Department of Physics and Astronomy

University of Heidelberg

Master thesis

in Computer Engineering

submitted by

Habib Gahbiche

born in Sousse

2018

# (Title)

# (of)

# (Master thesis)

This Master thesis has been carried out by Habib Gahbiche

at the

Institute of Environmental Physics

under the supervision of

Prof. Dr. Bernd Jähne

and

Prof. Dr. Karl-Heinz Brenner

Institute of Computer Engineering

**(Titel der Masterarbeit - deutsch):**

(Abstract in Deutsch, max. 200 Worte. Beispiel: **?**)
Lorem ipsum dolor sit amet, consectetur adipisici elit, sed eiusmod tempor incidunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquid ex ea commodi consequat. Quis aute iure reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint obcaecat cupiditat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.
Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi.

**(Title of Master thesis - english):**

(abstract in english, at most 200 words. Example: **?**)
Lorem ipsum dolor sit amet, consectetur adipisici elit, sed eiusmod tempor incidunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquid ex ea commodi consequat. Quis aute iure reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint obcaecat cupiditat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.
Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi.

# Contents

# 1 Introduction

Air-water gas exchange is important in many contexts in nature and engineering. Climate change is probably one of the most prevailing topics involving air-water exchange between the atmosphere and ocean (cite NIWA 2016).

The transfer rate of most gases between the atmosphere and ocean is controlled by processes just beneath the water surface. Wind blowing over the sea is the main driving force for all relevant physical processes. When the water surface is highly turbulent, gases can be more rapidly transferred toward or away from the surface. In particular, turbulent regions can generate breaking waves which in turn create bubbles that trap air from the atmosphere into the ocean (cite Terray E. Donelan 1996). These bubbles enlarge the air-water interface due to their additional surface under water. However, this does not necessarily mean that gas exchanges will occur for the whole bubble lifetime. In fact, smaller bubbles can reach equilibrium with the surrounding water and stop exchanging gases (cite Mischler). Therefore, determining bubble radii is crucial to properly model bubble induced gas exchange (Deane and Stokes 2002).

Previously developed bubble measuring methods in wind wave facilities include light scattering based methods (Jähne, Weis 1984), depth from focus methods (Geißler 1994) and more recently, light field methods (Mischler 2014). Other field methods such as R. Al-Lashi (2015) and Leifer et al. (2003) introduced instruments for imaging bubbles within breaking waves. These methods rely on thresholding and edge detection algorithms to produce binary images, where further techniques such as Hough transoform (cite Hough) are applied to extract bubble distributions. Many of these methods, however, are very sensitive to background illumination, which either reduces their accuracy (Zhong 2016).

The goal of this work is to develop a measurement technique that determines the bubble spectrum in a given measurement volume, i.e. determine bubble concentrations as a function of their radii in a well determined volume. In particular, we make use of the recent developments in object detection algorithms, while relying on machine learning for bubble classification in order to develop an accurate and robust bubble measuring technique.

In this work, we first discuss the relevant theoretical basics in chapter 2 starting with the physics behind capturing bubble images, followed by basics in image processing. Next, we briefly discuss related measurement techniques in chapter 3 and their importance to our work. In chapter 4, we describe our experimental setup in detail. Our proposed algorithm that analyses bubble images is described thoroughly in chapter 5.

# 2 Theory

In this chapter we explain the theoretical concepts relevant to this thesis. We start with explaining the physics behind our method in section 2.1, in particular how bubbles interact with light. Next, we discuss the mathematical basics necessary for image processing such as Fourier theory and edge detection in section 2.2. Section 2.3.1 explains the principle behind machine learning that our method relies on for classification. Finally, section 2.3 formally introduces the object detection problem and our chosen criteria for evaluation.

## 2.1 Bubble physics

In this section we briefly discuss the physical laws relevant to light-bubble interaction.

### 2.1.1 Reflection and Refraction

**Reflection** is the abrupt change in the direction of propagation of a light beam that strikes the boundary between two different media. Assuming the incoming light ray makes an angle $\theta_1$ with the normal of a plane tangent to the boundary, then the reflected ray makes an angle $\theta_1'$ with this normal and lies in the same plane as the incident ray and the normal. The reflection law is shown in figure 2.1a and is described as

$$\theta = \theta' \tag{2.1}$$

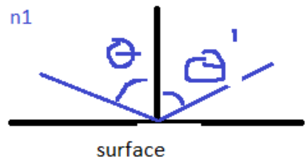
**Refraction** is the change in direction of propagation of a wave when the wave passes from one medium into another with a different index of refraction. The angle of the reflected beam is shown in figure 2.1b is given by Snell's law:

$$n_1 \sin(\theta_1) = n_2 \sin(\theta_2) \tag{2.2}$$

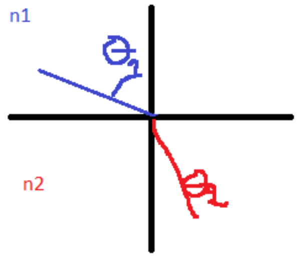
Where $n_1$ and $n_2$ are the indices of refraction of the media.

**Total reflection** occurs when angle of the incident beam is larger than the critical angle:

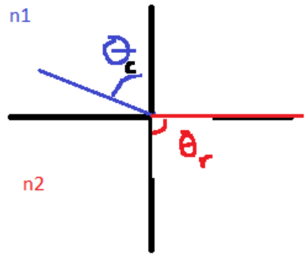
$$\theta_c = \arcsin\left(\frac{n2}{n1}\right) \tag{2.3}$$



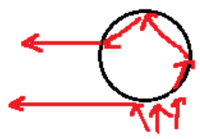

(a) Reflection: incident light beam gets reflected with the same angle relative to surface normal.



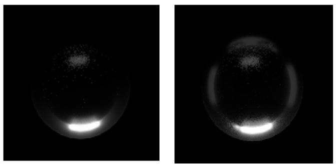

(b) Refraction: refracted light beam gets refracted according to Snell's law




(c) Critical angle is reached when refracted angle is 90°



(a) Light beams inside a bubble. Light paths are described with refraction and total reflection laws.



(b) Simulated bubbles with ray tracing with two different lighting conditions. Left: no ambient light. Right: with ambient light

Figure 2.2: Interaction of a light bubble with light rays

### 2.1.2 Bubble-light interaction

Since the smallest bubble radius (around 50-100 $\mu m$) is much larger than the wavelength of our light source (around 600 nm), diffraction within the air bubble can be neglected. Mie scattering effects are also only relevant for radii within the same order of magnitude as the wavelength (Demtroeder 2) so it can be neglected as well. Therefore, only refraction and reflection laws will be considered.

Figure 2.2a shows refracted rays inside an air bubble. Since the outside medium (water) is more dense, total refraction occurs at the lower bubble boundary. Simulation in figure 2.2b also shows that when a bubble is lit from bellow, two peaks can be observed. The lower peak is strong because it arises from total reflection on the lower bubble boundary, whereas the upper peak is much weaker. The second peak arises from internal (partial) reflections within the bubble and can be best explained by the Fresnel equations (Demtroeder 2). The different peaks characteristics will be exploited by our proposed algorithm in order to compute the bubble's depth and radius.

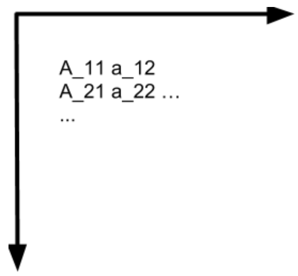



Figure 2.3: Notation and coordinate system

## 2.2 Image processing

In the following we represent an image as a two dimensional signal written as a matrix $\boldsymbol{g}$. $g_{m,n}$ denotes the pixel (i.e. picture element) at the $m$-th row corresponding to the $n$-th column. The chosen coordinate system is described in figure 2.3.

### 2.2.1 Fourier theory

The Fourier transform is an important image processing tool which is used to decompose an image into its since and cosine components. The output of the transformation represents the image in the Fourier or frequency domain, while the input image is the spacial domain. In the Fourier domain image, each point represents a particular frequency contained in the spatial domain image. The *continuous* two-dimensional Fourier transform is defined as

$$\mathscr{F}\{g(\mathbf{x})\} = \hat{g}(\mathbf{k}) = \int_{-\infty}^{\infty} g(\mathbf{x})\exp\left(-2\pi\mathrm{i}\mathbf{k}^T\mathbf{x}\right) \mathrm{d}\mathbf{x} \tag{2.4}$$

and the inverse Fourier transform

$$\mathscr{F}^{-1}\{\hat{g}(\mathbf{k})\} = g(\mathbf{x}) = \int_{-\infty}^{\infty} \hat{g}(\mathbf{k})\exp\left(-2\pi\mathrm{i}\mathbf{k}^T\mathbf{x}\right) \mathrm{d}\mathbf{x} \tag{2.5}$$

Where $\mathbf{x}$ and $\mathbf{k}$ are the two dimensional space and frequency vectors respectively.

