Design of Computer Vision Stream Gauge

Blair Munro#1, Katarina Godden#2

#Department of Electrical Engineering, University of Alaska Anchorage
Anchorage Alaska, United States of America

1bmunro@alaska.edu

2kgodden2@alaska.edu

Abstract — **Monitoring the height of streams is essential in order to manage water usage and availability in a specific region effectively. Analysis on the heights of bodies of water is critical data to measuring the changing climate. In addition, aquatic life in Alaska is very sensitive to changes in streamflow. However, monitoring a traditional stream gauge with a display intended to be read by a human analyst is very tedious, especially in regions such as Alaska. More than 12,000 rivers and 3 Million lakes cover over 40% of the state's surface area [1]. The cost of obtaining this data quickly becomes unreasonable with so much area to cover. In this report a design is proposed for automatic recording of stream gauge data using computer vision. Software will calculate the height of a body of water from images of a rectangular gauge in Whittier, Alaska. MATLAB software successfully computes the waterlevels of nine baseline images, with uncertainties of ±1".**

Introduction

The National Oceanic and Atmospheric Administration NOAA is interested in measuring the water flow in streams. Currently, NOAA maintains a network of nearly 9,000 stream water level gauges across the country. The immediate use for water level data is to monitor flood hazards that pose threats to local populations and commercial livelihoods [2]. This data can also directly relate to the ice coverage at the poles, which is currently at an all time low, in conjunction with the annual precipitation rates. Obtaining historical data of this type could allow more precise measurement of the climate's rate of change.

At a local level, Alaskan fish habitats are sensitive to large and small flow events. Capturing gauge height information also makes it possible to deduce information about flow rates [3]. Having more accessible water level data across the state of Alaska will allow fish and game, and fishery workers, to access real time information about changing spawning conditions [4].

A stream gauge typically quantifies river or stream height, known as the stage of the water. The measure of stage is called the gauge height, and the zero is typically the bottom of the riverbed, or the gauge itself [5]. Typical stream gauges require either in-field manual measurements, or complicated mechanical systems involving considerable material investment. Newer approaches to the problem involve using software to automatically determine the water height without continued human observance. Using the gauge images, this software will calculate gauge height and it will be customizable so that it can be used for any color stream gauge at any location. Customization choices will be made via a user interface, so advanced programming knowledge will not necessarily be required of the user to work with this system.

This software uses color detection in conjunction with image filtering and image erosion to estimate the height of the water to the nearest quarter inch from a camera that is approximately 100 ft from the gauge. These numbers are particular to the camera/gauge setup in Whittier. Identical cameras that are closer to the stream gauge will have a higher resolution. Hardware limitations and complications with the camera calibration are examined and will be discussed.

The purpose of this project is to design a computer vision software that will intake images of a stream gauge to determine the height of the water automatically. The testing data set captures images of a gauge in the port of Whittier, Alaska. As the water level fluctuates due to incoming and outgoing tides, an inexpensive wireless camera takes photos of the water level rising and falling against a brightly colored rectangular stream gauge. A water level detection system like the one outlined in this paper will be by default inexpensive, and extendible.

Project Goals

- *Create a system that can be implemented anywhere to measure the real time height of a body of water autonomously.*
- *Determine ideal field measurement apparatus.*
- *Utilize MATLAB image processing capabilities to infer water height from test data set as proof of concept.*
- *Develop user interface for customization of system specific values.*
- *Assess system limitations and image distortion implications.*

Methods

This water level measurement system breaks roughly into two categories: Hardware and software. Hardware includes the gauge, the camera, and installation of the two. Software is composed of the image processing components, and the user interface for setup purposes.

A. Hardware

The apparatus consists of a brightly colored stream gauge braced against an abutment to ensure it is perpendicular with the water level. For best results, the gauge color should be one that is not found naturally elsewhere near where the images that are being taken. There is a camera less than 100 ft away and approximately perpendicular to the gauge that will capture an image of the gauge once every time period. Above the gauge there is a grid that can be utilized for camera calibration and image detection if necessary. If the camera is close enough or the images are of sufficient resolution, the calibration would help reduce image distortion. Figures 1 and 2 show the apparatus from two different perspectives.

