Planar Laser Induced Florescence

Setup, Imaging, and Analysis

Samuel Harry

August 30, 2021

Contents

Chapter 1

Digital image processing

1.1 Introduction

The goal of the digital image processing described below is to identify the air-water surface emphasized by the planar laser induces fluorescence (PLIF) technique. This image processing assumes that there is a dark region at the top of the image, and that there is a connected bright region across the image at some elevation, or vice-versa. While this code has primarily been tested on PLIF images using a laser sheet, it could be applied to many types of fluid flow, e.g. water illuminated by a laser or turbulent aerated flow viewed through a transparent window. The algorithm described below applies to surfaces which are single valued for any given pixel column. This is a consequence of the simple surface pixel selection processing.

1.2 Algorithm outline

The air-water surface will be identified by find the uppermost valid local maximum of the image gradient. Here, the challenge is identifying *valid* candidates. Practical challenges can lead to unwanted imaging artifacts such as reflections, or undesired features such as splash up droplets. Identifying local maxima allows for interpolating the location of the maxima with simple functions, in this case a quadratic function is chosen. Interpolation of the maxima reduces the stair-step artifact created by discrete pixel locations.

An algorithm to analyze each image is outlined as follows:

- Preliminary processing (see §1.3)
  - Convert to grayscale if needed (see §1.3.1)
  - Normalize the image (see §1.3.2)
  - Median filter the image (see §1.3.3)
  - Denoise the image using non-local means de-noising (see §1.3.4)
  - Undistort the image if needed (see §1.3.5)
- Calculate the magnitude of image gradient with Scharr convolution filter (see §1.4)

- Select valid maxima of the gradient (see §1.5)
  - Identify the largest gradient in each column (see §1.5.1)
  - Identify sufficiently large gradients (see §1.5.2)
  - Identify water surface by phase (see §1.5.3)
  - Identify water surface by shape (see §1.5.4)
  - Identify all local maxima (see §1.5.5)
- Identify the uppermost maxima for each column with weighting function (see §1.6)
- Output integer values as surface line
- For each surface line location, super-sample the maxima with adjacent intensity values (details in §1.7)
- Output super-sample adjustment

Note that, with the exception of the super-sampling method, the algorithm is composed of existing image processing algorithms. These algorithms were chosen because they are available in commonly available image processing toolboxes, such as OpenCV and its implementations in Python, Matlab, Java, etc. This means that this algorithm could be easily implemented in the researcher's preferred programming language.

This algorithm is designed to work exclusively in image space with image concepts. Converting the identified surface line to a physical quantity is discussed in chapter 2. The details of each step is discussed in the following sections.

1.3 Preliminary processing

1.3.1 Grayscale images

In the case of PLIF, color images may be converted to grayscale with a simple weighted average, as the signal is primarily independent of color. Choosing a grayscale camera that has a higher sensitivity to light (ISO) may be preferred to a color camera. In some cases color can be useful for segmenting an image, although they are not discussed here.

1.3.2 Histogram normalization

Histogram normalization is used to ensure that bright spots are not too bright, and that dark spots are not too dark. This helps ensure a uniform intensity gradient across the water surface. In particular, it helps reduce the impact of 'hot-spots' in the laser sheet created by a cylindrical lens or powell lens. Generally, it makes analysis of the images more robust. Note that histogram equalization techniques tend to introduce more problems than they solve when applied to PLIF images, so they are not used or discussed in detail.

1.3.3 Median filter

For PLIF images of a water surface, a common artifact is the appearance of small particles in the water surface image. With the exception of turning the laboratory into a clean room, dust is unavoidable. Such artifacts may be readily mitigated with a median filter. A median filter is advantageous here because it is edge preserving, e.g. consider a step function with white noise. In contrast to a low pass filter like a mean filter, the median filter does not move or spread the step. The edge preserving feature is important because we wish to identify the edge between the air and water in the image. In addition, a median filter excels at removing isolated peaks, e.g. a delta function with white noise. The dust particles are both small and highly illuminated in the image, and appear as sudden jump in the water surface. Accordingly, they may be efficiently removed with a median filter. Consider the example shown in figure 1.1.