Images however are discrete two dimensional signals, we therefore need to apply the *Discrete* Fourier transform or DFT, defined as

$$\mathrm{DFT}\{g_{m,n}\} = \hat{g}_{u,v} = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} g_{m,n}\, \exp\left(-\frac{2\pi\mathrm{i}mu}{M}\right) \exp\left(-\frac{2\pi\mathrm{i}nu}{N}\right) \tag{2.6}$$

Similarly, the inverse 2-D DFT is defined as

$$\mathrm{IDFT}\{\hat{g}_{u,v}\} = g_{m,n} = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \hat{g}_{u,v}\, \exp\left(\frac{2\pi\mathrm{i}mu}{M}\right) \exp\left(\frac{2\pi\mathrm{i}nu}{N}\right) \tag{2.7}$$

### 2.2.2 Convolution

Convolution is one of the most important operations in signal processing. Convolving two signals $g$ and $h$ produces a third signal that expresses how the shape of one is modified by the other. Formally, we define the continuous convolution as follows

$$(g \star h)(\mathbf{x}) = \int_{-\infty}^{\infty} h(\mathbf{x}')g(\mathbf{x} - \mathbf{x}')\mathrm{d}\mathbf{x} \tag{2.8}$$

and the discrete two dimensional convolution as

$$g'_{m,n} = \sum_{m'=0}^{M-1} \sum_{n'=0}^{N-1} h_{m',n'} g_{m-m',n-n'} \tag{2.9}$$

One important property of convolution is that we can express it as a multiplication in the Fourier domain.

$$\mathscr{F}\{g \star h\} = NM\hat{h}\hat{g} \tag{2.10}$$

This property, together with the fast Fourier implementation of the Fourier transform allows a fast computation of convolutions.

At the edge of the image, we typically extend the image with zero values (i.e. zero padding). This introduces an error when applying filters at the image border and we will mostly exclude the border when using filters (see chapter 5 for more details).

### 2.2.3 Smoothing

Smoothing an image means convolving an image with a smoothing filter. A smoothing or averaging filters must ideally fulfill following conditions

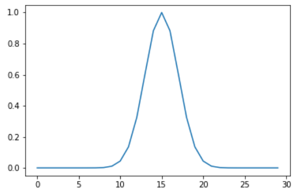
1. Zero-shift: $\Im(\hat{h}(\mathbf{k})) = 0$
2. Preservation of mean value: $\hat{h}(0) = 1$
3. Monotonous decrease: $\hat{h}(k_1) \leq \hat{h}(k_2)$ for $k_2 > k_2$
4. Isotropy: $\hat{h}(\mathbf{k}) = \hat{h}(|\mathbf{k}|)$ stimmt das ??

In this work, we will be using Gaussian filters for one and two dimensional smoothing. Although Gaussian filters are not ideal, e.g. isotropy is violated for small standard deviations, it is still a good approximation for an ideal low pass filter. Computing the Fourier transform (for convolution) is also faster for a Gaussian filter. The $m$-th component of a one dimensional Gaussian filter mask can be obtained from the Gaussian function

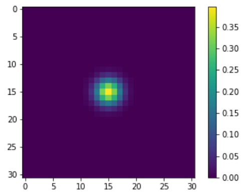
$$G_m = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(m-\mu)^2}{2\sigma^2}\right) \tag{2.11}$$

Where $\mu$ is the mean, i.e. Gaussian peak's position and $\sigma$ is the standard deviation, i.e. peak's width.

Figure 2.4 show a Gaussian curve in one and two dimensions as well as the result of convolving an image with a Gaussian filter mask. Note how the image becomes blurry, i.e. large wave numbers have been suppressed.




(a) 1D Gaussian signal with $\mu = 15$ and $\sigma = 2$




(b) 2D Gaussian signal with $\mu_x = \mu_y = 15$ and $\sigma_x = \sigma_y = 2$

(c) Original image

(d) After convolution with 2D Gaussian mask

Figure 2.4: Gaussian smoothing filter

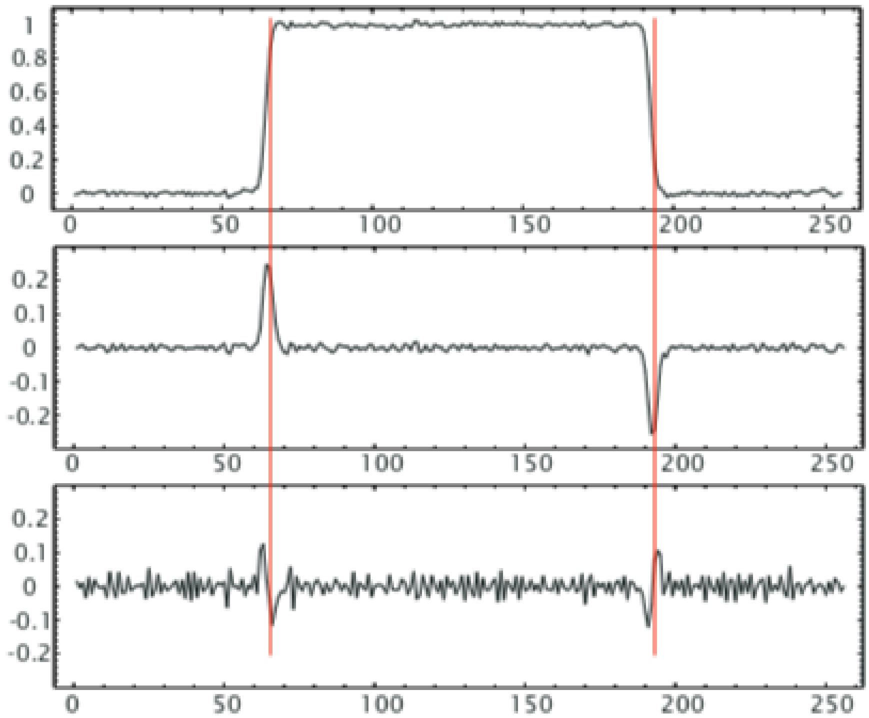



Figure 2.5: Original 1D signal. First derivative. Second derivative

### 2.2.4 Edges and Derivation

An edge can be defined as a set of continuous pixel positions where an abrupt change of intensity (i.e. gray value) occurs. Therefore, edge detection is based on differentiation, where in discrete images differentiation is replaced by discrete differences that are mere approximation to differentiation. There is also the need to not only know where edges are, but also how strong they are. Figure **??** shows that in the one dimensional case, edges can be detected by applying first and second derivatives to the signal.

In continuous space, a partial derivative operation is defined as

$$\nabla = \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right] \tag{2.12}$$

and its corresponding Fourier transform is

$$\mathscr{F}\{\nabla\} = 2\pi \mathrm{i}\mathbf{k} \tag{2.13}$$

For the second derivative we need to consider all possible combinations of second order partial differential operators of a two dimensional signal. The resulting $2 \times 2$

matrix is called the Hessian matrix

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2}{\partial x^2} & \frac{\partial^2}{\partial x \partial y} \\ \frac{\partial^2}{\partial y \partial x} & \frac{\partial^2}{\partial y^2} \end{bmatrix} \tag{2.14}$$

and its Fourier transform is

$$\mathscr{F}\{\mathbf{H}\} = -4\pi^2 \mathbf{k}\mathbf{k}^T \tag{2.15}$$

Edge detectors can be implemented as filters $h$ that operate on a two dimensional grid. From the above equations we can derive the general properties for these filters:

1. Zero-shift:
   - 90° phase shift for first order derivative, implying $\Im\{\hat{h}\} \neq 0$ and an antisymmetric filter mask, i.e $h_{-n} = -h_n$
   - a second order derivative operator must be symmetric in order to satisfy the zero shift property, i.e. $h_{-n} = h_n$
2. Suppression of mean value: $\hat{h}(k_i = 0) \Leftrightarrow \sum_{\mathbf{n}} h_{\mathbf{n}} = 0$
3. isotropy: For good edge detection, the edge detector's response must not depend on the direction of the edge.
   - first order derivative $\hat{h}(\mathbf{k}) = \pi \mathrm{i} k_i \hat{b}(|\mathbf{k}|)$
   - second order derivative $\hat{h}(\mathbf{k}) = \pi^2 k_i^2 \hat{b}(|\mathbf{k}|)$

   where $k_i$ denotes the wave number in the $i$-th direction and $b$ is an isotropic smoothing filter that fulfills the conditions

$$\hat{b}(\mathbf{0}) = 1, \qquad \nabla_k \hat{b}(|\mathbf{k}|) = \mathbf{0} \tag{2.16}$$

In this work we will be using the Sobel filter masks as defined in equation (2.17) in order to compute derivatives in $x$ and $y$ directions.

$$S_x = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \qquad S_y = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & 2 \\ 1 & 0 & -1 \end{bmatrix} \tag{2.17}$$

At each point in the image, the resulting gradient approximations can be combined to give the gradient magnitude image $S$ using:

$$S = \sqrt{(S_x \star G)^2 + (S_y \star G)^2} \tag{2.18}$$

where G is the input image. Figure 2.6 shows the result of applying the derivation operator on an image. Note how edges have different magnitude depending on their strength.