B. Camera Calibration

Camera Calibration is a way to correct optical distortion introduced by the camera itself [6]. Each camera is different, each with its own lens aberrations and sensor misalignment. In extreme cases, distortion introduced by these imperfections can warp simple image geometries into shapes that are too distorted to analyze.

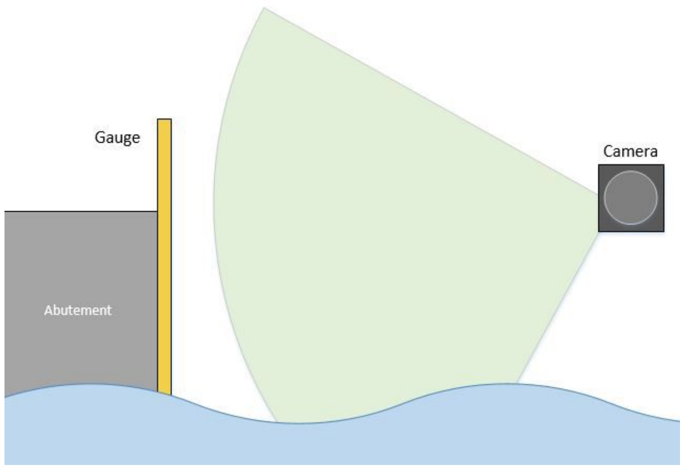


Figure 1: Apparatus - Side View

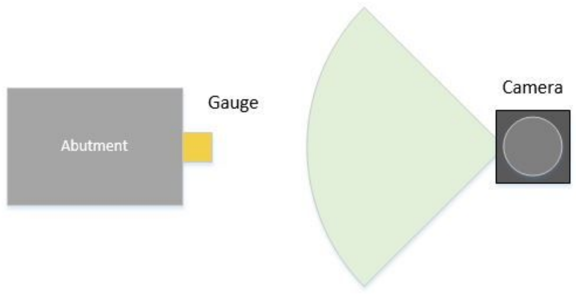


Figure 2: Apparatus - Aerial View

For cases where the colored stream gauge is not centered in the picture frame, and when the gauge is very close to the camera, distortion may become non-negligible. In most cases however, the gauge will be far enough away to neglect distortion. The simplest method for measuring stream gauge water levels is relatively insensitive to distortion of the rectangle itself.

In cases where distortion must be corrected, calibration and distortion removal is a relatively simple process. Calibration can be performed in the lab before deploying the camera to the field. Distortion removal occurs on an image by image basis, being the first step in image processing. A detailed outline of camera calibration and image distortion removal is provided in Appendix B.

C. Software

The software consists of 5 main stages: user set up, image ingestion, preparation and filtering, and measurements and analysis. The user setup is described in detail in Appendix A and the analysis is included in the results and conclusion of the report. A block diagram of this workflow is shown in Figure 3.

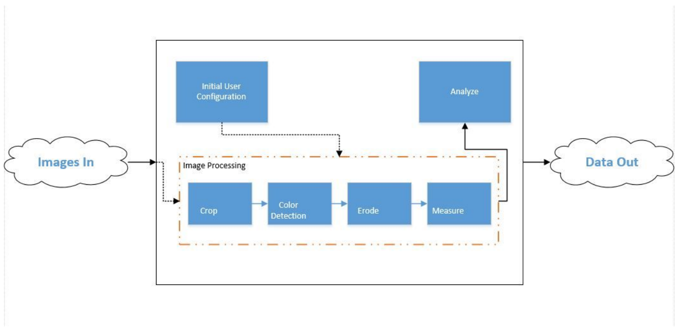


Figure 3: Software Block Diagram

1. *Image Ingestion*:

The images should be live or batch streamed to a specific location that can be pointed to within MATLAB. This could be customized to be done via a RESTful API, but for this proof of concept the images are directly loaded from a Google Drive Folder. The images can be in any standard image format. It is essential for the images to include the timestamp in the metadata in order to synthesize the changing water heights over time.