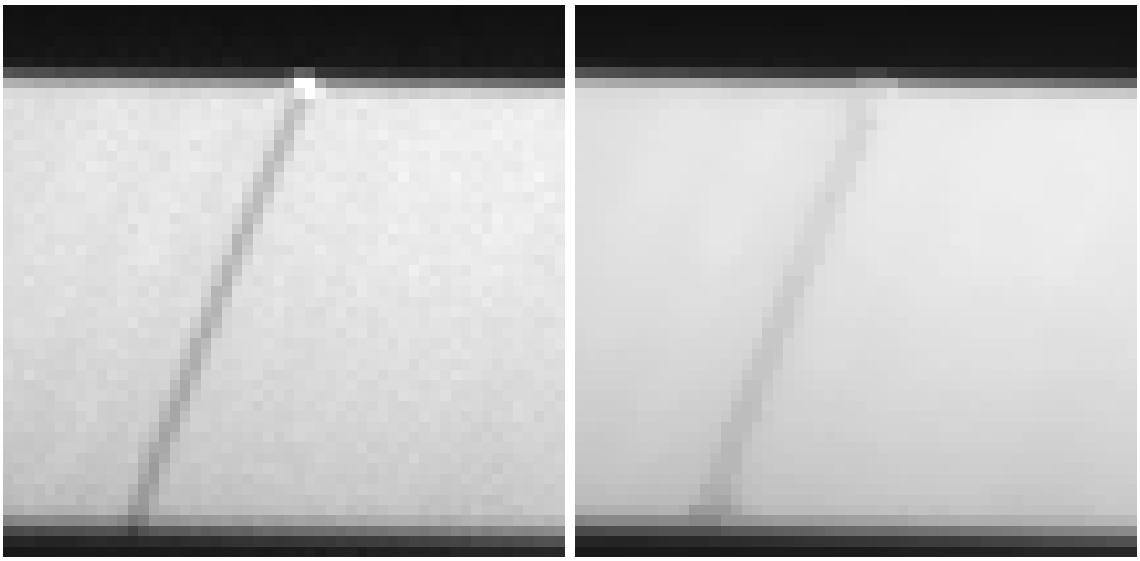

Figure 1.1: Small particle removal from the water surface of an example PLIF image. Original image shown left, median filtered image shown right.

1.3.4 Image de-noising

Image de-noising is important for taking a meaningful image gradient. It is known that gradient operators emphasize white noise, in this case the white noise introduced by the image sensor. Such noise may produce an artificial gradient when there is not one in the physical signal. To combat this, a popular method is low pass filtering, because these filters are well established in the signal processing community. Low pass filtering an image, therefore smoothing sharp edges, before high pass filtering an image in order to emphasize edges is contradictory to the goal presented here. In addition, there is no reason to assume that the intensity values in an image are dependent on the values near it.

Such observations are the motivation for implementing the non-local means denoising method. OpenCV implements a version developed by Buedes et al. (2011). This way images can be denoised while minimizing the smoothing of sharp edges. If the non-local

means denoising method is for some reason not available, consider the denoising properties of a median filter, as shown in figure 1.1.

1.3.5 Removing lens distortion

Lens distortion is removed with the model used in OpenCV ($> 4.1.0$). The model is as follows:

$$p'' = \frac{p''' - c_p}{f_p}$$
$$q'' = \frac{q''' - c_q}{f_q}$$
$$r^2 = (p'')^2 + (q'')^2$$
$$p' = p''\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + 2p_1 p'' q'' + p_2(r^2 + 2p''^2)$$
$$q' = q''\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + p_1(r^2 + 2q''^2) + 2p_2 p'' q''$$
$$p = f_p p' + c_p$$
$$q = f_q q' + c_q$$

where $I(p''', q''')$ describes the original image, p''' and q''' are the integer pixel coordinates of the original image, k_1, k_2, k_3, p_1, p_2 are lens distortion parameters determined in advance, and f_p, f_q, c_p, c_q are lens intrinsic parameters determined in advance. At this stage p and q are not integer values, and so the image must be re sampled onto an integer grid as a standard image. Typically, this is done by interpolating cubic splines to the image, however the exact method depends on the implementation used. The undistorted image is now ready for analysis.

1.3.6 Order of pre-processing steps

The order in which the pre-processing steps is not critical. A comparison between three different orders of application of the sub-steps shows qualitatively similar results. The image noise is typically within ± 0.1 pixels, and the spike from the small particle is reduced to ~ 0.3 pixels, regardless of the order of application.