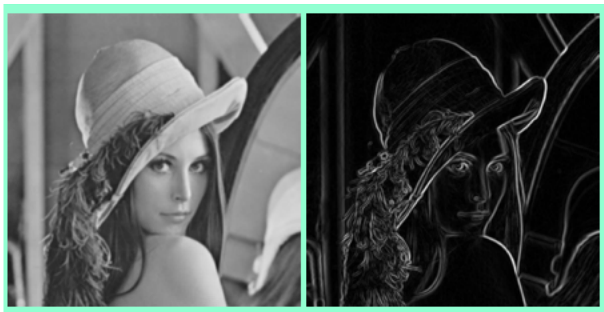


Figure 2.6: Left: original image. Right: gradient image

Figure 2.7: Ideal local neighborhood described by a unit vector $\tilde{\mathbf{n}}$

### 2.2.5 Orientation and Structure Tensor

Although derivation is useful to determine the gradient magnitude and its direction in an image, it doesn't tell us much about gradient directions in a specific neighborhood of a point. Figure 2.7 shows that in a neighborhood with ideal orientation gray values change in one direction only. Generally, the direction of local orientation can be denoted with a unit vector $\tilde{\mathbf{n}}$. If we orient the coordinate system along the principal directions, the gray values become a one dimensional function and a simple neighbourhood can be represented by

$$g(\mathbf{x}) = g(\mathbf{x}^T \tilde{\mathbf{n}}) \tag{2.19}$$

The drawback of this representation however, is that it cannot distinguish between neighborhoods with constant values and isotropic orientation distribution. So if we define the optimum orientation as the orientation that shows the least deviations from the directions of the gradient, we can expressed it using the unit vector $\tilde{\mathbf{n}}$ as

$$\nabla g^T \tilde{\mathbf{n}} = \cos[\angle(\nabla g, \tilde{\mathbf{n}})] \quad \Leftrightarrow \quad \left(\nabla g^T \tilde{\mathbf{n}}\right)^2 = |\nabla g|^2 \cos^2[\angle(\nabla g, \tilde{\mathbf{n}})] \tag{2.20}$$

We can see that this quantity is maximized when the orientation is along the unit vector $\tilde{\mathbf{n}}$, i.e. when $\nabla g$ and $\tilde{\mathbf{n}}$ are either parallel or antiparallel. Therefore, the following integral is maximized in a local neighborhood:

$$\int w(\mathbf{x} - \mathbf{x}') \left(\nabla g(\mathbf{x}')^T \tilde{\mathbf{n}}\right)^2 \mathrm{d}\mathbf{x}' \tag{2.21}$$

where the window function $w$ determines the size and shape of neighborhood around a point $\mathbf{x}$ in which the orientation is averaged. The maximization problem must be solved for each point $\mathbf{x}$, so we can write the maximization problem as follows:

$$\tilde{\mathbf{n}}^T \mathbf{J} \tilde{\mathbf{n}} \rightarrow \max \tag{2.22}$$

From equation 2.21 and 2.22 we can define **the structure tensor** as

$$\mathbf{J} = \int w(\mathbf{x} - \mathbf{x}') \left( \nabla g(\mathbf{x}') \nabla g(\mathbf{x}')^T \right) \mathrm{d}\mathbf{x}' \tag{2.23}$$

The $pq$-th component of this tensor is therefore given by

$$J_{pq} = \int_{-\infty}^{\infty} w(\mathbf{x} - \mathbf{x}') \left( \frac{\partial g(\mathbf{x}')}{\partial x'_p} \frac{\partial g(\mathbf{x}')}{\partial x'_q} \right) \mathrm{d}\mathbf{x}' \tag{2.24}$$

Rotating equation 2.22 into principle coordinate system yields:

$$\begin{bmatrix} n'_1 & n'_2 \end{bmatrix} \begin{bmatrix} J'_{11} & 0 \\ 0 & J_{22} \end{bmatrix} \begin{bmatrix} n'_1 \\ n'_2 \end{bmatrix} = J' = J'_{11} n'_1 + J'_{22} n'_2 \rightarrow \max \tag{2.25}$$

We can see that $J'$ is maximized for $\tilde{\mathbf{n}} = [1 \quad 0]^T$ (assuming $J'_{11} > J'_{22}$), where the maximum value is $J_{11}$, so solving this problem is equivalent to solving the eigenvalue problem for $\mathbf{J}$. We can then extract the orientation $\theta$ as follows:

$$\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} J_{11} & J12 \\ J_{21} & J_{22} \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{2.26}$$

Using trigonometric identities, this yields

$$\tan(2\theta) = \frac{2 J_{12}}{J_{11} - J_{22}} \tag{2.27}$$

For discrete images, we use Sobel filters as defined in equation 2.17 for derivation and a Gaussian smoothing mask introduced in section 2.2.3. Computing the elements of a structure tensor for an image $G$ therefore requires following steps:

1. $G_x = S_x \star G$
   $G_y = S_y \star G$
2. $J_{11} = M \star (G_x \times G_x)$
   $J_{12} = J_{21} = M \star (G_x \times G_y)$
   $J_{22} = M \star (G_y \times G_y)$

Where $M$ is a smoothing mask and $S_x$ and $S_y$ are derivation masks in $x$ and $y$ directions respectively.

Figure 2.8 shows an extracted orientation from a bubble image using the structure tensor.

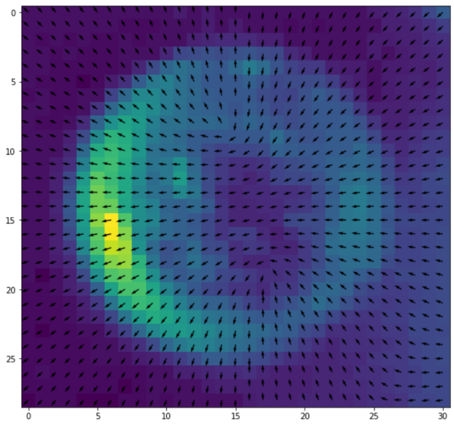



Figure 2.8: Orientation angle using sobel filters for derivation and a Gaussian mask with $\sigma = 1$ for smoothing.

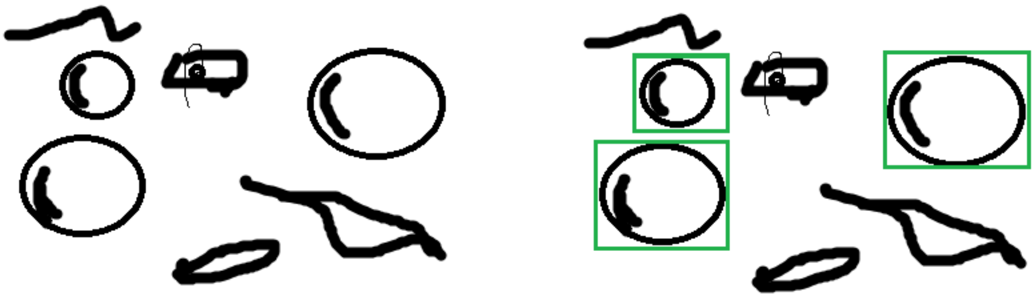


Figure 2.9: Left: Input image. Left: Output of an object detection algorithm dawn as bounding boxes around bubbles

## 2.3 The object detection problem

Our proposed algorithm recognizes bubbles (classification) in an image and estimates their respective centers and radii (localization). This problem of classification and localization is known as the object detection problem. Figure 2.9 shows a typical output of an object detection algorithm.

There has been a lot of progress in this field thanks to deep learning algorithms (see next section) that rely on training a relatively complex model a large amount of annotated data. The sate of the art algorithms include region based methods such as Faster R-CNN (cite **?**), where many candidate regions are first extracted and then classified and single evaluation methods such as YOLO (cite **?**) where bounding boxes and class probabilities are estimated with one single neural network in a single evaluation. Although these algorithms perform very well on typical photographs and support between 1000 and 9000 different classes, applying them to our problem yields very bad results (see chapter 5). We still however opt for a machine learning based approach for the classification part, and then perform localization with more straightforward methods.

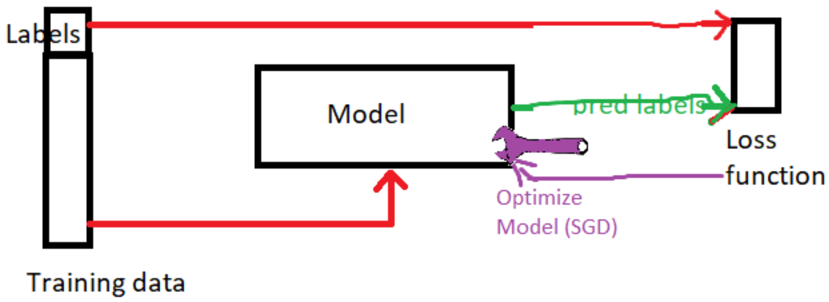
### 2.3.1 Classification and Machine Learning

In our work, we use machine learning techniques mainly for signal classification purposes. For instance, both of the proposed algorithms rely on binary classification using logistic regression and convolutional neural network. In this section, we explain the principle behind machine learning and the main idea behind the chosen model. A more thorough argumentation about the chosen architecture and model-specific parameters can be found in chapter 5.

