Open-Source Software for Water-Level Measurement in Images With a Calibration Target

K. W. Chapman

T. Gilmore
University of Nebraska - Lincoln

C. D. Chapman

F. Birgand

A. Mittelstet
University of Nebraska - Lincoln

See next page for additional authors

Authors
K. W. Chapman, T. Gilmore, C. D. Chapman, F. Birgand, A. Mittelstet, M. Harner, M. Mehrubeoglu, and J. E. Stranzl Jr.

ADVANCING EARTH AND SPACE SCIENCE

Water Resources Research®

METHOD

10.1029/2022WR033203

Technical Note: Open-Source Software for Water-Level Measurement in Images With a Calibration Target

Kenneth W. Chapman[1], Troy E. Gilmore[1], Christian D. Chapman[2], François Birgand[3], Aaron R. Mittelstet[1], Mary J. Harner[4], Mehrube Mehrubeoglu[5], and John E. Stranzl Jr.[1]

[1]University of Nebraska Lincoln, Lincoln, NE, USA, [2]MIT Lincoln Laboratory, Lexington, MA, USA, [3]North Carolina State University, Raleigh, NC, USA, [4]University of Nebraska at Kearney, Kearney, NE, USA, [5]Texas A&M University-Corpus Christi, Corpus Christi, TX, USA

Key Points:

- Measurement of water level in images requires waterline detection and conversion of the found position from pixel to world coordinates
- A free, commercial friendly, open-source software package that automates image analysis for scenes with a calibration target is introduced
- Trade-offs between software complexity and ease of use, and potential future improvements by the hydrologic community are highlighted

Correspondence to:

K. W. Chapman,
kchapman12@huskers.unl.edu

Citation:

Chapman, K. W., Gilmore, T. E., Chapman, C. D., Birgand, F., Mittelstet, A. R., Harner, M. J., et al. (2022). Technical note: Open-source software for water-level measurement in images with a calibration target. *Water Resources Research*, *58*, e2022WR033203. https://doi.org/10.1029/2022WR033203

Received 6 JUL 2022
Accepted 8 JUL 2022

Abstract Image-based water level measurements offer data quality assurance through visual verification that no other method can provide. GaugeCam Remote Image Manager-Educational 2 (GRIME2) is a mature, open-source commercial friendly software application that automatically detects and measures water level in laboratory and field settings. The software relies on a dedicated target background for water line detection and image calibration. The system detects the change in pixel gray scale values associated with the intersection of the water level at the target surface. Fiducials on the target background are used to precisely create a pixel to real world coordinate transfer matrix and to correct for camera movement. The presented software package implements the algorithms and automates the water level measurement process, annotation of images with result overlays, creation of animations, and output of results to files that can be further analyzed in a spreadsheet or with R or Python. These GRIME2 features are illustrated using imagery from a coastal marsh field site. Tradeoffs between workflow and algorithm complexity and ease of use are discussed and future improvements are identified with the intention that this Findable, Accessible, Interoperable, and Reusable-inspired software can be adopted, modified and improved by the user community. While image resolution, quality and other factors associated with field deployment (e.g., water surface roughness, sun glare, shadows, and bio-fouling) will have an impact on measurement quality, previous controlled laboratory testing that did not manifest these issues showed potential for accuracy of ±3 mm (Gilmore et al., 2013, https://doi.org/10.1016/j.jhydrol.2013.05.011).

1. Introduction

Water level in streams and rivers is one of the most fundamental hydrological measurements, and has been automated for a long time (Boiten, 2008). Early automation relied on mechanical recorders where levels were continuously logged with a pen on paper. Now, just about all water stage data are acquired using electronic sensors and numerically stored at the minute scale. Automated water level measurement approaches are thus well-suited to capture shorter-term and stochastic water level dynamics in most situations. However, all monitoring instruments do not directly measure depth but actually measure physical properties such as voltage or time, which are then translated into water levels using an embedded rating system. Because of exposure to the elements, the rating system tends to drift over time, requiring frequent on-site instrument calibration using manual measurements as a reference. Additionally, automated approaches provide only a sensor response, without contextual information about the conditions (e.g., presence of ice, obstructions, and debris) at the monitoring site. Even in the case where redundant sensors are installed to collect simultaneous measurements, it may be impossible to determine which sensor is correct.

Image-based water level measurement, within limitations, offers the benefits of both manual (visual) and automated measurement of water level (we will refer to manual and automated methods described above as "traditional" methods). Time-lapse imagery of a lake or stream can be used to extract both quantitative information (water level) and qualitative information (site conditions relevant to water level). The potential of image-based water level measurement is tantalizing. Many studies have focused on image analysis algorithms to measure water level (e.g., Chakravarthy et al., 2002; Gilmore et al., 2013; Kaplan et al., 2019; Lin et al., 2018; Noto et al., 2022; Schoener, 2018; Takagi et al., 1998) and/or suggested that image-based water level measurement is a viable method for hydrologic studies (e.g., Hamel et al., 2018; Tauro et al., 2018). However, to our knowledge, there is no fully functional open-source software package dedicated to the task of routine, image-based water level measurement with vertical accuracy typical for scientific studies (Harmel et al., 2006) and/or approaching the vertical accuracy standards of the United States Geological Survey (0.01 feet (3 mm) or 0.2% of effective stage (Turnipseed and Sauer, 2010)).

The objective of this paper is to describe the GaugeCam Remote Image Manager-Educational 2 (GRIME2) program that has been released as free, commercial friendly, open-source software. It should be noted that the ”2” in GRIME2 does not represent a program version, rather it is the name chosen to differentiate the open source project from the original GRIME software which was developed as lab software for a specific research project (Gilmore et al., 2013) and was not user friendly. The version of the program discussed in this paper is GRIME2 v0.1.0.4. It has been refined over the course of three studies: (a) a laboratory study of uncertainty in image-based water level measurement (Gilmore et al., 2013), (b) a field study that used the GaugeCam system (Etheridge et al., 2015), and (c) a robust field comparison with traditional sensor measurements (Birgand et al., 2022). These studies showed, respectively, the (a) ability to measure water level with an accuracy of ±3 mm under tightly controlled laboratory conditions, (b) ability to deploy the GaugeCam system for reliable field measurements at a remote research site, and (c) ability to measure within ±5 mm (90% confidence interval) over a range of about 1 m in a tidal creek.

Overall, GRIME2 meets two principal challenges: (a) detection of the water line in the captured images containing a specialized target background in the image scene and (b) conversion of the found position of the water line in the images as calculated in pixel units to a water level in physical world coordinate units such as inches or centimeters. While the algorithms presently implemented in GRIME2 were successfully used in previous studies, our hope is that others will either contribute improvements or replacements of those algorithms to the GRIME2 project or co-opt the program, whole cloth, to make whatever changes their research requires. The most recent installers can be obtained via Gilmore (2021) or Chapman (2021).

2. Materials and Methods

The focus of this technical note is the presentation of the methods to measure water level accurately and repeatably in images, but the GRIME2 project also includes materials to help users correctly set up the camera and calibration target to capture images the software can process. The GRIME2 programs and libraries were developed in C++ using commercial friendly, open-source libraries and programs that include OpenCV (OpenCV, 2019), Boost (Boost, 2021), ExifTool (Harvey, 2021), Qt (Qt, 2021), and ffmpeg (ffmpeg, 2021). The repository for the software is on GitHub (Chapman, 2021). There are tutorials and further explanatory information on camera and target setup at the GaugeCam website (Gilmore, 2021). All of these materials are also available in the article Digital Object Identifier (DOI) (Chapman & Gilmore, 2022).

GRIME2 automates tasks of interest to users who want to measure water level in multiple images and then output results for research or visual evaluation. With the application of a few simple procedures, the program can accomplish the following tasks:

1. Install a graphical user interface (GUI) version of the program on a Microsoft Windows computer
2. Create pixel to world coordinate calibration models (not to be confused with hydrology calibration models)
3. Measure water level for a single or multiple images
4. Output the water levels to a comma separated values file for analysis
5. Create images with overlays that show the found position of the waterline suitable for publication
6. Create animations of original images or images with overlays suitable for use on web sites
7. Run all functionality of the program from the command line or from within batch files to measure water level in archived images

The current manifestation of the program and algorithms was used successfully in three published studies. The GRIME2 workflow and algorithms were developed and evaluated in a laboratory study at North Carolina State University (NCSU) (Gilmore et al., 2013). In this study, measurements of water stage were reported to an accuracy of ±3 mm under tightly controlled laboratory conditions with a wireless security camera (Microseven Systems M7-RC550WS, now obsolete) positioned directly in front of a 2 m tall target and at approximately 3 m above bottom of the target and 8 m from the target to the camera. The camera had an 8 mm lens. GRIME2 was also used successfully in a field study in the North Carolina tidal marsh as a redundant system to measure stage

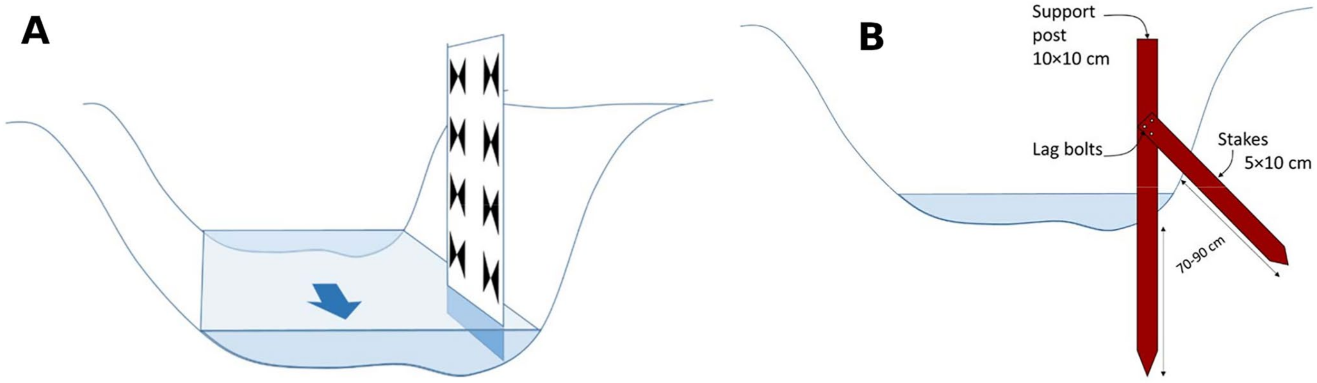


Figure 1. Image A is a representation of the dedicated target installed in a small (to keep size of figure small) stream. Image B is a suggested method for bracing and maintaining the target in the field so that it maintains orthogonality with the water surface.

(Birgand et al., 2022 and Etheridge et al., 2015). The camera used in the tidal marsh study was a Lookout V wireless internet camera from Colorado Video (http://www.colorado-video.com). It was equipped with an infrared cut filter during the daytime. The filter was mechanically removed at night to allow the IR to pass to the sensor. The camera was fitted with a weatherproof housing and mounted on a heavy metal pole. The LED IR illuminator was installed near the camera (about 70 cm to one side) to avoid direct glare on the target background at night. The camera was about 5.2 m away (horizontally) from the background target and about 1.6 m above ground.