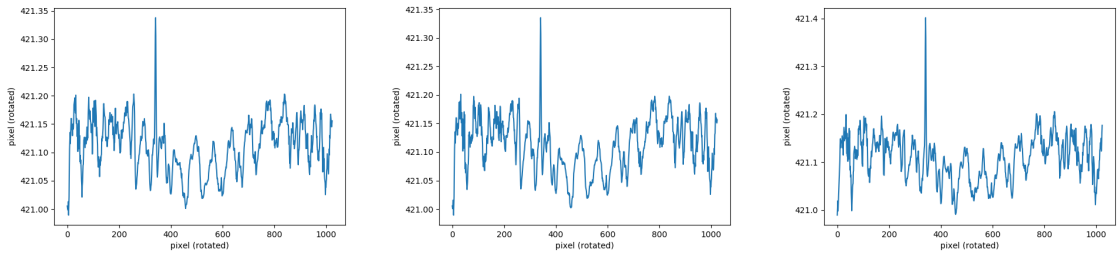

Figure 1.2: Comparison of order of preprocessing techiques. Left) median filter, normalization, then denoising. Center) normalization, median filter, then denoising. Right) denoising, median filter, then normalization. Data shown is from processing the same images shown in figure 1.1.

1.4 Gradient operators

Common convolution filters for estimating the image gradient are the central differences, Sobel, and Scharr filters, which are:

$$G_{x,central} = \begin{bmatrix} 1, 0, -1 \end{bmatrix} \; ; \; G_{y,central} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \; ;$$

$$G_{x,sobel} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \; ; \; G_{y,sobel} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \; ;$$

$$G_{x,scharr} = \begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix} \; ; \; G_{y,scharr} = \begin{bmatrix} 3 & 10 & 3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix} \; ;$$

where each kernel must be normalized such that the sum of its parts is one. Each kernel designed to be computation efficient and valid estimates of the gradient in their own way. I have struggled to find comprehensive work on the details of each, however, Scharr's filter is the result of his PhD thesis (evidently, but I can't read German). Both the Sobel and Scharr filter are more robust to noise than the central differences filter. Scharr's filter is chosen without a proper reason over the sobel filter. The gradient in both the x and y direction are estimated, then the magnitude and phase are taken in the usual way.

1.5 Finding valid maxima

In order to find the first *valid* local maximum of the gradient, gradients which are suitably small to be considered background noise must be ignored. Selecting valid local maxima can be achieved in a variety of ways.

1.5.1 Column maxima

The simplest and fastest method to find the water surface is to identify the largest gradient value in each pixel column. This method only works with careful physical preparation and may be invalided by unexpected imaging artifacts. The results are guaranteed to be a maxima or plateau edge, so no further refinement is needed. Use this if it works.

1.5.2 Segmentation

Standard segmentation methods can be used to find the water surface. In figure 1.3, an image gradient is segmented using both Otsu thresholding and kmeans-clustering. It can bee seen that kmeans-clustering with two clusters is analogous to Otsu thresholding, in this case. In contrast, utilizing multiple clusters allows for fine control of the segmentation. Utilizing too few clusters may not properly identify the connectivity of the surface, as shown in case of two cluster. Utilizing too many clusters may overly connect the surface, as shown in the case of four clusters. In this case, three clusters provides a sweet spot where the entire surface is

identified as a single connected group in addition to being isolated from the lower surface. Additionally, the number of clusters is a adjustable parameter and can be chosen based on the data being analyzed.