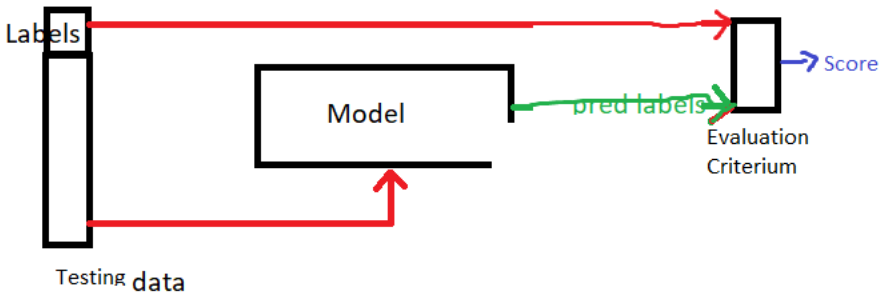
Machine learning is commonly defined as the practice of using algorithms to parse data, learn from it, and the make a determination or prediction about something. In our case, this means training our model with a large number of bubble and background instances, in order to predict whether a given signal corresponds to a bubble or not. Figure 2.10 summarizes the principle behind machine learning.

#### Terminology

- **Training** is changing the model's parameters based on the model's output. Training is also referred to as optimization. When training data is annotated, we speak of supervised training.
- **Testing** is assigning a score to a trained model based on annotated testing data. The score is typically by an evaluation criteria and is independent of the model itself.
- **Validation** is a form of testing, so it assigns a score to a trained model. Validation is needed when we want to choose the best model among some trained candidate models based on their testing performance. Since changing




(a) Training




(b) Testing

Figure 2.10: Training and testing in machine learning

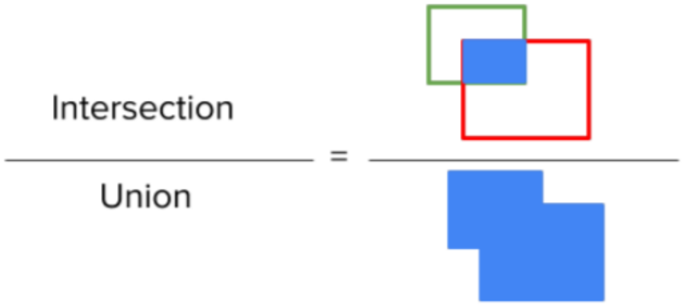



Figure 2.11: Definition of intersection over union. Green represents predicted bounding box and red is the ground truth. Areas are drawn in blue

a parameter of the algorithm (in this case the whole model) based on the output is equivalent to training by definition, we need an independent data set to compute the final score of the algorithm, i.e. validation data.

### Evaluation Criteria: IoU@p-mAP

We evaluate our object detection algorithm using the *intersection over union at p mean average precision* criteria as shown in figure 2.11. As the name suggests, we compute the area of intersection between the predicted and the ground truth bounding box and then divide it by union area of the two boxes. This means that IoU $= 1$ corresponds to a perfect prediction and IoU $= 0$ is a complete miss (assuming original images were annotated perfectly). So we choose a threshold $p$, so that IoU $>= p$ or simply IoU@p corresponds to a correct prediction or *true positive*.

*Mean average precision* is the average of the maximum precision at different recall values, where **precision** and recall are defined as

$$precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{2.28}$$

$$recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{2.29}$$

From the above definition it becomes clear that precision measures how accurate the predictions are and recall measures how well all the positives can be found.

Since our algorithm performs binary classification, *mean average precision* is equivalent to *average precision* in our case.

**Note:** This criteria only shows how well the algorithms is at detecting bubbles from an image, and does not necessarily make a strong statement on how well the measurement technique as a whole performs at estimating bubble size distributions in a water.

# 3 Related Work

In this chapter we briefly discuss two similar bubble measurement techniques that aim to estimate bubble concentrations in water. The first is a master's thesis by (cite Leonie), where bubbles, where a bright field method was used in lab, so that bubbles appeared as dark circles on camera, and the second is a field measurement technique (cite R. Al-Lashi, S. Gunn), where bubbles were artificially lit from below, yielding similar images to our own.

## 3.1 Bright field method

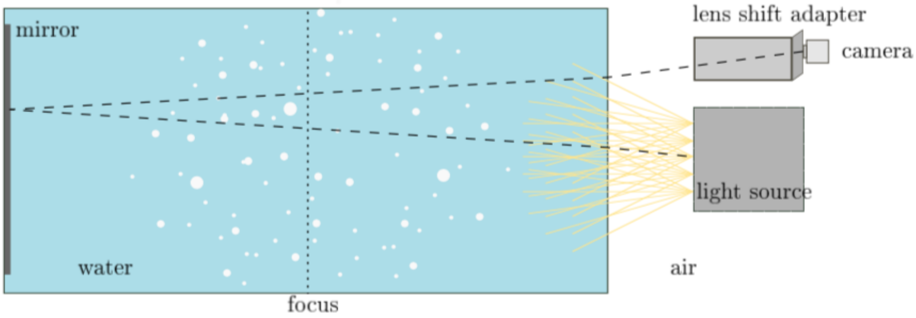
As an improved method from (cite Michler), this technique uses a bright field method as shown in figure **??**. Bubbles is this experimental setup are backlit, so that they appear completely dark except for one spot in the center (see section 2.1). Note the need to use a lens shift adapter because the incoming rays are not parallel.

Depth of field calibration was based on the edges between bubbles and background. The higher the edge's magnitude, the closer the bubble is to the focus plane. A calibration target made of hollow circles was used to calibrate the depth of field (figure 3.1c).

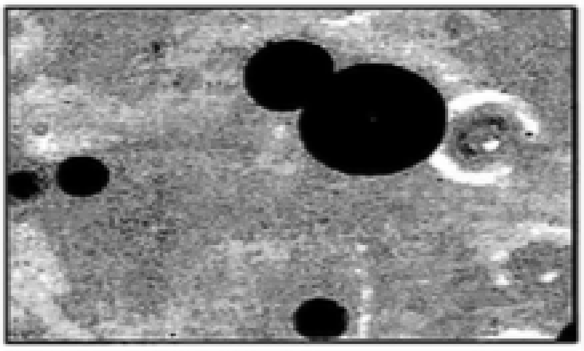
The images obtained by this method are arguably the most important advantage of this technique, since it reduces bubbles to simple circles, making them relatively easy to detect, whereas our algorithms need to recognize significantly more complex patterns than mere circles. In contrast to our method however, the bright field technique constrains how close the camera can get to the water surface. Also, (cite Leonie) does not define a criteria that measures how well the circle detection algorithm performs, so the assumption is that precision is close to perfect, which introduces an error to bubble concentration measurement that was essentially neglected. Nevertheless, this work was very important for newly developed method because it offered an estimation of the boundary conditions, such as the largest possible bubble radius and the largest possible bubble concentration, that were important to determine the different hyperparameters of our algorithm and therefore contributing to the accuracy of our algorithms.

## 3.2 Bubble optical imaging instrument

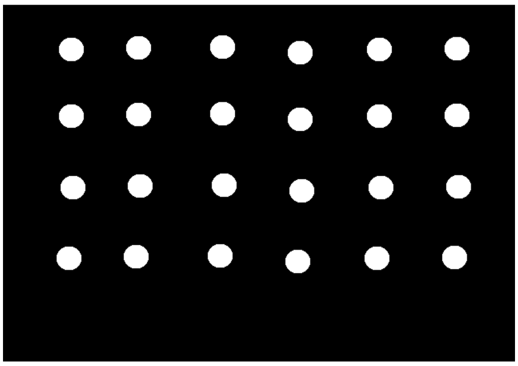
This technique submerges an optical imaging instrument (figure 3.2a) underwater and illuminates a thin slice of water of volume 4cm $\times$ 4cm $\times$ 5mm. We found this method particularly interesting because it produces images similar to our own.




(a) Experimental setup of a light field method at Aeolotron facility

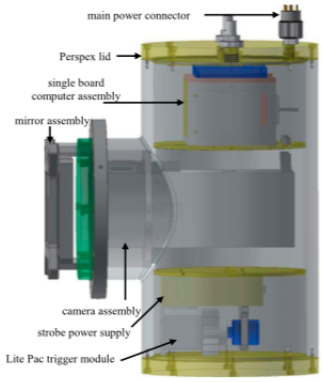


(b) Sample result image

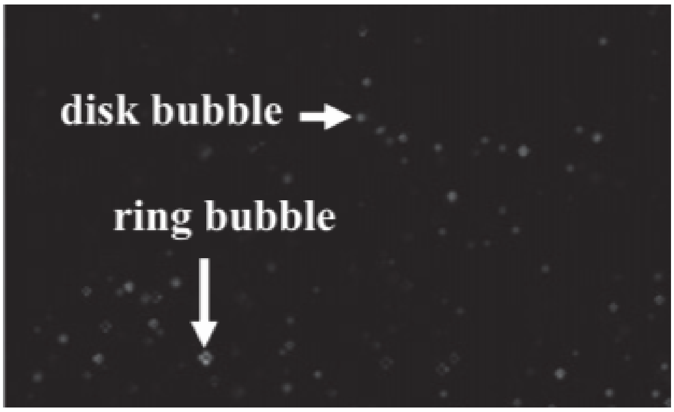


(c) calibration target

Figure 3.1: Bright field method




(a) Bubble optical imaging instrument. The submerged device




(b) Sample result image containing two types of bubbles: filled circles (disks) and rings.