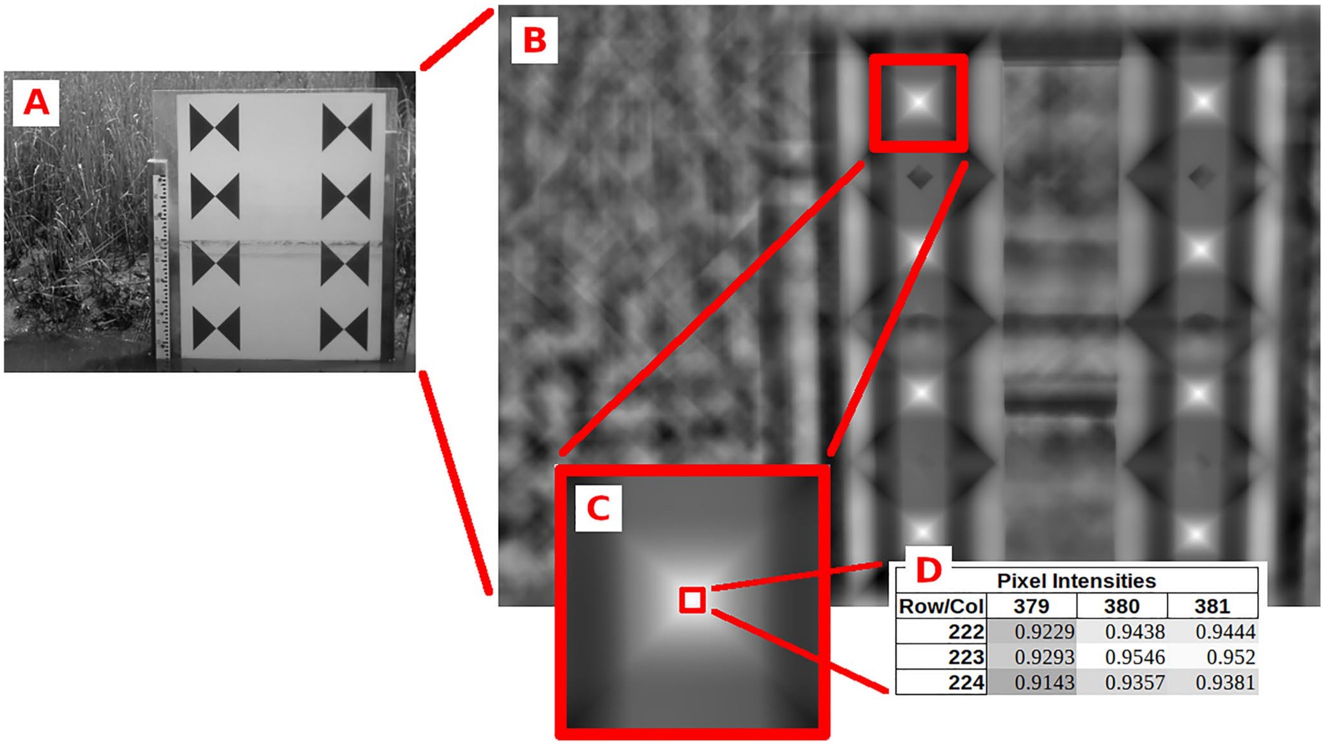
The system was designed with open source libraries to provide a software framework to automate the measurement of water stage for research, archival, and educational purposes. It should be noted that this paper is not about any specific camera, lighting, optical system, or other hardware or site specific differences. The GRIME2 licenses were chosen and the software architecture designed to be modified to perform algorithm research, test sources of variability in cameras, optical system, target mountings, target types, site specific considerations, accommodate different result output needs, etc.

In its current manifestation, GRIME2 requires a dedicated target installed in the field (i.e., a vertical white plane), against which waterline can be detected in images. The background target also contains bow-tie fiducials to establish the transfer matrix between pixel to real world coordinates (details below). The target background must be installed so that it is as orthogonal to the surface of the water as possible and the entire face of the target appears in images captured by the camera (see Figures 1 and 2a). The plane of the bow-tie target and the lines passing through each vertical column of bow-ties are expected to be orthogonal to the surface of the water.

The selection of algorithms, target patterns, result output formats, and other aspects of the program were driven by multiple, sometimes competing factors that required compromises. An example of such a compromise was the selection of the pixel to world and world to pixel calibration methodology described in Section 2.1. Other calibration techniques would have worked as well or better in terms of accuracy and accommodation of variability, but were not selected due to a heavier burden placed on end-users to create a calibration in the field, the needs of the specific site in the North Carolina tidal marsh for which the program was initially developed, ease of target creation, and potential for one button push calibration. Those benefits were offset by the need to rigidly place the target as orthogonally as possible in the water. Other calibration schemes would have required other compromises. Future research could include additional calibration methods, target patterns, result output formats, water line finding algorithms, and image preprocessing so the end user can make the compromises of their choice.

2.1. Calibration Image

Among all the images obtained, one image is first set aside to be the *calibration image*. It is usually chosen because of its quality and it contains all the fiducials. These methods expect the surface of the bow-tie calibration target to be orthogonal to the surface of the water. The calibration algorithm does four things in the *calibration image*: (a) Identifies the center of each calibration target bow-ties (Figures 2 and 3), (b) creates pixel-to-world and world-to-pixel coordinate homographies, (c) sets a water level region of interest (ROI) (the blue rectangle in



Pixel Intensities			
Row/Col	379	380	381
222	0.9229	0.9438	0.9444
223	0.9293	0.9546	0.952
224	0.9143	0.9357	0.9381

Figure 2. Image A is a typical bow-tie target image suitable for calibration as none of the bow-ties are below the water line. Image B is the best response image (the brighter the pixels in the response image corresponding to the bow-tie center positions, the stronger the template match) out of the set of response images created for each of the bow-tie templates shown in Figure 3. The eight brightest spots in Image B are the high template response that occurs at the center of each of the bow-tie patterns. Image C is the zoomed in area of the response image at the center of the second bow-tie from the top in the first column. Image D is further zoomed in to show the pixel columns and rows of the 3 × 3 area around the brightest pixel in Image C. The positions and the intensities at the center of Image D are used to calculate the subpixel position of the center of the bow-tie.

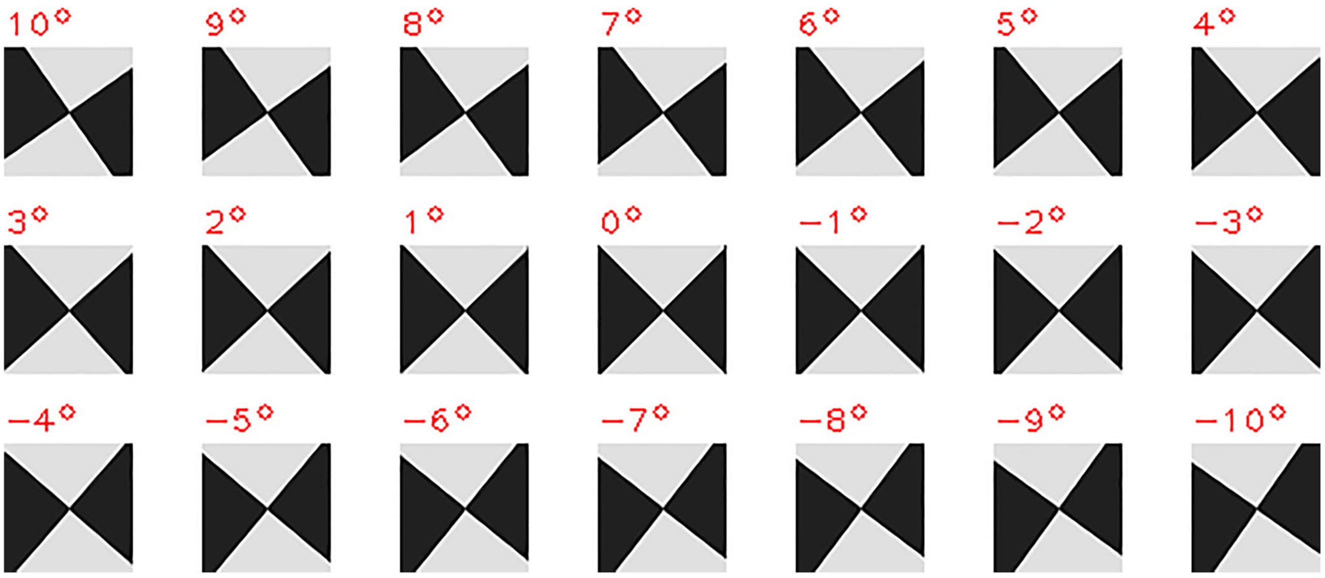


Figure 3. The bow-tie at the center of this figure labeled 0° is created synthetically, then rotated to create templates at 1° increments from −10° to 10°. These templates are used to search the calibration target image for the centers of all the bow-ties, returning the subpixel position of the center of each bow-tie.

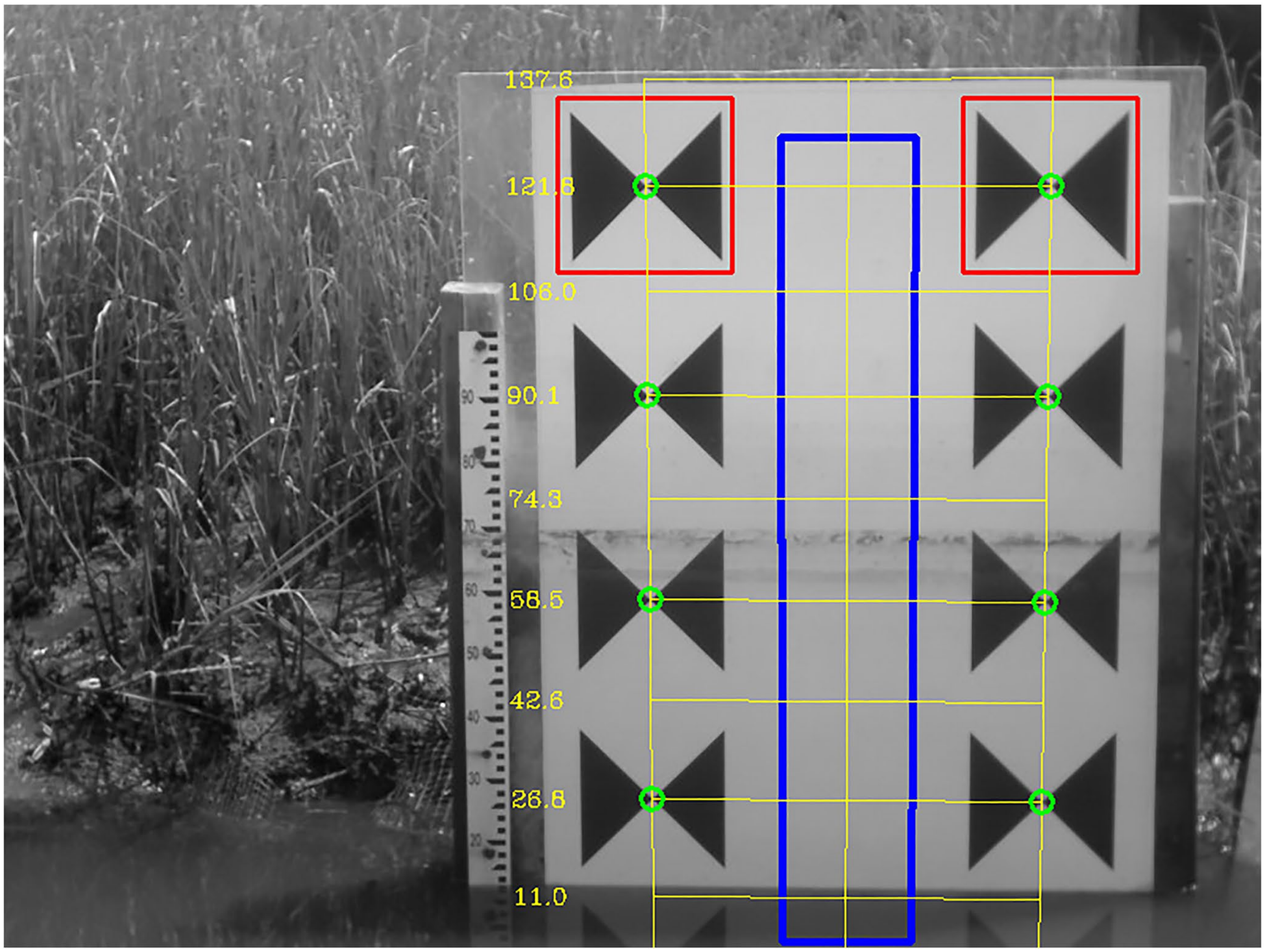


Figure 4. Overlay image that shows the results of a calibration including the found bow-tie grid in yellow, the world coordinates of the horizontal grid lines in yellow, the water line search area in blue, and the movement target search areas in red.

Figure 4) within which the water level search is performed, and (d) sets a ROI around the top two bow-ties (the red rectangles in Figure 4).