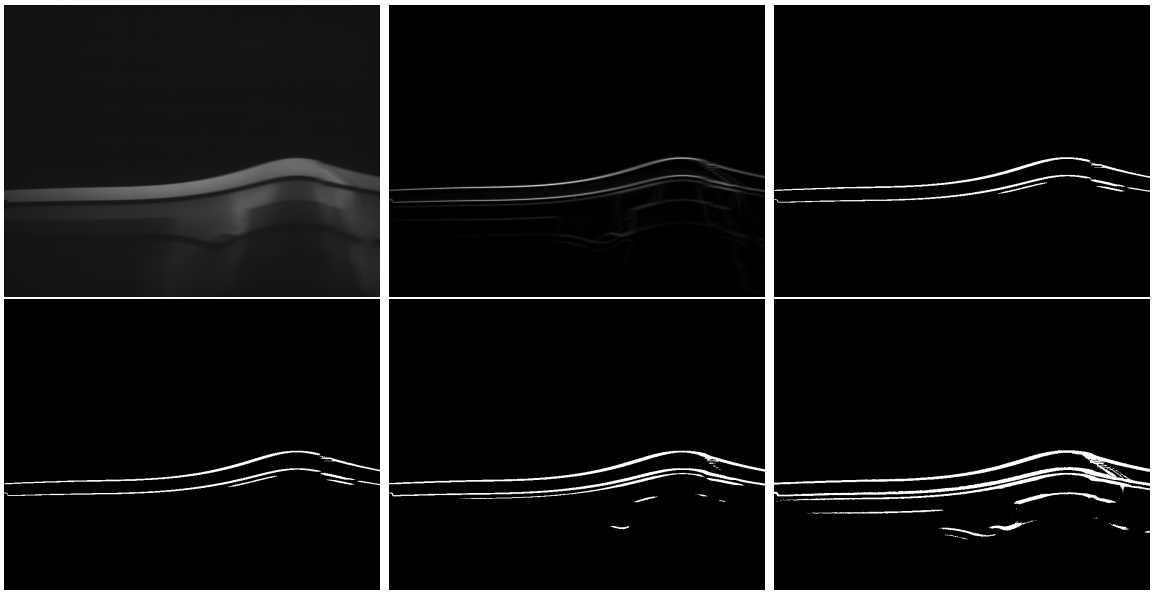

Figure 1.3: Segmentation of the gradient using a variety of methods. In order from top left to bottom right: First, the original image. Second, the image gradient. Third, the gradient is segmented using otsu thresholding. Fourth, the image is segmented using kmeans clustering with two clusters. Fifth, the image is segmented using kmeans clustering with three clusters. Sixth, the image is segmented using kmeans clustering with four clusters.

1.5.3 Gradient phase (optional)

Selecting gradient values by phase may be helpful if the water surface is relatively uniform. Clustering phase values into a low and high category allows water surfaces to be identified along a uniform edge, e.g. the upper and lower parts of an illuminated sheet. This can also be helpful where the water surface changes rapidly in a small region (e.g. wave breaking) when used in conjunction with a connected water surface as in §1.5.4. Separating by phase may lead to misleading results when the water surface is very steep, e.g. the meniscus effect near a vertical wall.

1.5.4 Connected water surface (optional)

It may be the goal to identify only a connected surface and remove any imaging artifacts or splash up that might identify a disconnected surface. To do this, first recognize that connectivity for binary images are easily analyzed for connectivity. Accordingly, the image mask created by finding locations where the image gradient is sufficiently large can be analyzed for connectivity. In the simplest case, the uppermost connected surface that spans the entire image may be used to identify the water surface. However, in some cases, the illuminated water surface does not span the entire image. This can lead to false positives if the widest

water surface is the incorrect one (e.g. the bottom surface of a wave tank). Use the phase of the gradient as discussed in §1.5.3 in addition to the shape to ensure the correct water surface is chosen.

1.5.5 Identifying Local Maxima & Plateaus

Local maxima are found by comparing the gradient to itself after dilation. Any pixel that has the same value before and after dilation is a local maxima or plateau. The dilation kernel used is:

$$K = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \ ;$$

such that local value must be greater than or equal to the value above and below itself. This definition of local maxima is used for simplicity, and local maxima need only be greater than or equal to the values above and below itself to implement the super-sampling method discussed in section 1.7.

This method can be expanded to find local maxima and plateaus in a larger region by expanding the kernel. For example, if the kernel was:

$$K = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \ ;$$

then the local value must be greater than the values two above and below itself, as well. In the application of PLIF, maxima in the vertical direction are of interest, and so only values in the column are compared, but this method could be extended to include information in adjacent columns.

1.6 Identifying the uppermost valid candidate

In order to select the uppermost valid surface candidate in each column, recognize that identified candidates mask have an intensity value of 1 at each location. Then, the mask can be multiplied by a weighted image which has the largest value at the top of the image, and decreases to 1 at the bottom. Then, when the images are multiplied together element-wise, the location with the largest intensity value is also the uppermost valid candidate in the column. The location of this candidate can be identified using a sorting algorithm which returns the index location of the largest value in each column.

Note that there must be a local maximum in *every* column for this to work, as most sorting algorithms return a random value if all values are equal in the group. This limitation can be avoided by returning an invalid value (-1, NaN, etc.) when there is no maximum in the column.

1.7 Stair step removal in gradient method

In order to reduce the stair step artifact resulting from digital imaging, the gradient intensity will be interpolated with a quadratic function. This can be achieved near the identified local maximum using the adjacent points to fit a quadratic. Then, the maxima of that parabola can be identified analytically. A quadratic function is chosen because it has a simple notion of concavity.