Figure 3.2: Bubble detection with the bubble optical imaging instrument

However, applying the algorithm from this paper did not yield good results. This is most likely due to the fact that our method does not illuminate the bubbles from above, making the bubble's circular shape not easily detectable with a Hough transform (cite Hough transform, Al-Lashi)

# 4 Experimental Setup

## 4.1 Requirements

In order to improve on previous works (cite Leonie, section 3), we require that our method fulfills following criteria:

- **R1: Minimal distance to water surface**: This requirement defined our experimental setup. As shown in figures 4.1 and 4.2, lighting bubbles from below gives more freedom for the camera to be closer to the water surface. This is more apparent when the camera faces the surface with a certain angle.
- **R2: Consistent image results**: Bubbles can look very different under different lighting conditions. For instance, bubbles might appear as dark disks or bright rings (chapter 3) depending how they are lit. We therefore require that the angle between camera and light source is always constant. This requirement guarantees a consistent pattern for bubbles across different water tanks.
- **R3: Images must contain all possibly retrievable information**. From our simulation (figure 2.2b) we know that bubbles are characterized by two peaks. Both of these – especially the upper, dimmer one – are required to be visible. This requires the camera to have a high signal-to-noise ratio and/or the light source to be very bright.

## 4.2 Aquarium Setup

The goal of this setup is to capture bubble images that can be used to prototype our algorithm, so we built the experiment in a transparent $1m \times 1m \times 1m$ water tank as shown in figure 4.1 where we could change most of the parameters such as lighting angle, camera and bubble concentration independently from each other. Measurement results are discussed in next chapter and image analysis is discussed in chapter 5.

In order to satisfy requirements in section 4.1, we opted for parameters as summarized in table 4.1.

We chose the Basler A1920 mainly because of its high signal-to-noise ratio, and therefore fulfilling requirement R3 given enough light. Its main drawback is its relatively low frame rate at the highest possible resolution, which makes bubble tracking more difficult, which in turn makes radius calibration more difficult (see

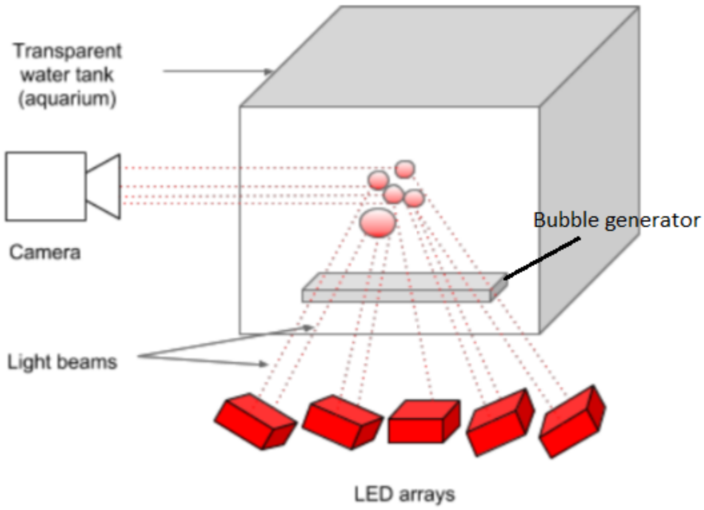



Figure 4.1: Experimental setup illustrating the 90° angle between the light source (LED array) and the camera. Note how LEDs are also oriented towards the camera's field of view to maximize the amount of light reaching the camera and therefore reduce noise.

section 5.4). Compared to an available high-speed camera with comparable maximum resolution, (PCO Dimax, 1,2kHz at $2000 \times 2000$ resolution) our chosen camera also has a smaller sensor. This means that the magnification factor is higher and therefore smaller bubbles can be better detected. A 100 mm macro lens was used in this setup. The large focal length yields a large magnification factor. Bubbles were produced by a bubble generator. Bubble radii therefore depend on the amount of air flowing through the bubble generator, measured in liter per minute or LPM. We acquired two sets of measurements, the first with a low bubble concentration, (1,8 LPM air flow), and the second with a much higher bubble concentration (3 LPM).

## 4.3 Aeolotron Setup

The Aeolotron is an experimental annular wind-wave facility with a diameter of about 10m and a width of about 60cm. Typical water height is 1m. Four wind engines generate reference wind speeds of up to 17m/s, enough to generate breaking waves that can produce air bubbles. Figure 4.2 shows a schematic view of the wind facility as well as the experimental setup.

The goal of this setup is to apply the knowledge gained from the previous setup on bubbles emerging from breaking waves, so we reproduced the setup from the aquarium at the Aeolotron, while keeping as many unchanged parameters as possible (e.g. same camera, same light source same lens etc...).

We also used a bubble generator first, to verify that the algorithm developed for the previous setup still works with measurements from the Aeolotron setup. Some

| | |
|---|---|
| Camera | Basler A1920-155um |
| Focal length | 100 mm |
| F-number | 5,6 |
| Exposure time | 100 us |
| Gamma | 0,8 |
| Resolution [px] | 1920 x 1200 |
| Sensor size | 11.3 mm x 7.1 mm |
| Air flow | 1,8 LPM / 3 LPM |
| Magnification factor | 30 um/px |
| Framerate | 100 Hz |
| Resulting field of view | 5,7cm x 3,6 cm |

Table 4.1: Parameters for aquarium setup

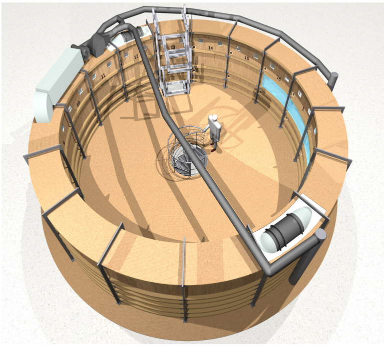
| | |
|---|---|
| Camera | Basler A1920-155um |
| Focal length | 100 mm |
| F-number | 5,6 |
| Exposure time | 130 us |
| Gamma | 0,7 |
| Resolution [px] | 1920 x 1200 |
| Sensor size | 11.3 mm x 7.1 mm |
| Air flow | 1,8 LPM |
| Magnification factor | 10 um/px |
| Framerate | 100 Hz |
| Resulting field of view | 1,9 cm x 1,2 cm |

Table 4.2: Parameters for Aeolotron setup

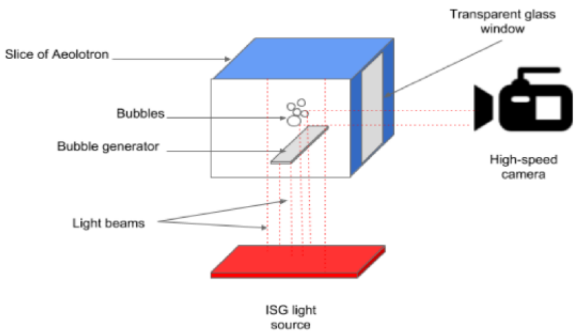
parameters such as focus distance and angle between the light source and bubbles were inevitably changed, which means we needed to calibrate the depth of field again. Table 4.2 summarizes the setup's parameters.

## 4.4 Measurement result

Figures 4.3, 4.4 and 4.5 show different images captured using a bubble generator in two different water tanks. For low bubble concentrations, images look very similar across the two different setups. However, when the concentration is very high (figure 4.4) bubbles change dramatically. The curvature can be easily distinguished and scattered light from other bubbles gives bubbles a more circular shape. Also, the background is much brighter for a high bubble concentration. The exact bubble shape will be discussed further in chapter 5.