2. *Preparation and Filtering:*

The images are cropped to show only the area in which the gauge could appear. This limits the noise that could be accrued during the detection. For the color detection method, the image is then cropped to take a tiny segment from the very top of the gauge that will never be covered by water. An average of the color over that segment is calculated, and then using an appropriate threshold the entire image is scanned for colors within that spectrum. The new image specifies if the color within a threshold is detected in each pixel. The result of this is a logical matrix, for which “1’s” represent the piece of the gauge that is uncovered. Figure 4 displays the blob detected using only simple color detection.

An additional technique attempted to detect the gauge was measuring the directional and magnitudinal gradients of the images. Without erosion, hole filling the blobs that were output using the color detection technique were very misshapen on the vertical edges as shown in the Figure 4.

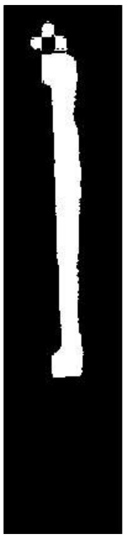


Figure 4: Blobby Edges

Figures 5 and 6 show the MATLAB documentation test example of calculating the magnitude and directional image gradients, and the same gradient calculations applied to one of our highest quality test images. [7] As is observable, the gradients are not effective in detecting the edges of the gauge, and in fact are even worse because of the poor camera resolution.

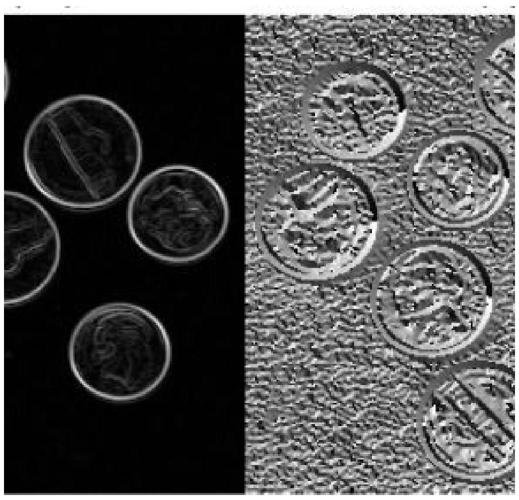


Figure 5: Gradient Potential - MATLAB [7]

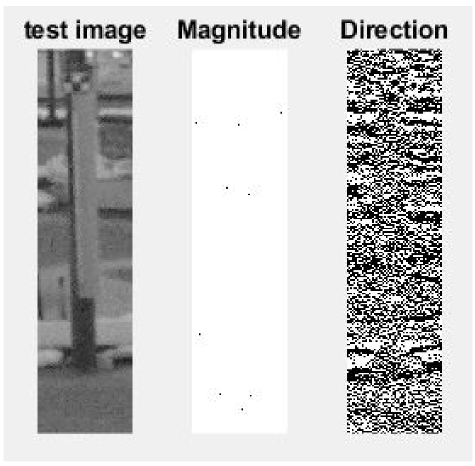


Figure 6: Test Image Gradients

However, as observable in Figure 4, the horizontal edges were very sharply detected using the color detection technique. This trend was true across the entire testing data set, so as a result it was decided in order to minimize error the color detection technique would be sophisticated, and the water height should be inferred from the maximum height of the detected blob, rather than the entire exposed surface area of the gauge.

Next, the image is eroded, removing any connected components that introduce noise as a result of the grid or any other sections of the images that may have introduced false positives. In addition, the area of the parts that are not eroded are filled to correct the false negatives. The eroded blob is shown in Figure 7. It is interesting to note that the erosion process smoothed out the horizontal sides of the gauge but made the vertical sides even more misshapen. This was also consistent across the testing data set.

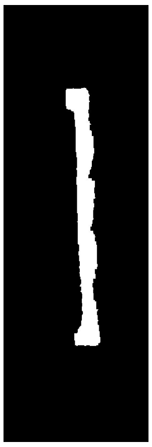


Figure 7: Eroded Image

The software does not work for images taken at night without lighting. If night data is desired there must either be a light mounted above the gauge to illuminate it, or possibly a flash on the camera could work as well.