The intensity values in the column of the image are fit with a quadratic function. Accordingly, it has values and local maxima defined by:

$$I(x) = ax^2 + bx + c \tag{1.1}$$

$$x_{max} = \frac{-b}{2a} \tag{1.2}$$

The problem is posed locally by choosing three points at $x = -1$, $x = 0$, and $x = 1$ such that the coefficients a, b, and c may be found by solving the system of equations:

$$I_{j-1} = a - b + c \tag{1.3}$$

$$I_j = c \tag{1.4}$$

$$I_{j+1} = a + b + c \tag{1.5}$$

The system can be solved by noting $c = I_j$ then solving for a and be in terms of I_{j+1}, I_j, and I_{j-1}:

$$I_{j-1} = a - b + I_j \tag{1.6}$$

$$I_{j+1} = a + b + I_j \tag{1.7}$$

$$a = I_{j-1} + b - I_j = I_{j+1} - b - I_j \tag{1.8}$$

$$b = \frac{1}{2}\left(I_{j+1} - I_{j-1}\right) \tag{1.9}$$

$$a = I_{j-1} + \frac{1}{2}\left(I_{j+1} - I_{j-1}\right) - I_j \tag{1.10}$$

such that the local maxima of the parabola, interpreted as an adjustment to the local maxima identified in previous analysis, can be found:

$$\Delta x = \frac{-b}{2a} = \frac{-\frac{1}{2}\left(I_{j+1} - I_{j-1}\right)}{2I_{j-1} + I_{j+1} - I_{j-1} - 2I_j} = \frac{I_{j-1} - I_{j+1}}{2I_{j-1} + 2I_{j+1} - 4I_j} \tag{1.11}$$

This has the advantage of being easy to implement to an arbitrary precision. As long as I_j is a local maxima, the adjustment will always be between $-\frac{1}{2} < \Delta x < \frac{1}{2}$. In the cases where I_j is a plateau edge, e.g. $I_j = I_{j+1}$, then $\Delta x = \frac{1}{2}$ ()the location selected is between the two largest values).

Chapter 2

Post Processing

2.1 Introduction

At this stage, it should be noted that that the image data has been processed on the pixel scale. In pixel space, everything is (by design) sampled at integer locations. This is not the case in physical space.

You may also choose to apply a homography to the entire image before applying the techniques in §1. If so, then these concepts need not be applied and the results from image processing may be converted directly to physical quantities. Doing the analysis that way has some limitations (slow to process) but can have some advantages (pixel column corresponds to a single lateral position).

2.2 Homography

In order to calculate the corresponding physical location from pixel space, introduce the homography defined by:

$$\vec{x}'' = \boldsymbol{H}\vec{p} \quad ; \tag{2.1}$$

$$\vec{x}' = \frac{1}{z''}\vec{x}'' \quad ; \tag{2.2}$$

$$\vec{x} = \boldsymbol{R}\vec{x}' \quad ; \tag{2.3}$$

where

$$\boldsymbol{H} = \begin{bmatrix} h_{11} & h_{21} & h_{31} \\ h_{12} & h_{22} & h_{32} \\ h_{13} & h_{23} & h_{33} \end{bmatrix} \quad ; \quad \boldsymbol{R} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad ;$$

$$\vec{p} = \begin{bmatrix} p \\ q \\ 1 \end{bmatrix} \quad ; \quad \vec{x}'' = \begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} \quad ; \quad \vec{x}' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \quad ; \quad \vec{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad ;$$

in such a way that $[0, y, 0]$ is parallel to $\vec{g}$.

There is much to say about $\boldsymbol{H}$, but for practical purposes, it must be recognized that any given pixel column, i.e. $[p_0, q, 1]$ for some given p_0, is generally not parallel to any other pixel column $[p_1, q, 1]$ for some $p_1 \neq p_0$ and is generally not parallel to $\vec{g}$. This implication of this is that when the water surface is identified along any given pixel column $[p_0, q, 1]$, the identified surface location varies in both the x and y direction. This means that the water surface is not sampled uniformly in space in either the vertical or lateral direction, except in the special case where the imaging plane is normal to gravity (practically challenging to achieve). Consequently, the lateral sampling location x_0 varies with the vertical sampling location y_0. The foregoing discussion is just a long winded way of saying that both the vertical and lateral sampling position must be accounted for through the analysis.