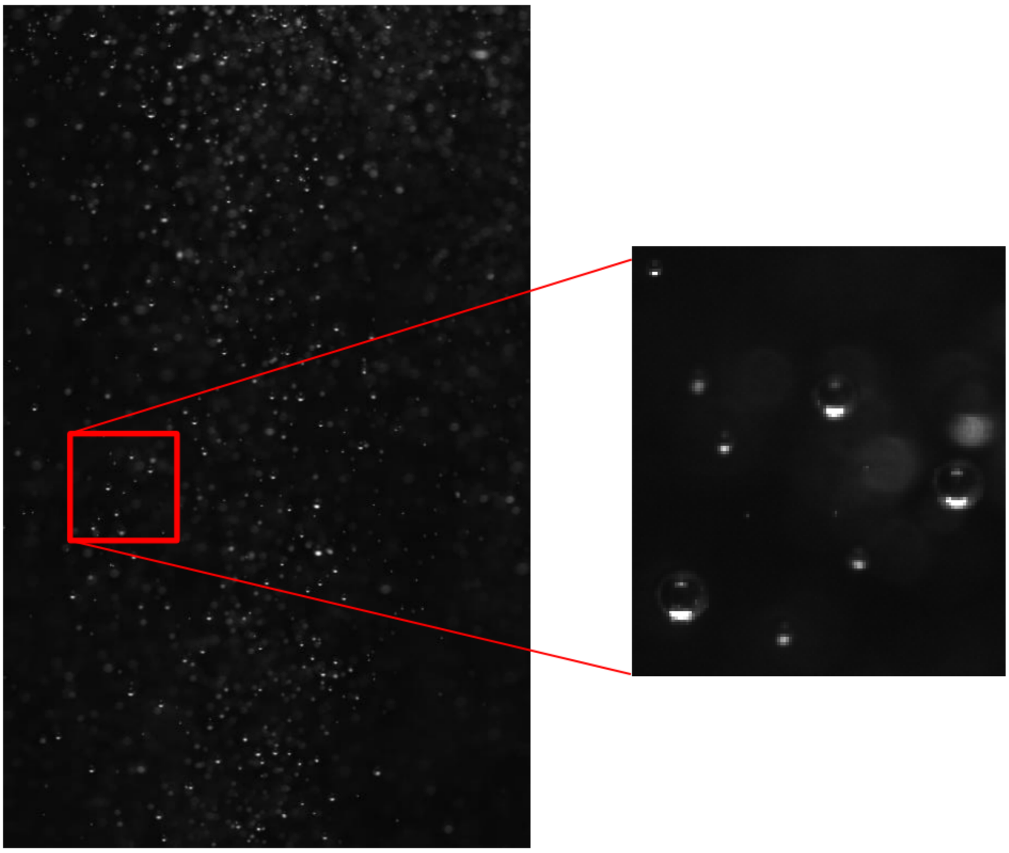


(a) Aeolotron wind wave facility

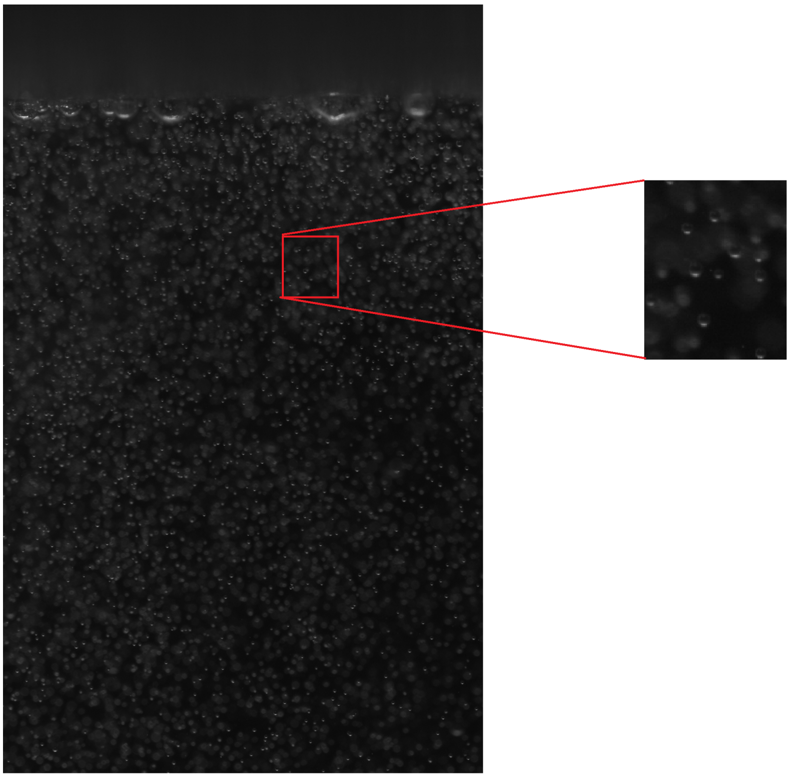



(b) Experimental setup

Figure 4.2: Aeolotron facility and experimental setup



(a) Exposure time = 100us, Gamma = 0,6. Both peaks are dimmer than in figure 4.3b



(b) Exposure time = 200$\mu s$, Gamma = 1. Lower peak is saturated but upper peak can be clearly seen

Figure 4.3: Sample image captured with aquarium setup for a low bubble concentration. Note how bubbles are characterized by one bright peak at the bottom and one dimmer upper peak, as expected from simulation.

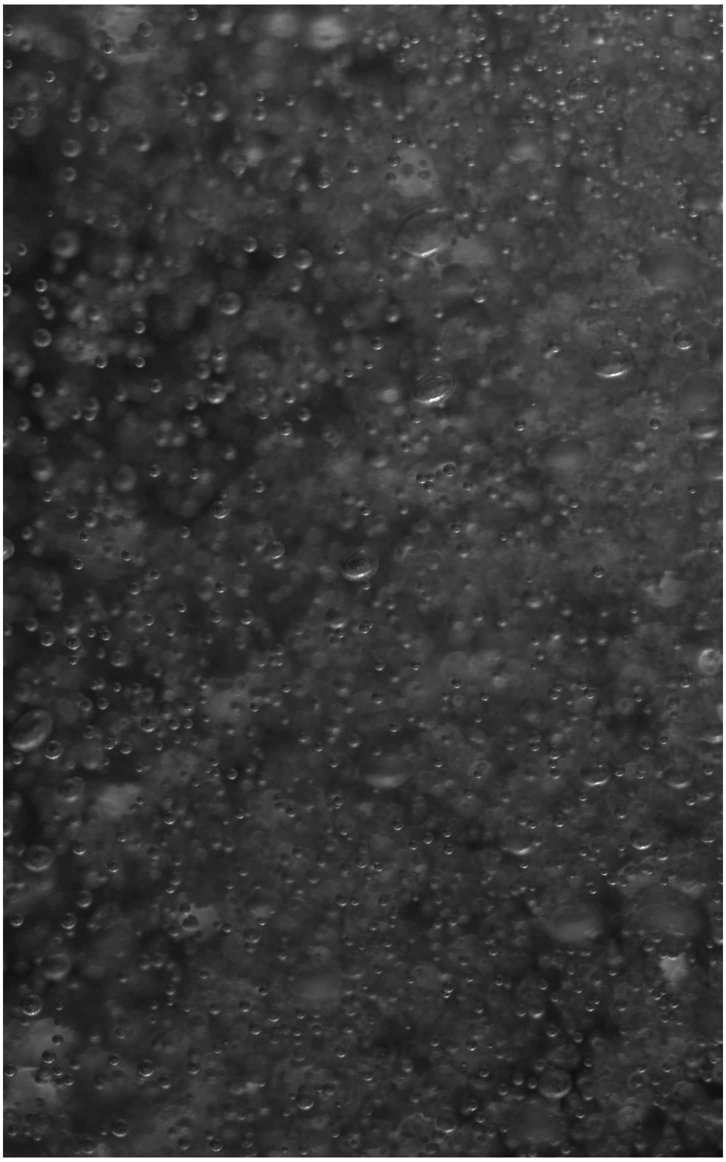


Figure 4.4: Sample image captured with aquarium setup for a high bubble concentration. The upper peak is not visible anymore. However, bubble curvatures are more recognizable, even for smaller bubbles