3. *Measurements:*

Each pixel in an image in the testing data corresponds to ¼ inch of exposed gauge for this test case. This relationship allows the relative water height to be determined by counting the number of pixels for which a color within the threshold is found in the image. To determine the number of pixels per inch, it is necessary to use a cursor tool to pick out the pixel coordinates of the lowest, and highest exposed pixels. In the case where the entire stream gauge is exposed, then the known length of the gauge must be divided into the number of exposed pixels picked out by the cursor tool. In the case where only part of the stream gauge is exposed, the exposed gauge length within the picture that pixels are counted must be measured in the field manually. Counted exposed pixels are divided by this measurement to provide a pixels per foot or inch.

Various averaging techniques were tested, but the most robust approach that had the least error was also the simplest approach. The maximum height of the gauge was calculated, rather than the total surface area exposed, because the vertical edges of the images were much more blobby than the top of the gauge and the water interface.

D. User Interface

Instructions and interactive functions were added so that a user can dynamically specify the cropping window without editing the source code. The user is given instructions for two different scenarios: where the entire gauge is visible or where only a portion of the gauge is unobstructed. Depending on this choice, the user is able to input the height of the exposed gauge and pick out the rectangle corners in the image. This makes the pixels per foot measurement customizable to different sets of data. Finally, the user specifies crop windows within the image the enable color detection and reduce image size. This allows for the code to scale across many different data sets. Detailed instructions for custom user set up is included in Appendix A.

Results

For typical program use, results are produced one image at a time. With each new image the program computes the water level, outputting the number in a tabulated data sheet.