The importance of $\boldsymbol{R}$ may be found by noting the accuracy of the imaging methods presented here are often more accurate than measurements taken with a ruler. As a result, the identified still water line may appear to have a linear trend, i.e. the reference grid used to calculate $\boldsymbol{H}$ may not be perfectly aligned with gravity. Indeed, the typical error in the Edwards Lab is 0.1mm per 300mm. Although this is small, it may be removed by finding $\boldsymbol{R}$ such that the measurements such that the linear trend is 0. It is difficult to imagine a better reference for horizontal than a still water surface.

2.3 Re-Sampling

Since the water surface has not been uniformly sampled in physical space, the overlapping physical region may have a different number of samples in either frame. To re-sample the data on a uniform grid, a few popular methods include nearest neighbor, linear interpolation, and cubic spline interpolation. These can all be achieved on an "unstructured" 2-D data set, like $\vec{x}$ in (2.3), using Qhull `http://www.qhull.org/` and its derivatives in scipy, matlab, mathematica, etc. Regardless of the method, it is useful to have uniformly sampled data.

2.4 Spatial and Temporal Alignment

In practice, it may be the case that data collected via precision imaging is more reliable than that measured with digital signal or a stick. The result is data collected from multiple trials may need a small adjustment to align the spatial and temporal axis. The functions provided find overlapping regions of the data and brute force check for a spatial and temporal offset that yield a better match for the overlapping regions. Multiple error metrics could be used as a similarity metric, but a simple average of the absolute differences seems to work well enough.

2.5 Patching

Even with excellent spatial and temporal alignment, the overlapping region should be merged into a single measurement. Suppose that the data is sampled for the same duration with

some overlapping region from a to b in space. Then, define two linear weighting functions

$$w_1 = \frac{1}{b-a}(x-a) \tag{2.4}$$

$$w_2 = \frac{-1}{b-a}(x-b) \tag{2.5}$$

$$w_1 + w_2 = 1 \tag{2.6}$$

such that $\vec{x}_{new} \to \vec{x}_1$ as $x \to a$ and $\vec{x}_{new} \to \vec{x}_2$ as $x \to b$. This approach may be used to smooth the overlapping region (a, b). Suppose that the data is sampled on x_N, then the surface may be smoothed with:

$$y_{new} = w_1 y_1 + w_2 y_2 \quad \text{where} \quad x_N \in (a, b) \tag{2.7}$$

Chapter 3

Estimation of still water depth

3.1 Introduction

The first image capture in an LIF sequence is of unique significance: at this stage, the camera calibration is completed, and the water surface is completely still. With this information, the water depth can be estimated.

3.2 The 3D problem

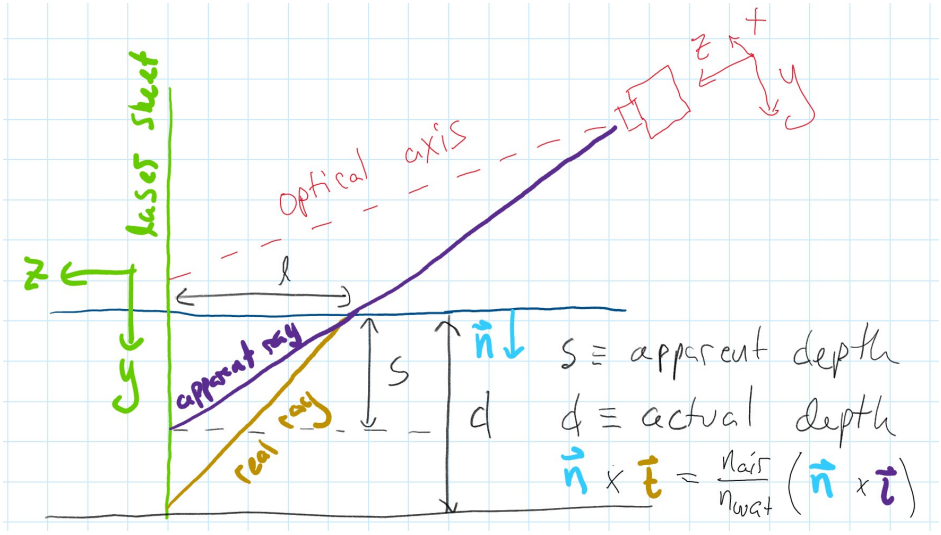

Figure 3.1: Definition sketch. Vectors in the optical coordinate, shown in red, are labeled $\vec{u}$, vectors in the laser sheet coordinate, shown in green, are labeled $\vec{X}$, the purple ray is labeled $\vec{i}$, the brown ray is labeled $\vec{t}$, the apparent depth is labeled s, the actual depth is labeled d, the normal vector to the water surface is labeled $\vec{n}$, the refractive index of air is labeled n_{air}, and the refractive index of water is labeled n_{wat}.