The vertical water range within which water level can be measured is set by item #3 as is shown by the blue rectangle in Figure 4. Water levels above or below the blue rectangle cannot be measured. The actual world coordinate range of the measurement area is dependent on the physical size of the calibration target. The ROIs set in item #4 are used to quantify the apparent movement of the target background (due in reality to camera movement) compared to that in the reference image, and to correct for this movement in the final calculated stage (see Section 2.1.3). The parameters for the calibration model and the positions of the water level search and movement detection ROIs are written to a calibration file that is loaded when a water level measurement is performed.

2.1.1. Finding Bow-Ties in the Image

The calculation of the pixel positions for each bow-tie on the calibration image is required to create the coordinate transform matrix and record the reference position of the top two bow-tie fiducials (details below). On the calibration image, all bow-ties must be visible to find the fiducial centers and thus their coordinates. A synthetic template of the shape of the bow-tie center is created and then rotated in 1° increments from −10° to 10° (see Figure 3). The rotations are necessary in case the angle of the bow-ties in each column are not parallel with the vertical columns of pixels in the captured images.

OpenCV's normalized cross correlation template match algorithm is run using the rotated template set on the calibration target image to create a “match space” as shown in Figure 2. This image is searched to find the eight highest intensity pixels that have the same spatial relationship as the centers of the bow-ties. The neighboring pixels around the found bow-tie centers are evaluated to calculate their subpixel position. Subpixel position of the bow-tie center is calculated in Equation 1:

$$u_{\text{subpixel}} = \frac{\sum_{y=u-1}^{u+1} \sum_{x=v-1}^{v+1} x i_{yx}}{\sum_{y=u-1}^{u+1} \sum_{x=v-1}^{v+1} i_{yx}} \qquad v_{\text{subpixel}} = \frac{\sum_{y=u-1}^{u+1} \sum_{x=v-1}^{v+1} y i_{yx}}{\sum_{y=u-1}^{u+1} \sum_{x=v-1}^{v+1} i_{yx}} \tag{1}$$

where r equals the row position and c equals the column position of the pixel at maximum intensity at a bow-tie center. These bow-tie pixel positions are used for the calculation of the pixel to world coordinate homography, the world to pixel coordinate homography, and the reference position for target movement detection. The subpixel estimation is a simplistic routine that could be easily improved by application of a Gaussian kernel or other subpixel techniques such as correlation interpolation, intensity interpolation, differential method, and phase correlation described in Tian and Huhns (1986).

There are additional good candidates for future releases of the program to search for the bow-ties and other patterns that could be used for calibration. These include algorithms like generalized Hough transform (Ballard & Brown, 1982) and commercial geometric pattern search tools similar in nature to Cognex's PatMax (Cognex, 2022) or Martox's Geometric Model Finder (Matrox, 2022) routines.

2.1.2. Pixel to World Coordinate Calibration Model

The pixel-to-world coordinate calibration model constructs pixel-to-world and world-to-pixel coordinate homographies and performs perspective transforms to convert between image coordinates and world coordinates. These maps are created using two data sets:

- 2D world coordinates of the eight bow-tie centers on the calibration target's face. These coordinates are specified by the user in a comma separated values (CSV) file with eight records. Each record contains horizontal and vertical displacements in cm of a bow-tie's center from an arbitrary zero point.
- 2D image (pixel) space coordinates (image row and column indices) corresponding to the bow-tie centers. The accuracy of these coordinates directly affect the accuracy of the calibration model. These coordinates are calculated when GRIME2's calibration button is pressed using a call to OpenCV's `matchTemplate` routine (OpenCV, 2019).

It is important to note that, with the current calibration algorithms, care must be taken to assure the target is rigid and the lines passing through the top and bow-ties in each column are orthogonal to the water surface. Error is introduced if the target is not orthogonal to the surface of the water. Future versions of the program would benefit from other calibration methods such as OpenCV's (OpenCV, 2019) `calibrateCamera` function that would better accommodate a calibration target that is not orthogonal to the water but would impose a heavier burden on the end user because each bow-tie center world coordinate point would need to be measured in three dimensions rather than just the two that are required for an orthogonal target.

In general, writing the horizontal and vertical displacements as (x, y) and corresponding image rows and columns as (u, v), the produced coordinate transform is characterized by some matrix $M_{\text{world}\to\text{image}} \in \mathbb{R}^{3\times3}$ with the property:

$$M_{\text{world}\to\text{image}} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} u \cdot f \\ v \cdot f \\ f \end{bmatrix} \tag{2}$$

Image coordinates (u, v) are obtained by dividing the outcome of the matrix multiplication by its third coordinate, f. The scale term f is necessary to give the coordinate transform a matrix representation.

The reverse transform, taking image coordinates into world coordinates uses the world to pixel coordinate homography. In particular, applications of the obtained perspective transforms are implemented by standard OpenCV routines (OpenCV, 2019). Figure 5 shows the calibration process in a flowchart.

The calibration target must remain planar for the measurements to be accurate. If the target becomes warped (e.g., by water pressure or an animal), accuracy will diminish. At calibration time, the root mean square error between found pixel positions of the centers of the bow-ties and those same pixel positions after they have been transformed to world coordinates and then re-projected back to image coordinates is calculated for all the pixel positions in the water level search region. An error threshold could be set to flag images that could contain a nonplanar target. Appendix A holds a table that shows this root mean square error (RMSE) for bow-tie target center pixel positions after they have been transformed to world coordinates and then reverse transformed back to image coordinates using the described calibration model. The images were taken as part of the Etheridge study (Etheridge et al., 2015) in the North Carolina tidal marsh. This number will become more important when three dimensional projections that make use of the camera intrinsic and extrinsic parameters are implemented.

2.1.3. Move Target Regions of Interest

With proper camera installation, the camera should be relatively stable. However, any slight changes in camera position (due to wind, soil moisture change around camera post, etc.) can have an impact on the final reading, which needs to be corrected. For the calibration image, the coordinates of the top two fiducials are recorded in the calibration file. For each run-time image, the positions of the top two bow-ties are calculated and compared to the coordinates of those of the calibration image, the difference of which is calculated to quantify the vertical movement of the camera (i.e., magnitude and offset). Obviously, to make these corrections, working images must have the top two bow-ties above the water line to perform the search. Figure 6 shows the result of the camera move adjustment calculation. If the top two targets are not in the search window either because they are obscured by high flow or large camera and/or target movement then the move calculation is not run, no adjustments are made and a warning is logged.

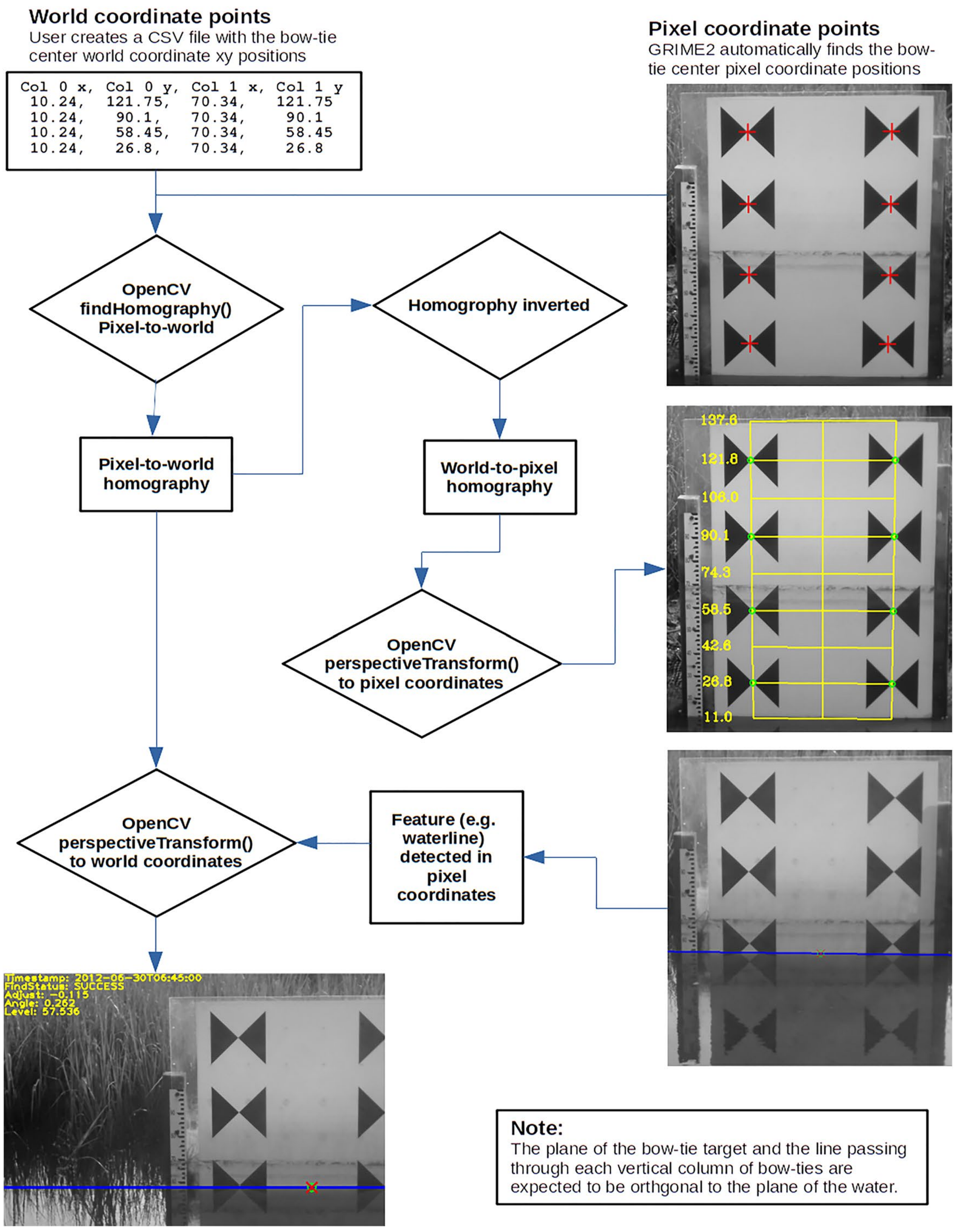
Change in the physical position of the camera and calibration target are likely to diminish the water level measurement accuracy.

2.2. Find Water Level in an Image

The blue rectangle shown in Figure 4 is the waterline search region. It is set automatically during the calibration procedure based on the found positions of the bow-ties and evaluated for a run-time image to find the water level. In a future version of GRIME2, it could be possible for a user to extend or move the search area as long as it remains on the same plane as the calibration target. Preprocessing is performed on the waterline search region to deal with issues such as biofouling, low turbidity (very clear) water, glint, turbulence, and other such issues. Combinations of morphology, Gaussian blurs and other smoothing, contrast limited adaptive histogram equalization (CLAHE), pyramid mean shift filtering, and non-linear contour detection and suppression were explored. A dilation followed by an erosion of the water line search area with a 1 × 9 (vertical) morphological kernel was selected as it was most effective on the North Carolina tidal marsh images used as a development data set. Figure 7 shows the result of the preprocessing that removes most of the biofouling while preserving the waterline. Flexibility in selection of kernel shapes and sizes and alternative preprocessing for different water scenes are subjects for future research and addition to GRIME2.