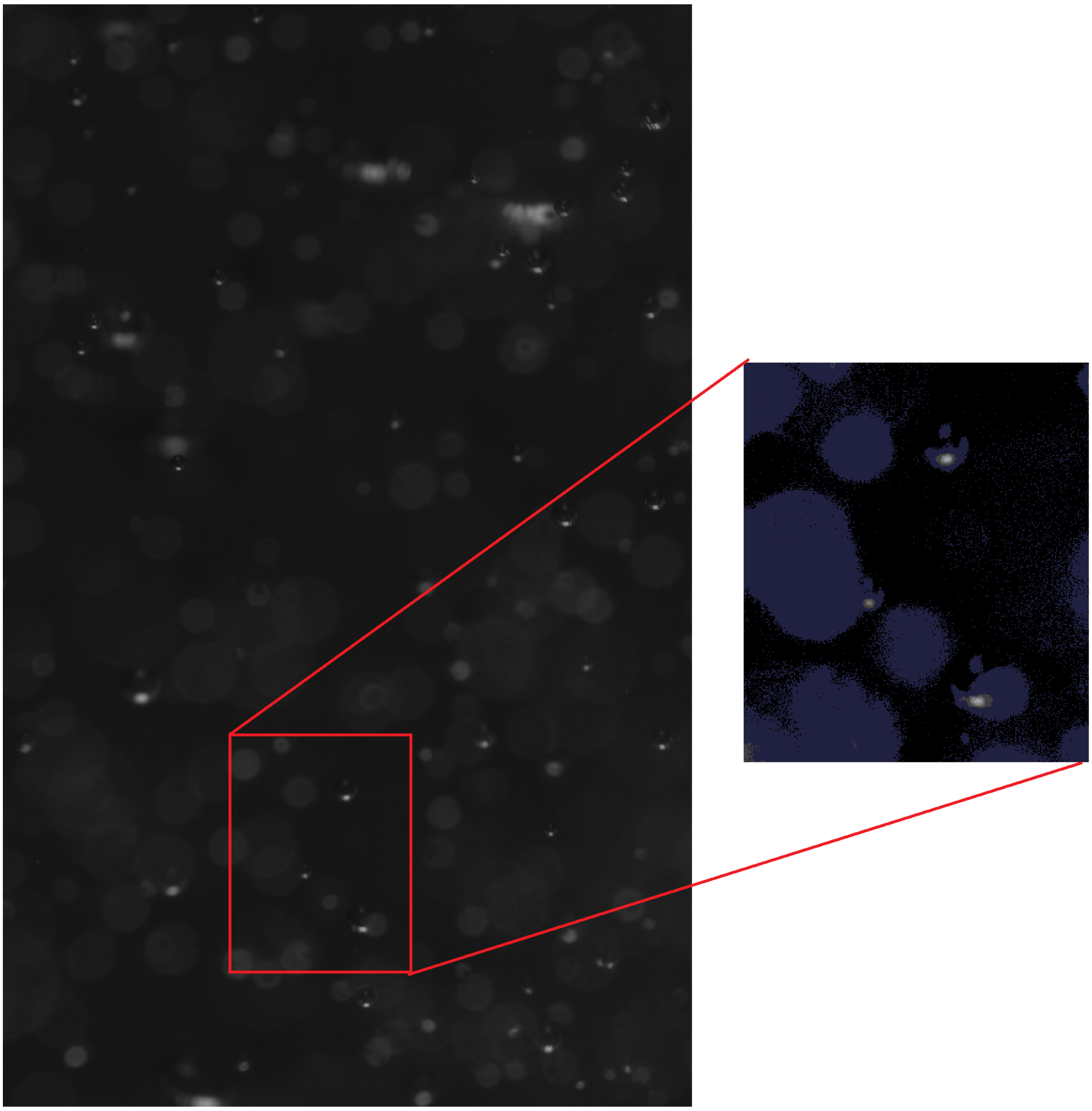


Figure 4.5: Image captured with Aeolotron setup. Bubbles were created by a bubble generator. Exposure time = , Gamma =

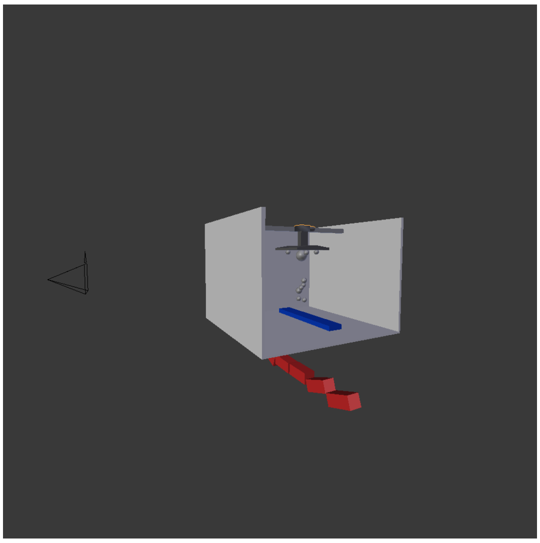


Figure 4.6: Setup for depth of field calibration. A high friction stick attached to a micrometer screw captures bubbles.

## 4.5 Calibration

### 4.5.1 Depth of Field

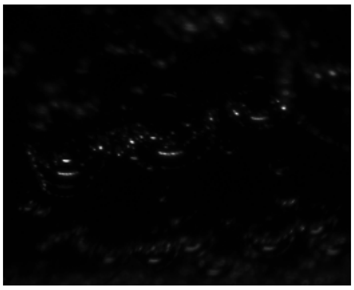
Since we want to determine the measurement volume, we need to not only detect bubbles in an image, but also determine their depth, i.e. distance to the focal plane. With the experimental setup shown in figure 4.6, bubbles can be trapped and moved along the depth axis.

Figure 4.7 shows images captured with this setup. Note how blurriness varies among images depending on the distance to focal point. In section 5.4 we discuss how to extract depth form these images. The setup is not perfectly stable, so resulting images were not perfectly aligned. As a workaround, we took additional images with a different light source at each position to make other surrounding objects visible, e.g. armature of calibration device that can later on be used for tracking and stabilization.

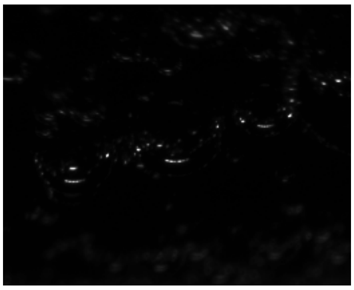
This calibration technique is practical, because we do not need to determine the absolute distance to the camera. So only relative distances to focal plane are needed for a depth calibration. This is important because the goal of calibration is to determine the measurement volume only and not necessarily the absolute position of the measurement volume to the camera.

### 4.5.2 Radius

Although simulation shows that the distance between the two dominant peaks in a bubble is proportional to the bubble radius, we cannot determine the real radius from measurement results alone (figure 4.3 and 4.5). So we need to change the



(a)



(b)

Figure 4.7: Two images from depth of field calibration setup at different distances from focal plane.

lighting in such a way that the real radius becomes measurable, i.e. the whole bubble becomes visible. In a similar way to the light field method (cite Mischler), we used an additional light source and a mirror in order to light up the bubbles from behind (figure 4.8). Then, we acquired images at a frame rate of 200 FPS and lit the additional light source every second frame. Results are shown in figure 4.9. In section 5.4 we identify bubbles lit from below with those lit from behind in order to estimate the proportionality factor between the real radius and the radius computed by our algorithm.

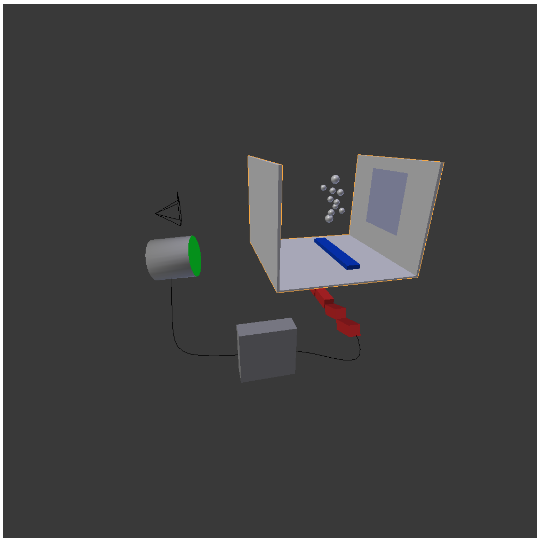


Figure 4.8: Setup for radius calibration at Aeolotron.