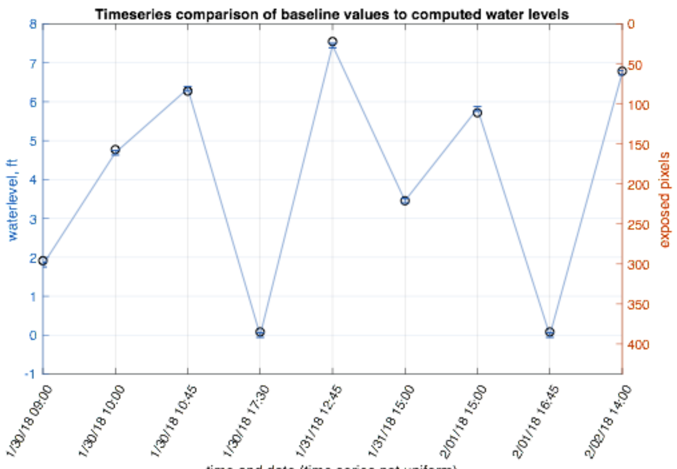


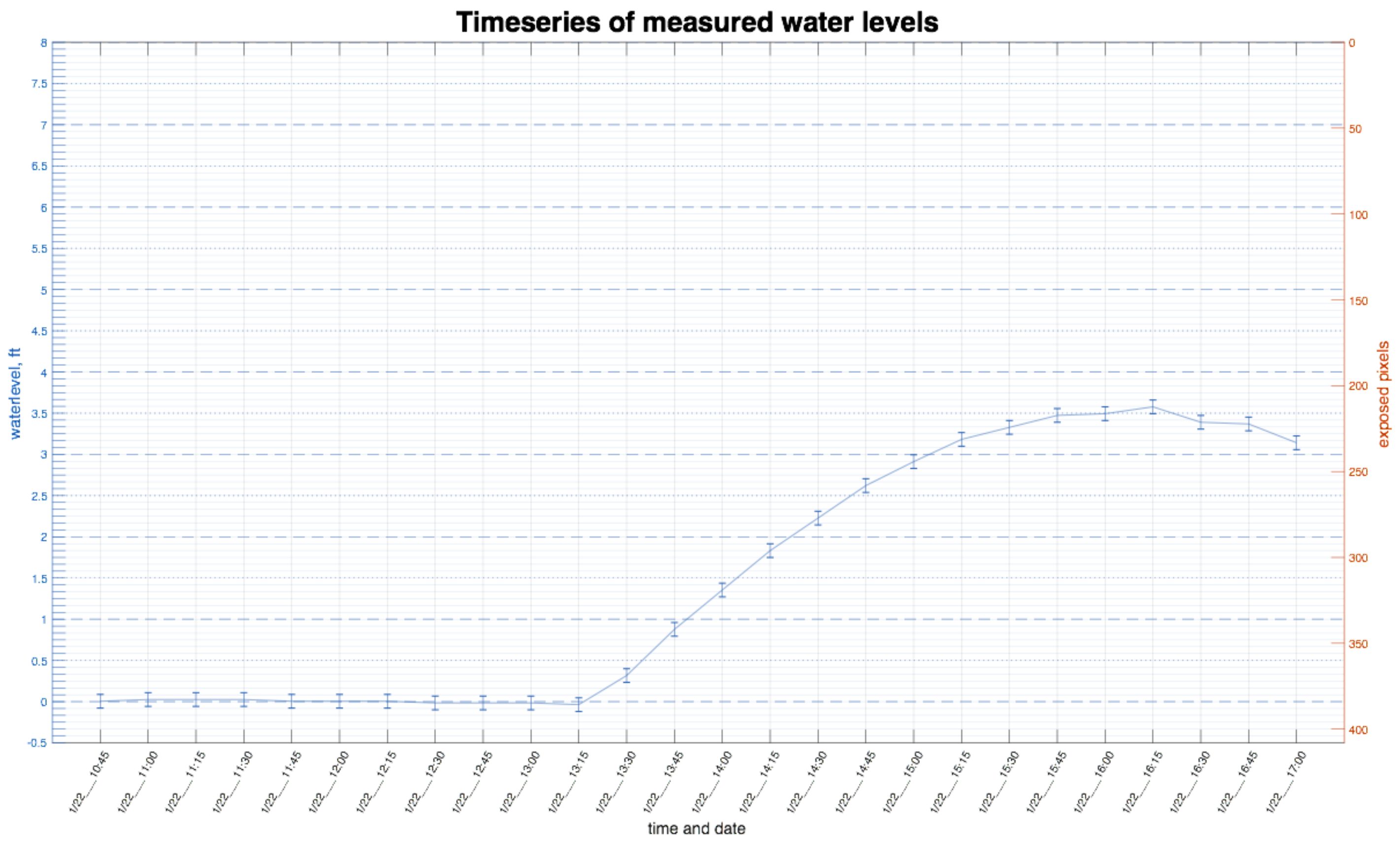
Figure 8 *above*: Baseline time series Results

Figure 9 *below*: Time series test batch of 26 images

Nine baseline images were chosen deliberately to represent a large range of different lighting conditions and water heights. The actual hand-measured heights of the exposed gauge were used for comparison with the algorithm results. Figure 8 displays a line graph of the timeseries comparison of the baseline measurements and the computed water levels.

In the time series results shown in Figure 8, the water height in feet is shown on the left vertical axis and the exposed pixels are quantified on the right vertical axis. The circles in conjunction with the error bars represent the hand taken baseline measurements and the error associated with them. The line displays the output computations from the algorithm.

A second time series is shown in Figure 9 below. This is a test batch of 26 images taken on January 22, 2018 between 10:45 AM and 5:00 PM. The tide in this series is below the stream gauge until 1:15 PM, where the tide rises to a maximum height of 3’7” at 4:15 PM. The tide then begins its descent until darkness falls at 5:00 PM.



Analysis and Discussion

A. *Error Analysis*

From the nine baseline measurements, an uncertainty was derived of ± 4 pixels (±1 in., ±0.08 ft.) due to camera resolution. Plus or minus 2 pixels were from the top of the stream gauge, and ± 2 pixels came from the bottom. This uncertainty will vary from case to case, changing due to differences in camera distance and camera type. In any case, the user will need to specify this pixel uncertainty in the setup process. Figure 10 displays the calculated exposed pixels, the absolute error in units of exposed pixels, and the percent difference all with reference to the measured baseline exposed pixels.