The steps include: write the $M \times N$ sub-image corresponding to this rectangle $\mathbf{R} \in \mathbb{R}^{M \times N}$. The search area is partitioned into N_r equally sized narrow vertical sub-regions (by default $N = 10$) shown in Figure 8 Image B in which we write as tall matrices $\mathbf{R}_1, \ldots, \mathbf{R}_{N_r} \in \mathbb{R}^{M \times (N/N_r)}$:

$$\mathbf{R} = \left[\mathbf{R}_1, \ldots, \mathbf{R}_{N_r}\right] \in \mathbb{R}^{M \times N}. \tag{3}$$


World coordinate points
User creates a CSV file with the bow-tie center world coordinate xy positions
Pixel coordinate points
GRIME2 automatically finds the bow-tie center pixel coordinate positions
OpenCV findHomography() Pixel-to-world
Homography inverted
Pixel-to-world homography
World-to-pixel homography
OpenCV perspectiveTransform() to pixel coordinates
OpenCV perspectiveTransform() to world coordinates
Feature (e.g. waterline) detected in pixel coordinates
137.6
121.8
106.0
90.1
74.3
58.5
42.6
26.8
11.0

```
Col 0 x, Col 0 y, Col 1 x, Col 1 y
 10.24,  121.75,   70.34,   121.75
 10.24,    90.1,   70.34,     90.1
 10.24,   58.45,   70.34,    58.45
 10.24,    26.8,   70.34,     26.8
```

```
Timestamp: 2012-06-30T06:45:00
FindStatus: SUCCESS
Adjust: -0.115
Angle: 0.262
Level: 57.536
```

Note:
The plane of the bow-tie target and the line passing through each vertical column of bow-ties are expected to be orthgonal to the plane of the water.

Figure 5. The bow-tie calibration target is installed orthogonally in the water so that both the plane of the target and the vertical lines passing through the centers of each bow-tie column are orthogonal to the surface of the water. The world coordinate positions of the centers of the bow-ties are measured and recorded in a comma separated values file. The image coordinates of the bow-tie centers are calculated using the process described in Section 2.1.1. The OpenCV perspectiveTransform function is called using the homography to transform the image coordinates of an item of interest (e.g., water level) to world coordinates. The homography is inverted to create a world to pixel coordinate homography used for testing calibration model error and to draw world coordinate features such as the calibration grid as overlays on the images.

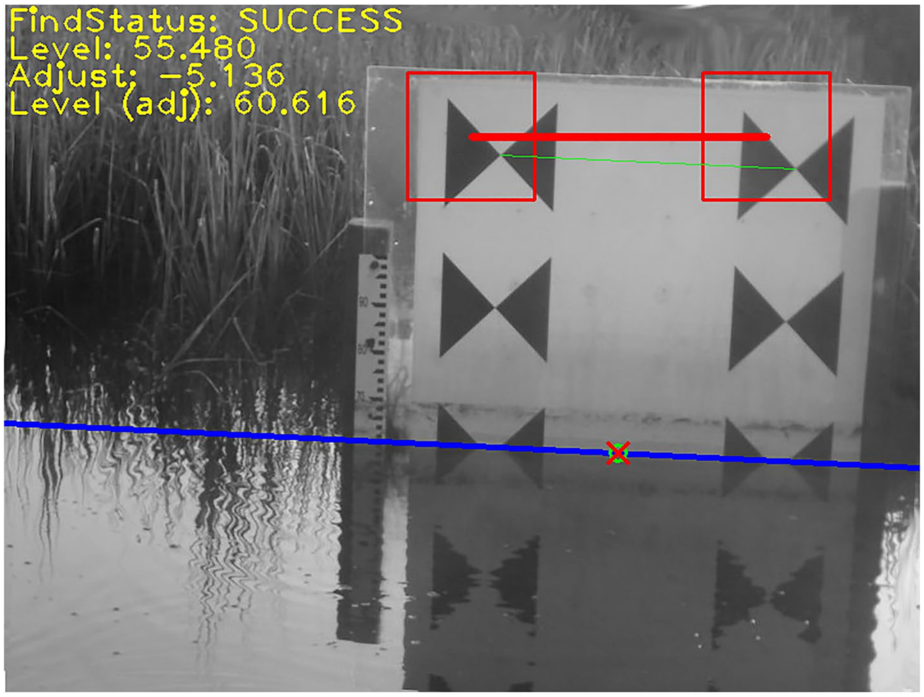


Figure 6. Move offset calculation using an exaggerated movement for illustration purposes. The red rectangles show the top two bow-tie search regions of interest as defined in the calibration image. The green line shows the position of the bow-ties as they were found during the water level search on the *run-time image*. The red line shows the positions of the bow-ties on the calibration image. "Level" is the water level before move adjustment. "Adjust" is the amount of adjustment calculated by the move offset calculation. "Level (adj)" is the final water level with move adjustment applied.

A vector of sums of pixel intensities for each row in these narrow sub-regions is calculated to yield N_r column vectors $\vec{S}_1, \ldots, \vec{S}_{N_r} \in \mathbb{R}^{M\times 1}$:

$$\vec{S}_n = \begin{bmatrix} \sum_{t=1}^{N/N_r} [\mathbf{R}_n]_{1,t} \\ \vdots \\ \sum_{t=1}^{N/N_r} [\mathbf{R}_n]_{M,t} \end{bmatrix} \in \mathbb{R}^{M\times 1}, \quad n = 1, \ldots, N_r. \tag{4}$$

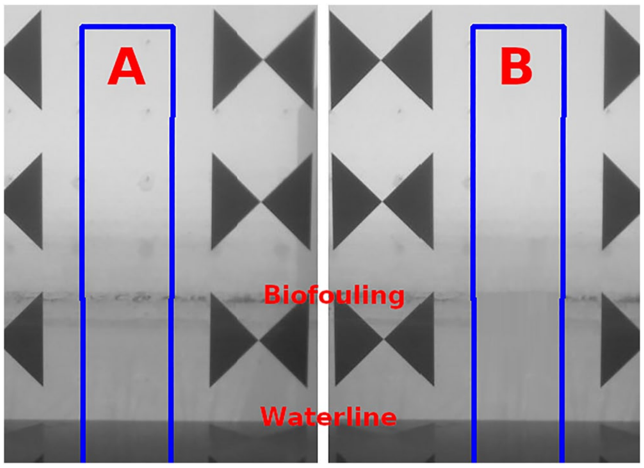


Figure 7. Image A shows an image with biofouling in the waterline search region. Image B shows the same image after preprocessing to remove most of the biofouling while preserving the waterline.

The leftmost yellow line shown in Figure 8 Image C is the sum of pixel intensities for one of the narrow sub-regions. The largest change in value from row to row from light (typically the background of the bow-tie target) to dark (typically the water) is identified from each $\vec{S}_n$, $n = 1, \ldots, N_r$. The x-coordinate for each sub-region is the horizontal center of the sub-region at the water line. That is, we define N_r 2D coordinates $\left(x_{\mathrm{edge},1}, y_{\mathrm{edge},1}\right), \ldots, \left(x_{\mathrm{edge},N_r}, y_{\mathrm{edge},N_r}\right) \in \mathbb{R}^2$:

$$x_{\mathrm{edge},n} = \left(n - \frac{1}{2}\right)\frac{N}{N_r}, \tag{5}$$

$$y_{\mathrm{edge},n} = \arg\max_{k\in\{2,\ldots M\}} \left[\vec{S}_n\right]_{k-1} - \left[\vec{S}_n\right]_k, \quad n = 1, \ldots, N_r. \tag{6}$$

For each $n = 1, \ldots, N_r$, a subpixel edge location estimate $y_{\mathrm{sub},n}$ is formed by using linear interpolation to find the zero-crossing of the second derivative of $\vec{S}_n$ past row $y_{\mathrm{edge},n}$:

$$y_{\mathrm{sub},n} = y_{\mathrm{edge},n} - \frac{\left[\Delta^2 \vec{S}_n\right]_{y_{\mathrm{edge},n}}}{\left[\Delta^2 \vec{S}_n\right]_{y_{\mathrm{edge},n+1}} + \left[\Delta^2 \vec{S}_n\right]_{y_{\mathrm{edge},n}}}, \quad n = 1, \ldots, N_r \tag{7}$$

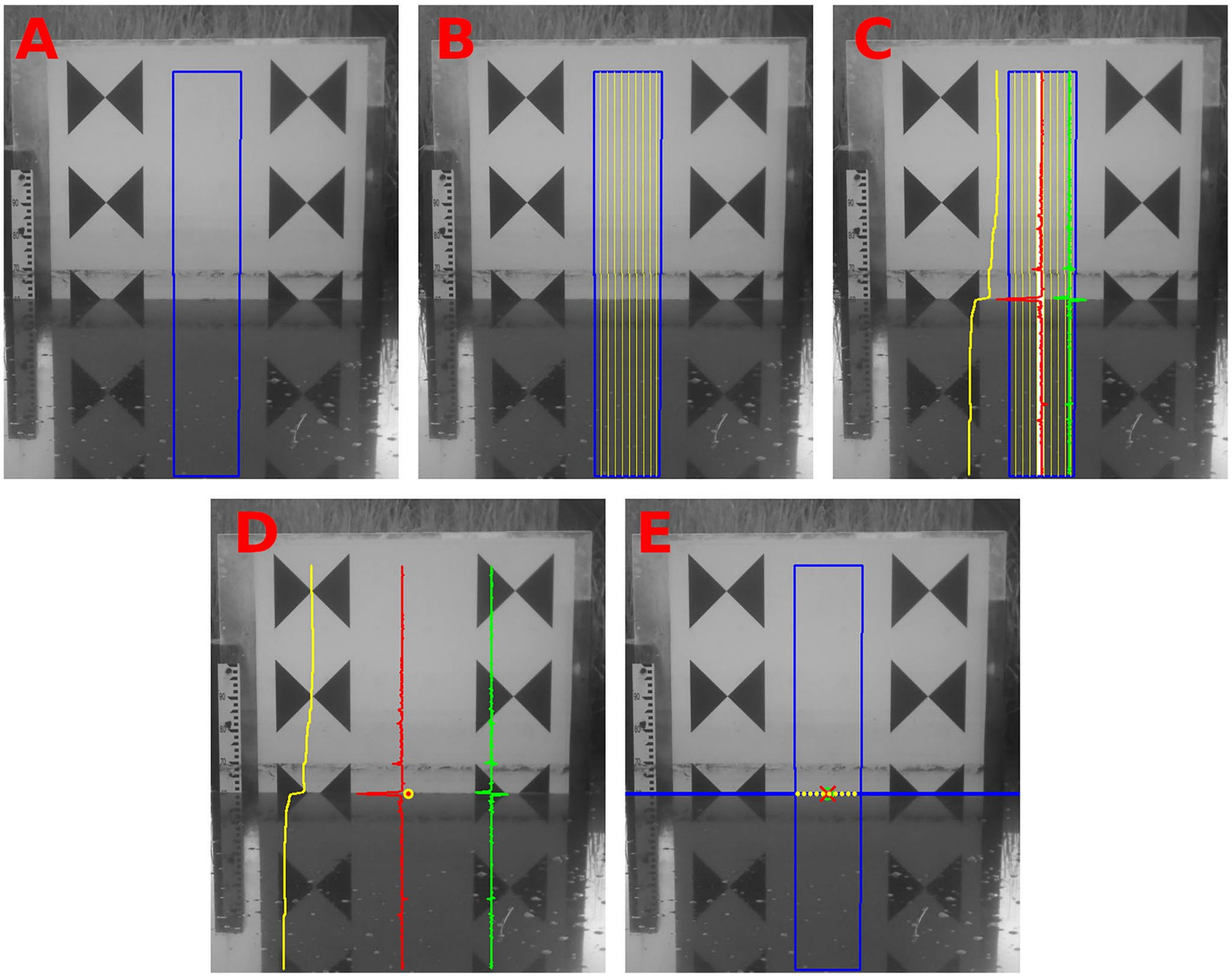


Figure 8. Water level search algorithm. Image A shows the water line search region of interest. Image B shows the vertical sub-regions within which individual point searches are performed. Image C shows the calculation results for an individual sub-region: The yellow line is the sum of the pixels in each row of the sub-region. The red line is the first derivative of the yellow line. The green line is the second derivative of the first line. The dot on the water line in Image D shows the location of the zero crossing of the second derivative for the sub-region. The yellow dots in image E are the found positions of all the sub-regions in the water line search region and the blue horizontal line is the result of a random sample consensus line fit of those points with the red X marking the water line at the center of the search region.

$$\left[\Delta^2 \vec{S}_n\right]_k = \left[\vec{S}_n\right]_k - 2\left[\vec{S}_n\right]_{k-1} + \left[\vec{S}_n\right]_{k-2}, \quad k = 3, \ldots, M. \tag{8}$$

Notice that the correction term to $y_{\text{edge},n}$ is positive since values of $\vec{S}_n$ are falling in the locale of the row $y_{\text{edge},n}$. The rightmost green line shown in Figure 8 Image D is the second derivative of the sum of pixel intensities for one of the narrow sub-regions.

