program. If future plans go according to plan, the program will be tested by selected teams within AguaClara. The team has already discussed several aspects of this phase of the development process, including possible teams, motivation, and specific ways of testing the software. Once the necessary equipment arrives to the AguaClara facilities, these ideas will be implemented.

The current program lacks a means of scaling from pixels to real-world units. This is essential if the program is to provide meaningful data pertaining to the size of each floc, not just relativistic information. Graduate students working in conjunction with AguaClara have previously developed an application that accomplishes this. Therefore, for next semester, the team hopes to analyze the source code behind this program and incorporate it into the Floc App.

Semester Schedule

Task Map

Figure 5: Task Map

Task List

1. Finish comments on the code and organize the GUI to be more readable (9/19) - All team members
2. Research ways that continuous data capture and analysis could be implemented in LabVIEW (10/13) - All team members
3. Create the program that can continuously capture and analyze data (10/27) - All team members
4. Design a hardware component that could be easily used in the lab and on the field by operators (11/18) - All team members
5. Distribute the program and gather user feedback (12/15) - All team members

References

LVbum42 (2013). Community: Snap and save frame from IMAQdx grab - national instruments.

Siwei 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.