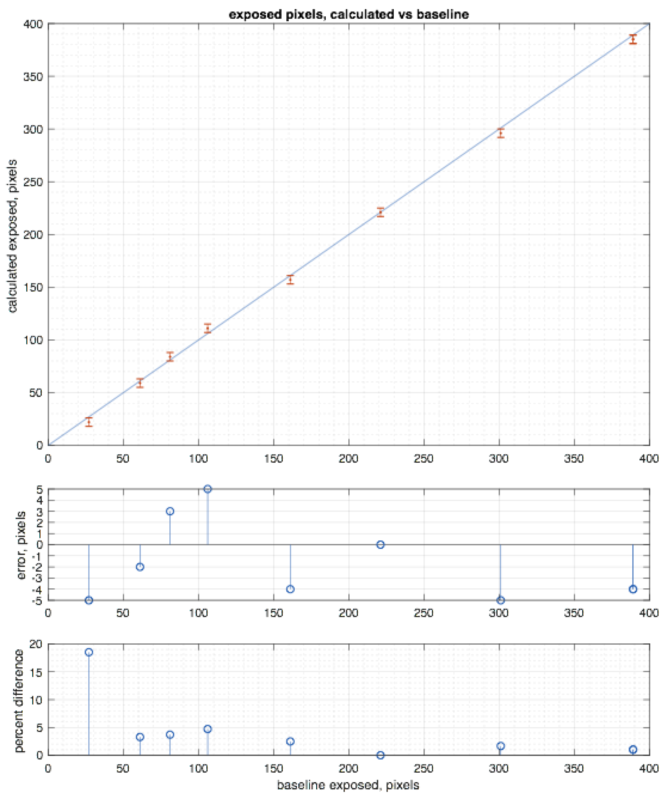


Figure 10: Results with error

The percent difference of the trial results ranges from 0% to 18.5%. Overall, there is an average percent difference of 4%, and a median percent difference of 2.5%. Although the 18.5% difference represents an outlier point, the measurement itself is only one pixel outside range of the 4 pixel uncertainty. The other larger percent differences occur because those measurements were taken for the case where the water level is high and a smaller number of pixels are exposed. This illustrates a systematic sensitivity to error when calculating exposed pixels for high water levels.

As for the error itself, the exposed pixel calculations are for the most part systematically negative. This could be due to the blob erosion process.

B. *Logistical problems*

There are logistical problems related to the placement of the camera in reference to the gauge rectangle. To ensure accurate results with this preliminary version of the software, the camera must be level and point directly at the rectangle as shown in Figures 1 and 2. In situations where this is difficult, it may be easiest to turn the rectangle toward the camera.

If the camera points toward the rectangle at an angle, then the shape of the rectangle might not appear rectangular in the image. If the rectangle lies in the periphery of the image, then the camera lens might introduce distortion.

C. *Hardware and Software Limitations*

Due its simplicity, this method requires strict camera installation guidelines to ensure accurate measurements. If the camera is close to the stream gauge, then the camera must face the rectangle straight-on. This can be accomplished by rotating the rectangle slightly to face the camera. If the camera is far away, then it is permissible to have the camera view from a slight angle.

Night conditions represent a large limitation because the camera employed operates only under visible light conditions. For low light, color recognition becomes impossible and the blob spreads to encompass the entire image as shown in Figure 11. To overcome this, one simple solution would be to install a light to illuminate the rectangular gauge.

A limitation imposed by software requirements is that the camera must remain fixed in the field. If something were to bump the camera, the rectangular gauge would not be in the same position within the image frame. When the software goes to sample the color at the tip of the gauge, it will most likely sample a portion of the image that is not the rectangle itself. This will in turn generate irregular blobs that produce nonsensical measurements. Fortunately this would be an easy fix, for the user will only need to re-calibrate using the setup function.