A user-defined "minimum relevant drop" $\rho > 0$ is specified. If not enough points are found to satisfy the minimum relevant drop criteria, a RANSAC line fit is not possible, and the line detection fails due to an insufficient found point count. Figure 9 Image A shows an image where there were not enough water line edge points found to perform a line fit. The coordinates $(x_{\text{edge},n}, y_{\text{sub},n})$ for all $n \in \{1,\ldots,N_r\}$ satisfying the criterion

$$\max_{k\in\{2,\ldots M\}} \left[\vec{S}_n\right]_{k-1} - \left[\vec{S}_n\right]_k > \rho \tag{9}$$

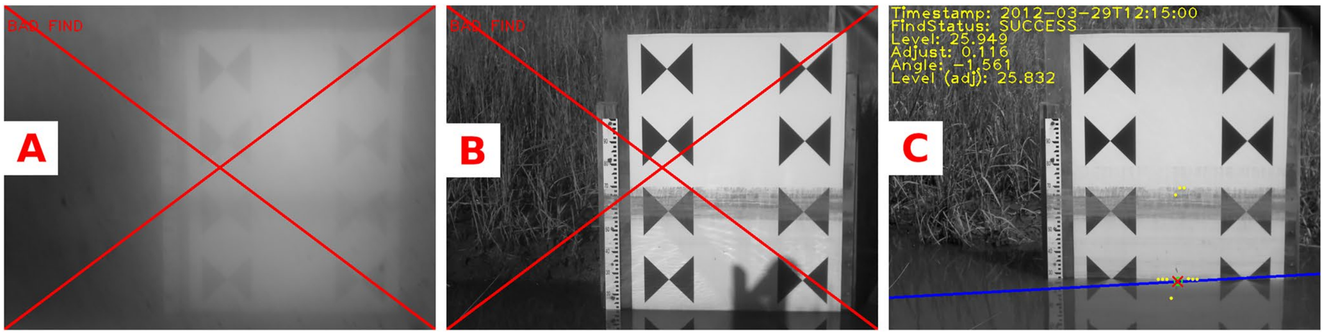


Figure 9. Overlay image that shows "bad" water line finds. Image A shows an image where not enough water edge points were found to calculate the line because of the fog. Image B shows an image where the found line points did not form a straight line due to the shadow in the image coupled with the false line at the high water mark. Image C shows an image where a water line was found with an angle that does not exactly follow the water line due to the false line points found at the high water mark and one point over the water. These kinds of lines can be identified by the calculated angle of the line shown on the overlay image and in the comma separated values file or the false points in the overlay image.

are collected as a set P. A random sample consensus (RANSAC) line fit is applied to the coordinates in P to give an estimate of the water line parameterized by slope $m_{\mathrm{wl}} \in \mathbb{R}$ and offset $b_{\mathrm{wl}} \in \mathbb{R}$ as:

$$y = m_{\mathrm{wl}} x + b_{\mathrm{wl}}. \tag{10}$$

The blue horizontal line in Figure 8 Image E is the result of fitting the candidate points shown as yellow dots from each of the sub-regions using the RANSAC line fit. The final estimate for the water level is defined as the y position of the calculated line, at the center of the water line search area, corresponding to $x_{\mathrm{wl}} = N/2$:

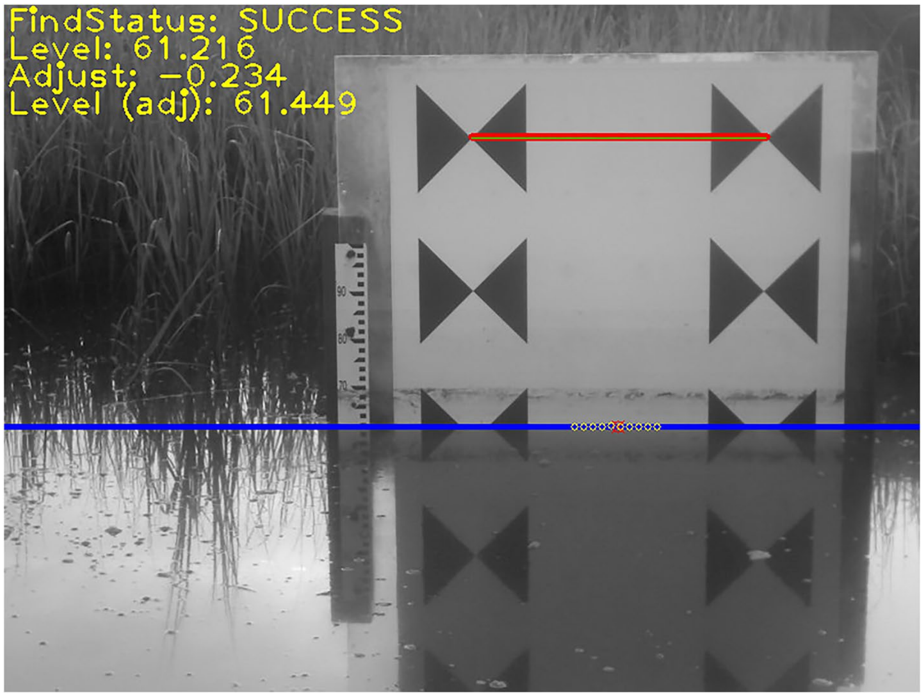
$$(x_{\mathrm{wl}}, y_{\mathrm{wl}}) = \left(\frac{N}{2}, m_{\mathrm{wl}}\frac{N}{2} + b_{\mathrm{wl}}\right). \tag{11}$$

The slope m_{wl} is expected to be near zero as long as the camera is oriented level to the actual water line. If the angle of the found line does not fall within a user defined range (i.e., the magnitude of m_{wl} is too large), or there are fewer than five candidate line points above the user defined magnitude threshold (i.e., P contains less than five coordinate pairs), the water level search calculation is flagged as unsuccessful. Figure 9 Image B shows an image where the water line edge points were found improperly so that the line find failed due to an invalid line angle. The resulting image space coordinate $(x_{\mathrm{wl}}, y_{\mathrm{wl}})$ is converted into world coordinates through the transform described in Section 2.1.2. The green circle on the yellow line with the red X overlay in Figure 8 Image E is the final water level estimate point.

Unsuccessful line detection can be a result of a variety of problems such as fog, shadows, glints, reflections, debris at the water line, wildlife between the camera and bow-tie target. Figure 9 shows some example images where line detection failed. Unsuccessful line detection can be transitory in a series of images in the case of things like wildlife occlusions, fog, and shadows. For more extreme events like damage or destruction to the bow-tie target or large camera movement, human intervention to restore the system could be required. It can be possible to recover information from images with unsuccessful line detection in some cases. When a transducer is coupled with a camera, data gaps can be filled when one or the other fails to make proper measurements. This could include visual inspection of a relatively small number of images where the line find fails.

2.3. Find Water Level in All Images in a Folder Hierarchy

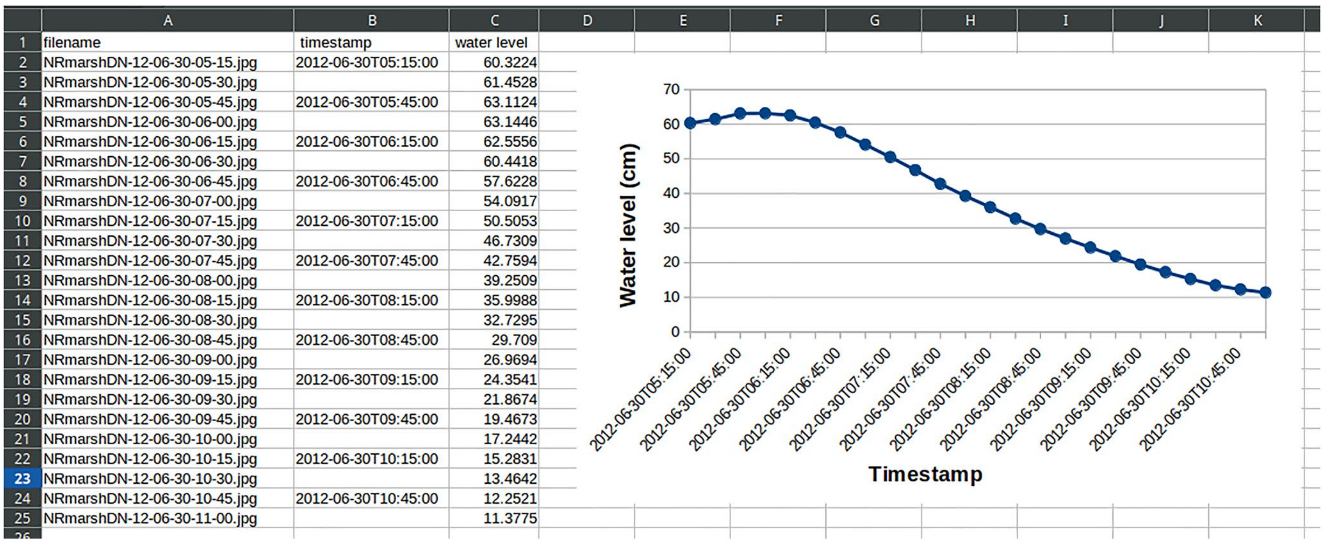
The program can also calculate the water level for each run-time image in a specified folder or for all images in an entire hierarchy of folders. Results are written to a CSV file and/or as annotated image files of the same format as is shown in Figure 10. The CSV file includes the file name of each image, capture timestamp, detected water level, angle of the water line, and any move adjustment that was made. Figure 11 is a graph created with data from



```
FindStatus: SUCCESS
Level: 61.216
Adjust: -0.234
Level (adj): 61.449
```

Figure 10. Overlay image that shows the results of a line find including the water level and movement calculation as text in yellow in the upper left corner of the image, a red line between the found positions of the move targets, a blue line where the water level line was found, yellow circles where points on the line were found, and a red circle and cross-hair of the point on the line reported as the water level.

the values calculated from each found water level in a folder of images and saved in a CSV file during a folder run. The CSV file also includes lines for the images where the water level search failed. An example of the CSV file format is shown in Appendix B.



	A	B	C
1	filename	timestamp	water level
2	NRmarshDN-12-06-30-05-15.jpg	2012-06-30T05:15:00	60.3224
3	NRmarshDN-12-06-30-05-30.jpg		61.4528
4	NRmarshDN-12-06-30-05-45.jpg	2012-06-30T05:45:00	63.1124
5	NRmarshDN-12-06-30-06-00.jpg		63.1446
6	NRmarshDN-12-06-30-06-15.jpg	2012-06-30T06:15:00	62.5556
7	NRmarshDN-12-06-30-06-30.jpg		60.4418
8	NRmarshDN-12-06-30-06-45.jpg	2012-06-30T06:45:00	57.6228
9	NRmarshDN-12-06-30-07-00.jpg		54.0917
10	NRmarshDN-12-06-30-07-15.jpg	2012-06-30T07:15:00	50.5053
11	NRmarshDN-12-06-30-07-30.jpg		46.7309
12	NRmarshDN-12-06-30-07-45.jpg	2012-06-30T07:45:00	42.7594
13	NRmarshDN-12-06-30-08-00.jpg		39.2509
14	NRmarshDN-12-06-30-08-15.jpg	2012-06-30T08:15:00	35.9988
15	NRmarshDN-12-06-30-08-30.jpg		32.7295
16	NRmarshDN-12-06-30-08-45.jpg	2012-06-30T08:45:00	29.709
17	NRmarshDN-12-06-30-09-00.jpg		26.9694
18	NRmarshDN-12-06-30-09-15.jpg	2012-06-30T09:15:00	24.3541
19	NRmarshDN-12-06-30-09-30.jpg		21.8674
20	NRmarshDN-12-06-30-09-45.jpg	2012-06-30T09:45:00	19.4673
21	NRmarshDN-12-06-30-10-00.jpg		17.2442
22	NRmarshDN-12-06-30-10-15.jpg	2012-06-30T10:15:00	15.2831
23	NRmarshDN-12-06-30-10-30.jpg		13.4642
24	NRmarshDN-12-06-30-10-45.jpg	2012-06-30T10:45:00	12.2521
25	NRmarshDN-12-06-30-11-00.jpg		11.3775

Figure 11. This graph was created with data from the values calculated from all the detected water levels in a folder of images and saved in a comma separated values file.