In three dimensions, it is easier for this problem to use linear algebra notation. Using the calibration board, it is possible to find the change of basis $\boldsymbol{R}$ where

$$\vec{x} = \boldsymbol{R}^T \vec{u}$$

where $\vec{u}$ is a pixel coordinate and $\vec{x}$ is the physical coordinate system from the calibration board. Note: here $\vec{u}$ and $\vec{x}$ are unit vectors, and so rotations are achieved, but not scaling. Using the image of the still water surface, it is possible to find the change of basis $\boldsymbol{R'}$ where

$$\vec{X} = \boldsymbol{R'}^T \vec{x}$$

where $\vec{X}$ is the physical coordinate system with X at the intersection of the still water surface and laser plane, Y is approximately inward normal to the water surface, and Z is approximately normal to the laser plane. Again, $\vec{X}$ is a unit vector.

Change of basis transformations which are pure rotations have the identity $R^{-1} = R^T$ and so

$$\vec{X} = \boldsymbol{R'}^T \boldsymbol{R}^T \vec{u}$$

i.e. $\vec{X}$ represents $\vec{u}$ in the laser plane coordinate. Choosing $\vec{u}_{bot}$ to be the pixel position of a light ray which identifies the glass bottom, the unit vector, $\vec{i}$, which represents the angle at which the light ray contacts the still water surface can be found:

$$\vec{i} = \boldsymbol{R'}^T \boldsymbol{R}^T \vec{u}_{bot}$$

using snells law in vector form, the unit vector $\vec{t}$ which propagates in the water can be found:

$$\vec{t} = \sqrt{\left(1 + \left(\frac{n_{air}}{n_{wat}}\right)^2 \left[1 - \left(\vec{n} \cdot \vec{i}\right)^2\right]\right)} \vec{n} + \frac{n_{air}}{n_{wat}} \left[\vec{i} - \left(\vec{n} \cdot \vec{i}\right) \vec{n}\right]$$

By scaling the vector $\vec{t}$ to have a depth of d, and scaling the vector $\vec{i}$ to have a depth of s, they both must traverse the same lateral distance, l. This establishes the relationship:

$$\frac{d}{s} = \frac{i_3 t_2}{i_2 t_3}$$

where i_2 and t_2 are the components of the vectors which point downward, i_3 and t_3 are the components of the vectors which point normal to the laser plane.

Lastly, note that the apparent water depth s and consequently d can be found:

$$s = (\boldsymbol{H}\vec{u}_{top} - \boldsymbol{H}\vec{u}_{bot}) \cdot \vec{n}$$
$$d = \frac{i_3 t_2}{i_2 t_3} (\boldsymbol{H}\vec{u}_{top} - \boldsymbol{H}\vec{u}_{bot}) \cdot \vec{n}$$

where $\vec{u}_{top}$ is the pixel position of the water surface, and $\vec{u}_{bot}$ is the pixel position of the glass bottom as identified in the image, and $\boldsymbol{H}$ is found in the typical way. Now, the water depth d can be calculated with the lens intrinsic parameters, a photo of a calibration grid, and a LIF photo of the still water condition.

3.3 Example

Consider the following calibration image and still water image:

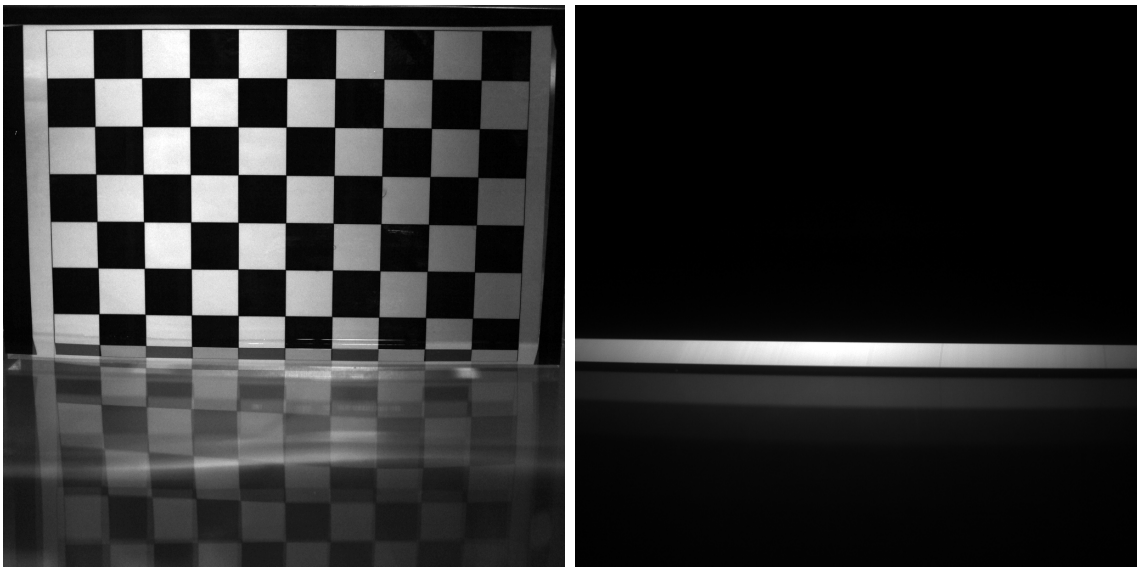

Figure 3.2: Example still water condition. Calibration board shown left, still water surface shown right.

the intrinsic camera matrix is given:

$$\boldsymbol{A} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1895.89 & 0 & 503.09 \\ 0 & 1895.78 & 524.40 \\ 0 & 0 & 1 \end{bmatrix} \quad ;$$

the rotation matrix from pixel coordinates to calibration grid coordinates is calculated

$$\boldsymbol{R}^T = \begin{bmatrix} 0.99947801 & -0.00392189 & -0.03206767 \\ 0.00959514 & 0.98385068 & 0.17873383 \\ 0.03084882 & -0.17894823 & 0.98337474 \end{bmatrix} \quad ;$$

and the rotation matrix from calibration grid coordinates to world coordinates is calculated

$$\boldsymbol{R'}^T = \begin{bmatrix} 0.999928 & 0.01199971 & 0. \\ -0.01199971 & 0.999928 & 0. \\ 0. & 0. & 1. \end{bmatrix} \quad ;$$

the apparent depth is calculated for all pixel positions is calculated

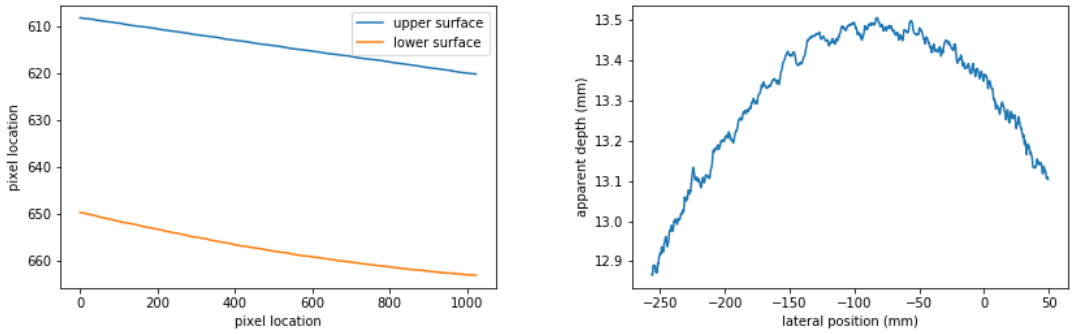

Figure 3.3: Apparent depth from example. Pixel location of upper and lower surface shown left, apparent water depth shown right.

lastly, the actual water depth can be estimated

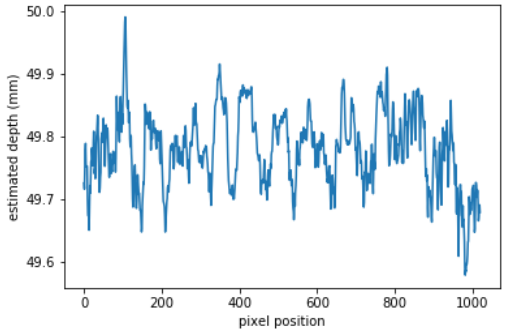

Figure 3.4: Estimated water depth from example.