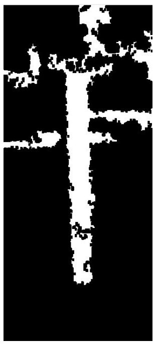


Figure 11: Night Image

Relating to the above case are high water level limitations. If a stream gauge is too short, then high water levels could reach up and impinge on the color sample crop window. This again would lead to nonsensical measurements due to irregular blob patterns. For low water levels, short rectangles might be completely exposed, and measurements will hit and stay at zero. An easy way to avoid problems related to these limitations is to ensure that the colored rectangle is sufficiently long.

Finally, this system only works if the color of the rectangle is uniform. Scum and algae buildup will interfere with the color search algorithm. To avoid this, simple annual maintenance of the gauge will be necessary. This will entail a fresh coat of paint or a good scrub.

Conclusions

Project goals are met as follows:

- A maximally simple system to measure water levels is determined, relying on the combination of specific physical setup requirements, and a MATLAB program that is portable to C.
- The ideal field measurement apparatus is determined to be a simple, long, colored rectangle of comparable length to overall water change. Pointed at this rectangle is any camera capable of wirelessly relaying images with timestamp data.
- MATLAB is used to identify the colored rectangle and calculate the amount exposed, thus providing a quantitative measure of the water level in a given image.
- For setup, a MATLAB user interface is crafted, designed for users to set up the measurement system for any camera/gauge apparatus.
- To maintain simplicity, strong constraints are placed on the possible ways for the apparatus to be set up. This allows, in most cases, for image distortion to be neglected.

The final system is capable of taking typical water level measurements to within 5% difference of actual values. Higher percent differences occur, but only for extremely high water levels. Due to the simplicity, the system is versatile and can be configured to measure both tidal and river water levels.

Although the system studied in this report requires the camera to look head-on at the rectangular gauge, the algorithm employed can extend to include cameras setup at an angle. In these cases the rectangle will appear trapezoidal, but the measurement algorithm will still only require information about the height of the exposed rectangle to generate a value. With the addition of camera calibration, this system will also be able to incorporate poorly aimed camera angles where the rectangle is at the edge of the image frame where distortion becomes an issue.

In the future, due to the simplicity of this program, numerous stream gauges can be integrated using cloud computing. Images can be transmitted wirelessly to cloud space, processed using a small C implementation of the program, then discarded. Information about water levels can then distribute among interested parties such as NOAA, USGS, and private entities.

Appendix A: User Software Setup

The purpose of this appendix is to outline the specific software setup process required of the user, for a given stream-gauge/camera installation.

The user should set up the camera so that the images are ported to a specific network location. It is essential that the camera be configured so that the time stamp from when the photo was taken be stored in the metadata of each photo. Without this, it will not be possible to automatically infer the time of each photo without human observation and data entry.

Next, the user must crop a test image to provide the program with relevant information about where to search within an image. The setup program should be run once on an image for which the entire gauge is showing. This will ensure that the cropping window is taken over the correct area for the specific use case. The first cropping window is meant to isolate the rectangular stream gauge from the rest of the image. In the event that the stream gauge is partially submerged, the user must provide an image wherein the exposed length of the rectangle is known from a field measurement.

Additionally during this cropping step, the user will specify a small window with which the color threshold will be determined. It is very important that this window be a section of the gauge that is very near the top and will never be covered by water. This ensures that the color detected will always be an average of the gauge color, specific to the distinct lighting conditions of each individual image. After doing this once, it will not need to be done again for each camera-gauge pair unless the system is disturbed.

After this information is attained, the setup program saves the data to a .mat file to be accessed by the main image measurement program. At this point, the measurement program can be customized so that each time a new image is added to the folder, the software determines the height of the stream and computes the water level. Alternatively, it can be configured to run a batch of images from a specific time interval.

Appendix B: Camera Calibration

The purpose of this appendix is to discuss in more detail the relevance of camera calibration, and also to discuss available avenues to take in the case that calibration is necessary.