2.4. Create Animations

The program allows a user to select folders of images and create animations in Graphics Interchange Format (GIF) format to easily review results for anomalies and create visualizations for presentations. An example of a GIF created with GRIME2 can be viewed at GaugeCam.org Example gif (2021) or in the article DOI (Chapman & Gilmore, 2022).

2.5. Batch File Operation

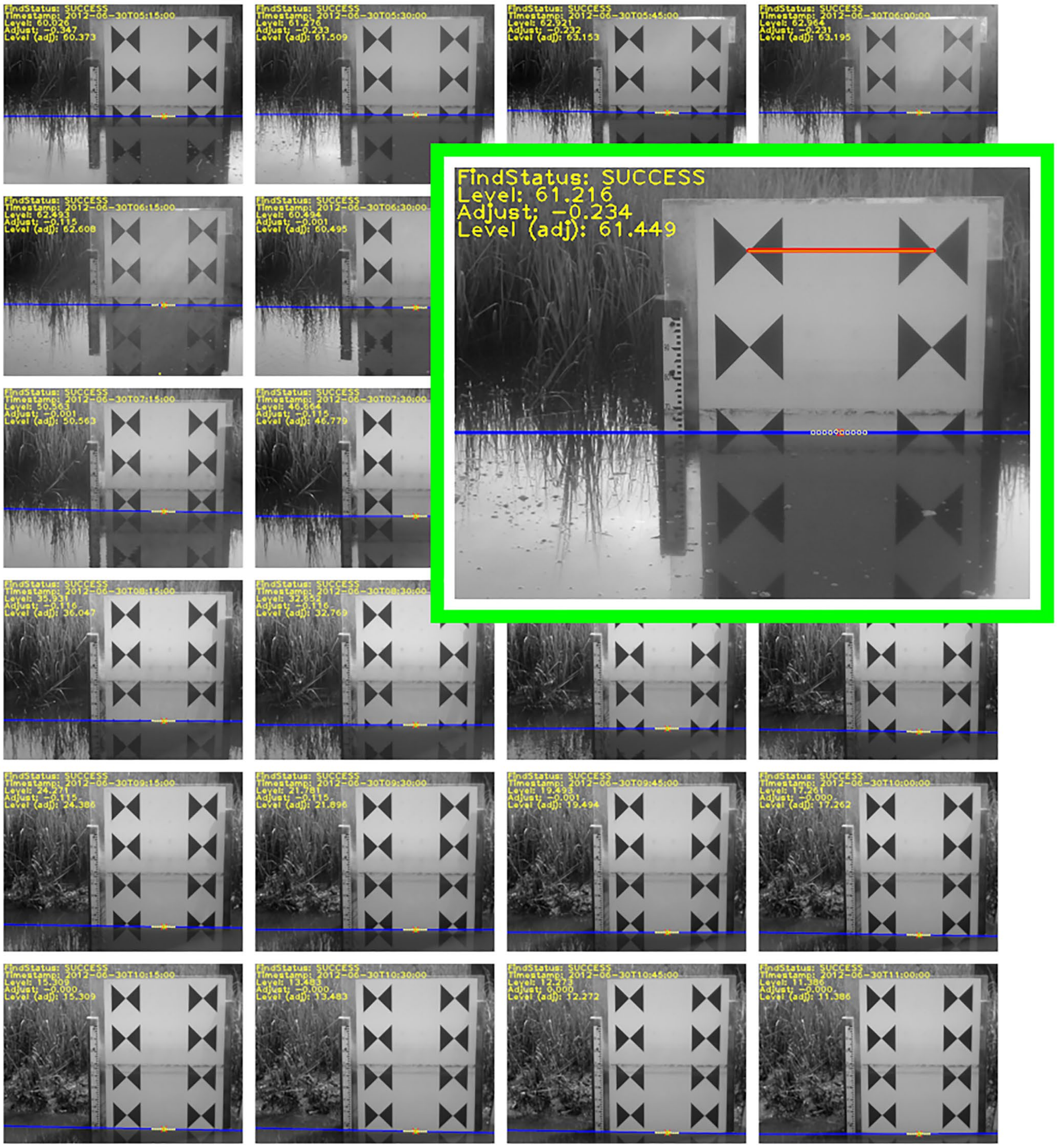
While it is possible to perform calculations and generate results for many images using the GUI, the command line interface version of the program provides a facility to run all the functionality of the GUI program including calibration, single image line finding, folder runs and creation of GIF animations. The command line version of the program is particularly useful for running multiple sets of data using a batch file. The results can be sent to individual CSV files or appended to an existing one. The following is an example of the contents of a simple batch file with commands and parameters to process two folders of images:

```
./grime2cli --run_folder --timestamp_from_filename ^
                    --timestamp_length 10 ^
                    --timestamp_pos 0 ^
                    --timestamp_format "yy-mm-ddTHH-MM" ^
                     "/home/kchapman/Documents/personal/unl/Projects/
GRIME2/gcgui/config/2012_demo/05" ^
                    --calib_json "/home/kchapman/Desktop/calib.json" ^
                    --csv_file "/home/kchapman/Desktop/result.csv" ^
                    --result_folder "/home/kchapman/Desktop/results"
./grime2cli --run_folder --timestamp_from_filename ^
                    --timestamp_length 10 ^
                    --timestamp_pos 0 ^
                    --timestamp_format "yy-mm-ddTHH-MM" ^
                     "/home/kchapman/Documents/personal/unl/Projects/
GRIME2/gcgui/config/2012_demo/06" ^
                    --calib_json "/home/kchapman/Desktop/calib.json" ^
                    --csv_file "/home/kchapman/Desktop/result.csv" ^
                    --result_folder "/home/kchapman/Desktop/results"
```

3. Results and Discussion

The results generated by a GRIME2 run include overlay images for each calculation and a CSV file that contains the results. The CSV file created for the run described in Section 2.5 is shown in Figure 12 and the result overlay images shown in Figure 11. The graph shown for the CSV file was created with a spreadsheet program. The CSV file generated from a longer run of measurements from February and March of 2012 taken in the North Carolina Tidal Marsh was used to generate the hydrograph shown in Figure 13. In this run of 2,975 images, the detection line algorithm failed 1.2% of the time (35 images) for a 98.8% success rate. At this success rate, visual inspection of images with undetected water line (e.g., 35 images in this case) is a tractable task.

Tests of these methods were reported (Gilmore et al., 2013) to measure water level to within ±3mm for over 80% of the time in a controlled laboratory setting. These results were from a single laboratory study and are expected to vary greatly the in the field depending on image resolution and quality, lighting, lens and optical system, target size and quality, field conditions, and a variety of other factors. The methods were subsequently used in the field (Etheridge et al., 2015). Third-party users have also adopted the program for use in their on-going research (University of Kansas, J. Wilhelm, pers. Comm., 13 October 2021) and operational needs (Idaho Power, C. Welcker, pers. comm., 3 May 2021).



```
FindStatus: SUCCESS
Timestamp: 2012-06-30T05:15:00
Level: 60.026
Adjust: -0.347
Level (adj): 60.373

FindStatus: SUCCESS
Timestamp: 2012-06-30T05:30:00
Level: 61.276
Adjust: -0.233
Level (adj): 61.509

FindStatus: SUCCESS
Timestamp: 2012-06-30T05:45:00
Level: 62.921
Adjust: -0.232
Level (adj): 63.153

FindStatus: SUCCESS
Timestamp: 2012-06-30T06:00:00
Level: 62.964
Adjust: -0.231
Level (adj): 63.195

FindStatus: SUCCESS
Timestamp: 2012-06-30T06:15:00
Level: 62.493
Adjust: -0.115
Level (adj): 62.608

FindStatus: SUCCESS
Timestamp: 2012-06-30T06:30:00
Level: 60.494
Adjust: -0.001
Level (adj): 60.495

FindStatus: SUCCESS
Timestamp: 2012-06-30T07:15:00
Level: 50.563
Adjust: -0.001
Level (adj): 50.563

FindStatus: SUCCESS
Timestamp: 2012-06-30T07:30:00
Level: 46.664
Adjust: -0.115
Level (adj): 46.779

FindStatus: SUCCESS
Level: 61.216
Adjust: -0.234
Level (adj): 61.449

FindStatus: SUCCESS
Timestamp: 2012-06-30T08:15:00
Level: 35.931
Adjust: -0.116
Level (adj): 36.047

FindStatus: SUCCESS
Timestamp: 2012-06-30T08:30:00
Level: 32.652
Adjust: -0.116
Level (adj): 32.769

FindStatus: SUCCESS
Timestamp: 2012-06-30T09:15:00
Level: 24.271
Adjust: -0.115
Level (adj): 24.386

FindStatus: SUCCESS
Timestamp: 2012-06-30T09:30:00
Level: 21.781
Adjust: -0.115
Level (adj): 21.896

FindStatus: SUCCESS
Timestamp: 2012-06-30T09:45:00
Level: 19.493
Adjust: -0.001
Level (adj): 19.494

FindStatus: SUCCESS
Timestamp: 2012-06-30T10:00:00
Level: 17.261
Adjust: -0.000
Level (adj): 17.262

FindStatus: SUCCESS
Timestamp: 2012-06-30T10:15:00
Level: 15.309
Adjust: -0.000
Level (adj): 15.309

FindStatus: SUCCESS
Timestamp: 2012-06-30T10:30:00
Level: 13.483
Adjust: -0.000
Level (adj): 13.483

FindStatus: SUCCESS
Timestamp: 2012-06-30T10:45:00
Level: 12.273
Adjust: 0.000
Level (adj): 12.272

FindStatus: SUCCESS
Timestamp: 2012-06-30T11:00:00
Level: 11.386
Adjust: -0.000
Level (adj): 11.386
```

Figure 12. These overlay images were created during the same water level inspection run that was performed to create the results from the comma separated values file shown in Figure 12.

4. Conclusions

This technical note introduces a Graphical User Interface (GUI) program and a Command Line Interface (CLI) program with a Microsoft Windows installer and a set of libraries to measure water level in images that feature a calibration target to measure water levels using images in small streams that have relatively tranquil waters.

10.1029/2022WR033203

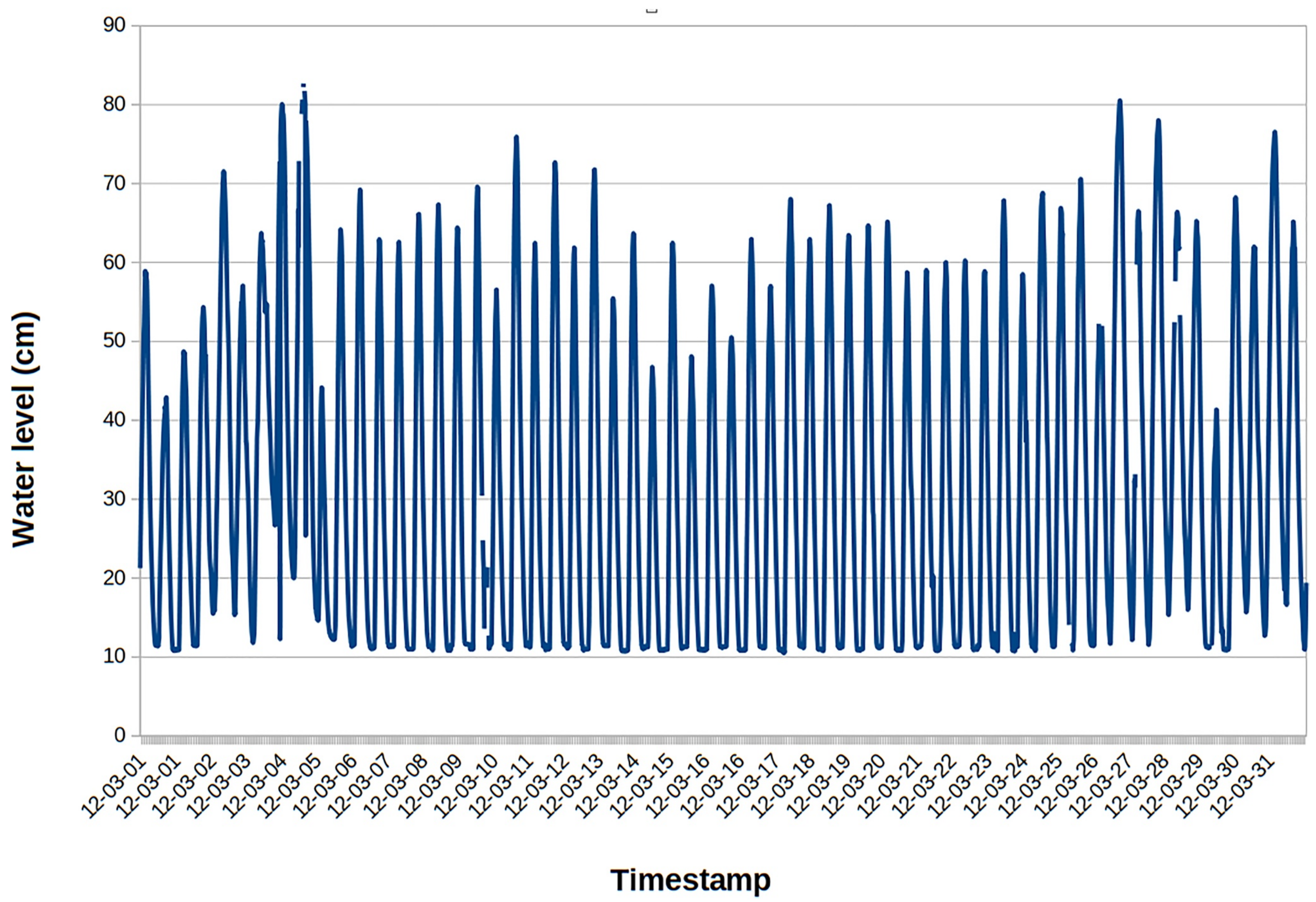


Figure 13. Tidal stage variations created from the comma separated values result file calculated from images of the North Carolina Tidal Marsh for March 2012.

The system is easy to use and provides sufficient documentation to make it suitable for deployment in real-world projects where commonly used sensors do not provide as much information as is required by researchers and operational users, or where data gaps may exist due to failure of more traditional water-level sensors. The source code is available for evaluation and modification with commercial friendly licenses and can be downloaded from GitHub (Chapman, 2021) or the article DOI (Chapman & Gilmore, 2022).

A limitation of the software is that it depends on the presence of a bow-tie target fixed in the water to allow it to perform its measurements. Future research could include the addition of algorithms and processes that provide the same functionality but require a less obtrusive calibration target or no calibration target at all.

Appendix A: Calibration Error

Calibration error due to application of the pixel to world coordinate transform for a typical case was calculated by finding the bow-tie center image coordinates in the seven images taken in the North Carolina tidal marsh shown in Figure A1 under varying lighting conditions with some change in position of the calibration target and the camera. These pixels coordinates are then converted to world coordinates using the homography created when the system was calibrated to calculate the perspective transform for each center point. The world coordinate points are then converted back to pixel coordinate points with a reverse transform. The RMSE for the variation in *X*, *Y*, and Euclidean distance between the original image coordinates and those calculated after the perspective and reverse transforms are shown in Table A1. The full-size images and the raw data from which these calculations were made can be found in the article DOI (Chapman & Gilmore, 2022).

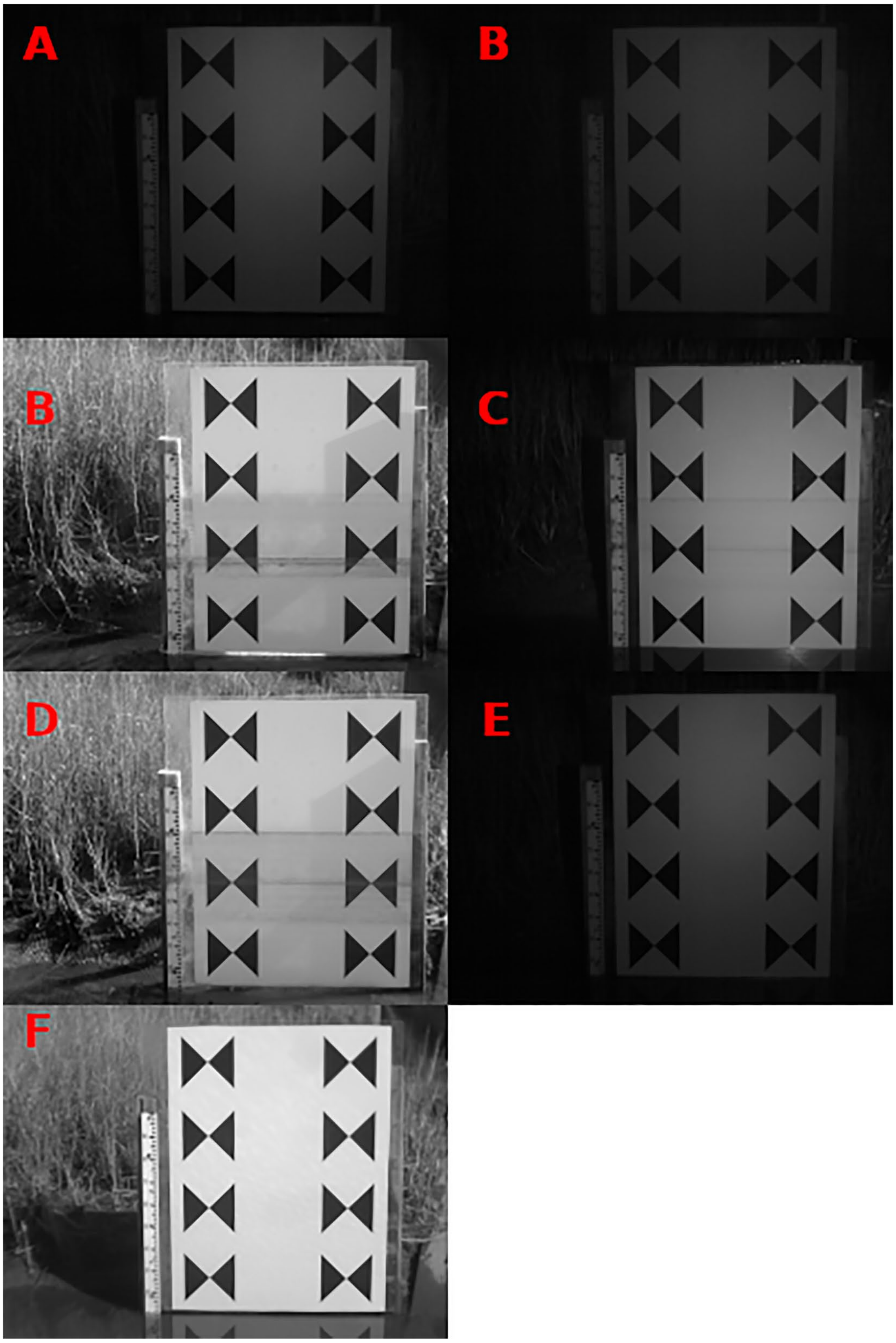


Figure A1. Images used to show the Root Mean Square Error (RMSE) of the calibration target bow-tie center pixel positions after they are transformed from pixel to world coordinates then back to image coordinates. The results are shown in Table A1. The images in this figure are associated with the following filenames in the table: A is NrmarshDN-12-02-01-23-30.png, B is NrmarshDN-12-02-03-00-30.png, C is NrmarshDN-12-02-12-09-01.png, D is NrmarshDN-12-02-23-05-00.png, E is NrmarshDN-12-02-28-09-30.png, F is NrmarshDN-12-02-01-19-30.png, and G is NrmarshDN-12-02-02-12-30.

Table A1
Root Mean Square Error (RMSE) for Pixel Positions After Pixel to World Transform Followed by Reverse World to Pixel Transform of a Bow-Tie Target in the North Carolina Tidal Marsh in Seven Different Poses

Filename	*X* RMSE	*Y* RMSE	Euclidean distance RMSE
NRmarshDN-12-02-01-19-30.jpg	$1.208e^{-11}$	$1.258e^{-11}$	$1.744e^{-11}$
NRmarshDN-12-02-01-23-30.jpg	$1.213e^{-11}$	$1.351e^{-11}$	$1.815e^{-11}$
NRmarshDN-12-02-02-12-30.jpg	$1.104e^{-11}$	$2.501e^{-11}$	$2.734e^{-11}$
NRmarshDN-12-02-03-00-30.jpg	$8.886e^{-12}$	$1.285e^{-11}$	$1.563e^{-11}$
NRmarshDN-12-02-12-09-01.jpg	$1.302e^{-11}$	$1.316e^{-11}$	$1.851e^{-11}$
NRmarshDN-12-02-23-05-00.jpg	$1.755e^{-11}$	$2.915e^{-11}$	$3.402e^{-11}$
NRmarshDN-12-02-28-09-30.jpg	$1.075e^{-11}$	$2.480e^{-11}$	$2.702e^{-11}$

Note. The raw data for these calculations can be found in the article digital object identifier (Chapman & Gilmore, 2022). The values should be more meaningful when 3d calibrations are implemented

Appendix B: CSV File Field Descriptions and Example File

When comma separated values (CSVs) result output is enabled for a run of all the images within a folder or folder structure from the grime2cli program, a large number of fields are populated with one row for each image. Table B1 is an example of the abbreviated set of columns shown in the GUI when CSV file creation is enabled for a run of the images in the folder from the GUI program. Tables B2, B3, B4, B5, and B6 show the field names and descriptions for the full CSV file. Failure status descriptions and additional result information are for future versions of the program.

Table B1
Example CSV Results Shown in Graphical User Interface

Filename, timestamp, status, water level, line angle, level adjustment
NRmarshDN-12-02-01-14-45.jpg,2012-02-01T14:45:00,SUCCESS,21.844,0.483,0.936
NRmarshDN-12-02-01-15-00.jpg,2012-02-01T15:00:00,SUCCESS,24.562,−0.162,0.936
NRmarshDN-12-02-01-15-15.jpg,2012-02-01T15:15:00,SUCCESS,26.712,0.462,0.936
NRmarshDN-12-02-01-15-30.jpg,2012-02-01T15:30:00,SUCCESS,28.500,0.456,0.936
NRmarshDN-12-02-01-15-45.jpg,2012-02-01T15:45:00,SUCCESS,30.285,0.443,0.936
NRmarshDN-12-02-01-16-00.jpg,2012-02-01T16:00:00,SUCCESS,31.813,0.438,0.936
NRmarshDN-12-02-01-16-15.jpg,2012-02-01T16:15:00,SUCCESS,33.334,−0.938,0.936
NRmarshDN-12-02-01-16-30.jpg,2012-02-01T16:30:00,FAIL,−9999999.000,−9999999.000,−9999999.000
NRmarshDN-12-02-01-16-45.jpg,2012-02-01T16:45:00,SUCCESS,31.220,−0.708,0.702
NRmarshDN-12-02-01-17-00.jpg,2012-02-01T17:00:00,SUCCESS,29.746,−0.261,0.702
NRmarshDN-12-02-01-17-15.jpg,2012-02-01T17:15:00,SUCCESS,28.163,−0.364,0.702

Table B2
Image Path, Find Status, Time Status, and Calculated Water Level Points

Field name	Description
imgPath	Full image path
findSuccess	Water level find status
Timestamp	ISO timestamp
waterLevel	Calculated water level
waterLevelAdjusted	Calculated water level with move adjustment
calcLinePts-angle	Angle of the calculated water line