Calibration removes distortion caused by camera lens aberrations and sensor misalignment. Calibration does *not* remove distortion caused by the geometric difference in an image from its true shape due to harsh angles of camera perspective. For example, calibration will remove the bubble-like distortion caused by the camera lense. But if a rectangle looks like a trapezoid due to viewing it at an angle, it will be impossible to make the trapezoidal image appear as a perfect rectangle, as if looking head-on.

For the large majority of cases, camera calibration will be completely unnecessary. In fact, as discovered in our case, camera calibration is not feasible from large distances. Attempting to force calibration can make distortion worse. For most cases, the distance between the camera and the stream gauge should be large enough to render distortion insignificant. This is the case for the Whittier test setup. Regardless, at smaller distances results should remain insensitive to distortion. This is because we use an algorithm that relies on lengthwise linear distance to determine water levels. Initially, we attempted to use exposed surface area do deduce water levels, but it became clear that this would expose our measurements to unnecessary error due to distortion.

Every camera has a unique calibration, because every camera and lense is different. To calibrate, one must gather many images of a uniform checkered grid from many different angles and distances. The more variation in the calibration set, the better the calibration will be. In practice, the images must be quite close for distortion to become measurable. These images of the grid are then processed, by identifying all the square corners and specifying the actual size of each square. The calibration software then stitches together these images to recreate the 3D space containing the camera and grids. This is summarized by calibration data.

This process is something that if needed, can be performed in a lab. The favorable approach to this is to use the *Camera Calibration Toolbox for Matlab*, by Jean-Yves Bouguet from Caltech [8]. After performing the calibration, the calibration data is to be added as a .mat file to the main program folder for a particular stream gauge. The calibration step is then to be activated within the main measurement program. For each run, before manipulating the image, the calibration data will be applied to remove image distortion.

In order for the calibration to work, the grid used in the lab for calibration must be mounted immediately above the stream gauge. To be able to remove distortion, the calibration software must first identify the grid and calculate distances for the actual grid location in the field. Only after identifying the grid and calculating the extrinsic properties can the software remove distortion in an image.

Table of Figures

Figure 1: Apparatus - Side View
Figure 2: Apparatus - Aerial View
Figure 3: Software Block Diagram
Figure 4: Blobby Edges
Figure 5: Gradient Potential - MATLAB
Figure 6: Test Image Gradients
Figure 7: Eroded Image
Figure 8: Time series Results
Figure 9: 26 image test batch
Figure 10: Results with error
Figure 11: Night Image

References

[1] Alaska Department of Fish and Game, "Rivers and Lakes," *Alaska Department of Fish and Game,* 2018. [Online]. Available: http://www.adfg.alaska.gov/index.cfm?adfg=rivers.main [Accessed: Mar. 1, 2018]

[2] National Oceanic and Atmospheric Administration, "Water," *National Weather Service*, 2018. [Online]. Available: https://water.weather.gov [Accessed: Apr. 1, 2018]

[3] The USGS Water Science School, "How stream height relates to streamflow," *U.S. Geological Survey*, 2016. [Online]. Available: https://water.usgs.gov/edu/gageflow.html [Accessed: Apr. 14, 2018]

[4] S. Mauger, "Kenai Peninsula Fish Habitat Partnership NFHAP Proposal," *Cook Inletkeeper*, unpublished.

[5] North Dakota Water Science Center, "How Does USGS Collect Streamflow Data?," *U.S. Geological Survey*, 2008. [Online]. Available: https://nd.water.usgs.gov/gage/how2.html [Accessed: Apr. 14, 2018]

[6] MathWorks, Natick, MA, USA, *Documentation, What Is Camera Calibration?*. (2018) [Online]. Available: https://www.mathworks.com/ help/vision/ug/camera-calibration.html [Accessed: Feb. 1, 2018]

[7] MathWorks, Natick, MA, USA, *Documentation, imgradient*. (2018) [Online]. Available: https://www.mathworks.com/help/images/ref/imgradient.html [Accessed: Apr. 14, 2018]

[8] J. Bouguet. *Camera Calibration Toolbox for Matlab*. (2015) [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/ [Accessed: Feb. 1, 2018]