Table B2
Continued

Field name	Description
calcLinePts-lftPixel-x	Leftmost pixel x pos of the found water line
calcLinePts-lftPixel-y	Leftmost pixel y pos of the found water line
calcLinePts-ctrPixel-x	Center pixel x pos of the found water line
calcLinePts-ctrPixel-y	Center pixel y pos of the found water line
calcLinePts-rgtPixel-x	Rightmost pixel x pos of the found water line
calcLinePts-rgtPixel-y	Rightmost pixel y pos of the found water line
calcLinePts-lftWorld-x	Leftmost world x pos of the found water line
calcLinePts-lftWorld-y	Leftmost world y pos of the found water line
calcLinePts-ctrWorld-x	Center world x pos of the found water line
calcLinePts-ctrWorld-y	Center world y pos of the found water line
calcLinePts-rgtWorld-x	Rightmost world x pos of the found water line
calcLinePts-rgtWorld-y	Rightmost world y pos of the found water line

Table B3
Reference Move Line Points and Angle

Field name	Description
refMovePts-angle	Angle of the reference line between the top bow ties
refMovePts-lftPixel-x	Left pixel x pos of the ref line between the top bow ties
refMovePts-lftPixel-y	Left pixel y pos of the ref line between the top bow ties
refMovePts-ctrPixel-x	Center pixel x pos of the ref line between the top bow ties
refMovePts-ctrPixel-y	Center pixel y pos of the ref line between the top bow ties
refMovePts-rgtPixel-x	Right pixel x pos of the ref line between the top bow ties
refMovePts-rgtPixel-y	Right pixel y pos of the ref line between the top bow ties
refMovePts-lftWorld-x	Left world x pos of the ref line between the top bow ties
refMovePts-lftWorld-y	Left world y pos of the ref line between the top bow ties
refMovePts-ctrWorld-x	Center world x pos of the ref line between the top bow ties
refMovePts-ctrWorld-y	Center world y pos of the ref line between the top bow ties
refMovePts-rgtWorld-x	Right world x pos of the ref line between the top bow ties
refMovePts-rgtWorld-y	Right world y pos of the ref line between the top bow ties

Table B4
Found Move Line Points and Angle

Field name	Description
foundMovePts-angle	Angle of the move check line between the top bow ties
foundMovePts-lftPixel-x	Left pixel x pos of the move check line between the top bow ties
foundMovePts-lftPixel-y	Left pixel y pos of the move check line between the top bow ties
foundMovePts-ctrPixel-x	Center pixel x pos of the move check line between the top bow ties
foundMovePts-ctrPixel-y	Center pixel y pos of the move check line between the top bow ties
foundMovePts-rgtPixel-x	Right pixel x pos of the move check line between the top bow ties
foundMovePts-rgtPixel-y	Right pixel y pos of the move check line between the top bow ties
foundMovePts-lftWorld-x	Left world x pos of the move check line between the top bow ties
foundMovePts-lftWorld-y	Left world y pos of the move check line between the top bow ties

Table B4
Continued

Field name	Description
foundMovePts-ctrWorld-x	Center world x pos of the move check line between the top bow ties
foundMovePts-ctrWorld-y	Center world y pos of the move check line between the top bow ties
foundMovePts-rgtWorld-x	Right world x pos of the move check line between the top bow ties
foundMovePts-rgtWorld-y	Right world y pos of the move check line between the top bow ties

Table B5
Move Offset Points and Angle

Field name	Description
offsetMovePts-angle	Angle offset between the reference and move check lines
offsetMovePts-lftPixel-x	Left pixel x offset between the reference and move check lines
offsetMovePts-lftPixel-y	Left pixel y offset between the reference and move check lines
offsetMovePts-ctrPixel-x	Center pixel x offset between the reference and move check lines
offsetMovePts-ctrPixel-y	Center pixel y offset between the reference and move check lines
offsetMovePts-rgtPixel-x	Right pixel x offset between the reference and move check lines
offsetMovePts-rgtPixel-y	Right pixel y offset between the reference and move check lines
offsetMovePts-lftWorld-x	Left world x offset between the reference and move check lines
offsetMovePts-lftWorld-y	Left world y offset between the reference and move check lines
offsetMovePts-ctrWorld-x	Center world x offset between the reference and move check lines
offsetMovePts-ctrWorld-y	Center world y offset between the reference and move check lines
offsetMovePts-rgtWorld-x	Right world x offset between the reference and move check lines
offsetMovePts-rgtWorld-y	Right world y offset between the reference and move check lines

Table B6
Water Line, Found Vertical Swath Points Used in RANSAC Line Calculation

Field name	Description
foundPts[0]-x	Find water line x position of the first vertical swath #0
foundPts[0]-y	Find water line y position of the first vertical swath #0
foundPts[1]-x	Find water line x position of the first vertical swath #1
foundPts[1]-y	Find water line y position of the first vertical swath #1
foundPts[2]-x	Find water line x position of the first vertical swath #2
foundPts[2]-y	Find water line y position of the first vertical swath #2
foundPts[3]-x	Find water line x position of the first vertical swath #3
foundPts[3]-y	Find water line y position of the first vertical swath #3
foundPts[4]-x	Find water line x position of the first vertical swath #4
foundPts[4]-y	Find water line y position of the first vertical swath #4
foundPts[5]-x	Find water line x position of the first vertical swath #5
foundPts[5]-y	Find water line y position of the first vertical swath #5
foundPts[6]-x	Find water line x position of the first vertical swath #6
foundPts[6]-y	Find water line y position of the first vertical swath #6
foundPts[7]-x	Find water line x position of the first vertical swath #7
foundPts[7]-y	Find water line y position of the first vertical swath #7

Table B6
Continued

Field name	Description
foundPts[8]-x	Find water line x position of the first vertical swath #8
foundPts[8]-y	Find water line y position of the first vertical swath #8
foundPts[9]-x	Find water line x position of the first vertical swath #9
foundPts[9]-y	Find water line y position of the first vertical swath #9

Disclaimer

Christian D. Chapman is currently an MIT Lincoln Laboratory employee. No laboratory funding or resources were used to produce the result/findings reported in this publication.

Conflict of Interest

The authors declare no conflicts of interest relevant to this study.

Data Availability Statement

The Windows installer and source code can be downloaded from Github at https://github.com/gaugecam-dev/GRIME2 (Chapman, 2021). Tutorials and other explanatory links are available at https://gaugecam.org/ (Gilmore, 2021). All these materials can also be downloaded from the article digital object identifier at http://doi.org/10.32873/unl.dr.20220301 (Chapman & Gilmore, 2022).

Acknowledgments
This work was supported by the U.S. Department of Agriculture—National Institute of Food and Agriculture NEB-21-177 (Hatch Project 1015698). Additional support was provided by the University of Nebraska Research Council through a Grant-in-Aid grant funded through a gift from the John C. and Nettie V. David Memorial Trust Fund. Suggestions from Frank Engel and two anonymous reviewers greatly improved the manuscript.

References

Ballard, D. H., & Brown, C. M. (1982). 4.3.5 generalizing the Hough transform. In *Computer vision* (pp. 128–131).
Birgand, F., Chapman, K., Hazra, A., Gilmore, T., Etheridge, R., & Staicu, A. (2022). Field performance of the gaugecam image-based water level measurement system. *PLOS Water. Accepted*.
Boiten, W. (2008). *Hydrometry*. CRC Press.
Boost (2021). The boost c++ libraries. Retrieved from http://www.boost.org/
Chakravarthy, S., Sharma, R., & Kasturi, R. (2002). Noncontact level sensing technique using computer vision. *IEEE Transactions on Instrumentation and Measurement*, *51*(2), 353–361. https://doi.org/10.1109/19.997837
Chapman, K. W. (2021). Grime2. Retrieved from https://github.com/gaugecam/GRIME2
Chapman, K. W., & Gilmore, T. E. (2022). Technical note: Open-source software for water-level measurement in images with a calibration target [source code, binaries, and data set] in University of Nebraska-Lincoln data repository. University of Nebraska consortium of libraries - uncl. https://doi.org/10.32873/unl.dr.20220301
Cognex (2022). Cognex patmax object location. Retrieved from https://www.cognex.com/products/machine-vision/vision-software/vision-tools/object-location/patmax-object-location
Etheridge, J. R., Birgand, F., & Burchell, M. R., II. (2015). Quantifying nutrient and suspended solids fluxes in a constructed tidal marsh following rainfall: The value of capturing the rapid changes in flow and concentrations. *Ecological Engineering*, *78*, 41–52. https://doi.org/10.1016/j.ecoleng.2014.05.021
ffmpeg (2021). ffmpeg. Retrieved from https://ffmpeg.org/
GaugeCam.org example gif. (2021). Retrieved from http://gaugecam.org/wp-content/uploads/2021/08/grimey105fps025size.gif
Gilmore, T. E. (2021). Gaugecam.org. Retrieved from https://gaugecam.org
Gilmore, T. E., Birgand, F., & Chapman, K. W. (2013). Source and magnitude of error in an inexpensive image-based water level measurement system. *Journal of Hydrology*, *496*(2013), 178–186. https://digitalcommons.unl.edu/natrespapers/762/
Hamel, P., Riveros-Iregui, D., Ballari, D., Browning, T., Célleri, R., Chandler, D., et al. (2018). Watershed services in the humid tropics: Opportunities from recent advances in ecohydrology. *Ecohydrology*, *11*(3). https://doi.org/10.1002/eco.1921
Harmel, R. D., Cooper, R. J., Slade, R. M., Haney, R. L., & Arnold, J. G. (2006). Cumulative uncertainty in measured streamflow and water quality data for small watersheds. *Transactions of the ASABE*, *49*(3), 689–701. https://doi.org/10.13031/2013.20488
Harvey, P. (2021). Exiftool. Retrieved from https://exiftool.org/
Kaplan, N. H., Sohrt, E., Blume, T., & Weiler, M. (2019). Monitoring ephemeral, intermittent and perennial streamflow: A dataset from 182 sites in the attert catchment. *Earth System Science Data*, *11*, 1363–1374. https://doi.org/10.5194/essd-11-1363-2019
Lin, Y.-T., Lin, Y.-C., & Han, J.-Y. (2018). Automatic water-level detection using single-camera images with varied poses. *Measurement*, *127*, 167–174. https://doi.org/10.1016/j.measurement.2018.05.100
Matrox (2022). Matrox mil pattern recognition. Retrieved from https://www.matrox.com/en/imaging/products/software/sdk/mil/tools/pattern-recognition
Noto, S., Tauro, F., Petroselli, A., Apollonio, C., Botter, G., & Grimaldi, S. (2022). Low-cost stage-camera system for continuous water-level monitoring in ephemeral streams. *Hydrological Sciences Journal*, *0* (ja). https://doi.org/10.1080/02626667.2022.2079415

OpenCV (2019). Open source computer vision library. Retrieved from https://opencv.org

Qt (2021). Qt. Retrieved from https://qt.io/

Schoener, G. (2018). Time-lapse photography: Low-cost, low-tech alternative for monitoring flow depth. *Journal of Hydrologic Engineering*, *23*(2). https://doi.org/10.1061/(ASCE)HE.1943-5584.0001616

Takagi, Y., Tsujikawa, A., Takato, M., Saito, T., & Kaida, M. (1998). Development of a noncontact liquid level measuring system using image processing. *Water Science and Technology*, *37*(12), 381–387. https://doi.org/10.2166/wst.1998.0564

Tauro, F., Selker, J., van de Giesen, N., Abrate, T., Uijlenhoet, R., Porfiri, M., et al. (2018). Measurements and observations in the xxi century (moxxi): Innovation and multi-disciplinarity to sense the hydrological cycle. *Hydrological Sciences Journal*, *63* (2), 169–196. https://doi.org/10.1080/02626667.2017.1420191

Tian, Q., & Huhns, M. N. (1986). Algorithms for subpixel registration. *Computer Vision, Graphics, and Image Processing*, 200-233.

Turnipseed, D. P., & Sauer, V. B. (2010). *Stage measurements at gaging stations: U.S. geological survey techniques and methods book 3*. U.S. Geological Survey. Retrieved from http://pubs.usgs.gov/tm/tm3-a7/