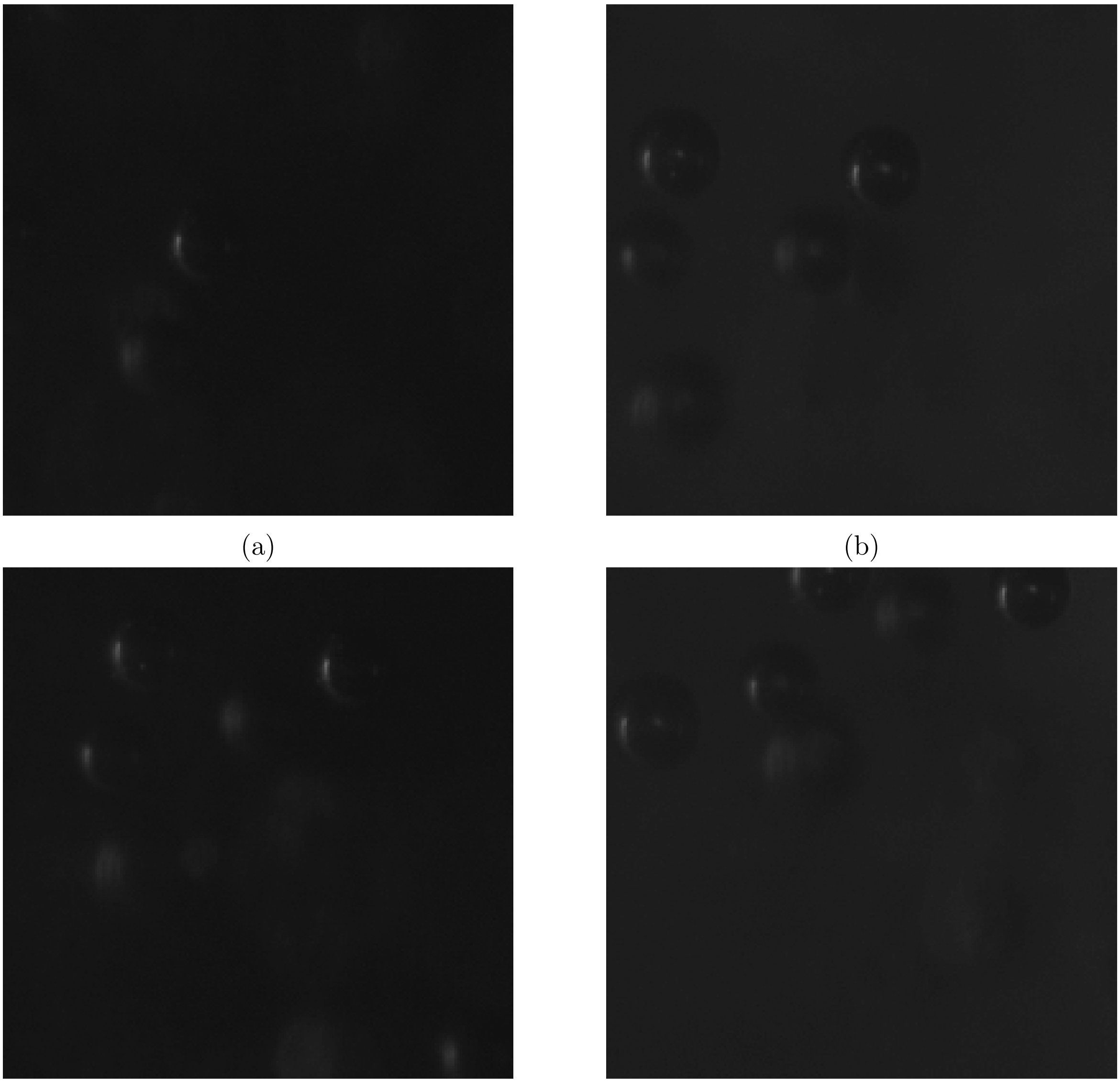



Figure 4.9: Radius calibration images

# 5 The Algorithm

In this chapter we discuss two important object detection algorithms that can classify bubbles in an image and estimate their radii. The first algorithm, which we refer to as *BubbleLow* or simply algorithm 1 is optimized to work for small bubble concentration (e.g. figure 4.3) whereas algorithm 2, also referred to as *BubbleCurves* is better suited for bubble images with very high concentrations (figure 4.4).

## 5.1 Motivation

We mentioned in section 2.3 that the state of the art object detection algorithms are based on deep learning algorithms such as Faster R-CNN (cite frcnn Leute) and YOLO (Cite yolo Leute). Training these algorithms with our data (simulated and real) did not yield good results. Training a pre-trained R-CNN algorithm (i.e. transfer learning) is not meaningful either because most pre-trained models have large anchors (or rank regions boxes) and are fine tuned to work for real world photographs that contain typical objects such as humans, cars and pets.

This led us to the conclusion that such algorithms are not well suited for detecting small and dense objects (images with a low bubble concentration are also considered to have dense objects, since they contain several hundred bubble instances per image that are very close to each other). This suspicion was confirmed by (cite Zhenhua Chen et Al.) who also added that these algorithms not only rely on a large amount of training data, but also require the data to be rich in features, which does not apply to our data.

Interestingly though, scaling down anchors and training an R-CNN algorithm (cite rcnn, Projektpraktikum?) from scratch did perform slightly better, with an IoU@0.5mAP of 68%. This however, is far from ideal, so developing a different approach was necessary to achieve better results.

## 5.2 BubbleNet

This algorithm detects bubbles in an image with low bubble concentration and estimates their radii. The algorithm extracts candidate signals from images, classifies these signals and then computes a radius from a signal. A vanilla version of the algorithm is described in algorithm 1. Note that the algorithm returns a list of rectangles (i.e. bounding boxes around bubbles) so that we can evaluate its performance. Later on, only bubble radii, i.e. rectangles' widths and depths are needed to construct a bubble spectrum.

**Algorithm 1** BubbleNet

**Input** Image with low bubble concentration $G$.
**Output** List of rectangles $Rec$, List of depths $Dep$
Obtain local maxima $loc\_max$ in $G$
**for** Each $lm$ in $loc\_max$ **do**
    Extract vertical profile signal $v_p$ with length $L$.
    **if** BubbleNet($v_p$) **then**
        Compute radius $r$ and center $c$ from signal using equation 5.1
        Compute real radius $r'$ using radius calibration equation 5.6
        Compute depth $d$ using equation 5.5
        append($Rec$, toBoundingBox($r'$, $c$))
        append($Dep$, $d$)
    **end if**
**end for**

### 5.2.1 Radius from Bubble

A closer inspection of bubbles shows that a bubble can be described well by a vertical cross section going through the middle of the bubble. For instance, figure 5.1 shows that a vertical profile $v_p$ is characterized by a two salient peaks: A bright peak that emerges from total reflection as explained in figure 2.2 as well as a dimmer, yet still visible second peak at a distance $d$ from the first peak. Furthermore, simulation shows that the distance $d$ is proportional to the real bubble radius (figure 5.2). Note that in figure 5.2 the distance $d$ was computed on rendered images, whereas the radius is a given parameter that is determined before rendering. The step wise increase in $d$ can be explained by sampling, i.e. when the radius increases by an amount smaller than a pixel, $d$ will either not change or increase by exactly 1 pixel.

In order to reach a better precision, we perform a Gaussian fit around the two peaks. Here it is necessary to initialize the fit routine with reasonable values, otherwise both fits would converge towards the same function. So we first smooth the signal with a 1D Gaussian filter, and then determine the absolute maximum $m_1$ within the signal corresponding to the first peak. The second peak corresponds to the highest local maximum $m_2$ that comes right after $m_1$. Finally, we use $arg(m_1)$ and $arg(m_2)$ as initial parameters for a Gaussian fit around the peaks. The final result is given by

$$r_{\text{measured}} = \frac{|\mu_1 - \mu_2|}{2} \tag{5.1}$$

for the radius and

$$c_{\text{measured}} = \left( \frac{|\mu_1 + \mu_2|}{2}, v_{p,y} \right) \tag{5.2}$$

for the bubble center where $\mu_1$ and $\mu_2$ are the resulting parameters from the Gaussian fit that correspond the first and second peak respectively.

From previous considerations it becomes clear that a radius calibration is needed to determine the real radius from the measured radius. This is discussed in sections 4.5.2 and 5.4.2.

### 5.2.2 Signal extraction

So far, we discussed how we can determine the radius from a given vertical cross section of a bubble. However, the question remains, how do we know the signal corresponds to a bubble in the first place? This is by far more challenging than determining the bubble radius and will be discussed thoroughly in the following sections.

As we can see from measurement results (section **??**), extracting local maxima from an image gives good starting points for vertical profile extraction. Starting from a local maximum, we need to go a few pixels downwards $l_d$ to cover the whole peak and a certain distance upwards $l_u$ until the second peak is reached. $l_d$ and $l_u$ are hyperparameters that add up to the total signal length $L$. The choice $l_u$ depends on the largest bubble radius that we want to detect. From (cite bubble instrument system) we know that bubbles large enough to loose their spherical shape are very rare and their contribution to gas transfer can be neglected. Also, (cite Leonie) showed that the maximum bubble diameter observed at the Aeolotron facility lies at around 500 $\mu m$ (i.e. 50 pixels). As for $l_d$, a constant value of 4 pixels is good enough to cover the large peak, because the value returned by the fit usually lies within 1 to 2 pixels away from the local maximum.

### 5.2.3 Signal classification

After having extracted a one dimensional signal from a potential bubble, we classify it using a convolutional neural network that we call *BubbleNet*.

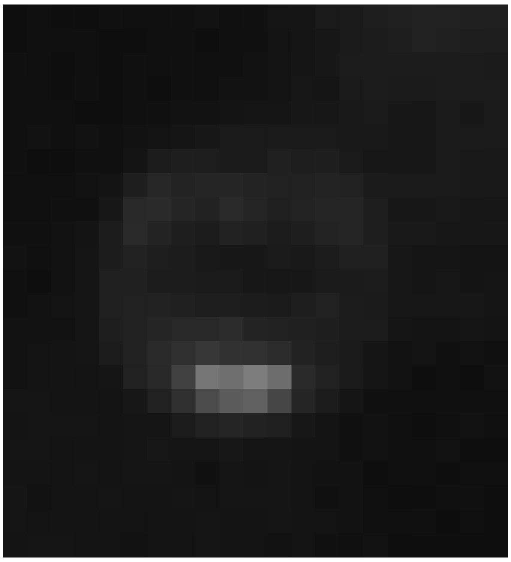
#### BubbleNet architecture

The architecture of our BubbleNet is inspired from (cite Dan Li et al.), where the authors used a 1D convolutional neural network to classify electrocardiogram signals (ECG). The architecture of our CNN is shown in figure 5.3. We used two convolutional layers with MaxPool layers in between and two dense layers, where the last dense layer is also the output.

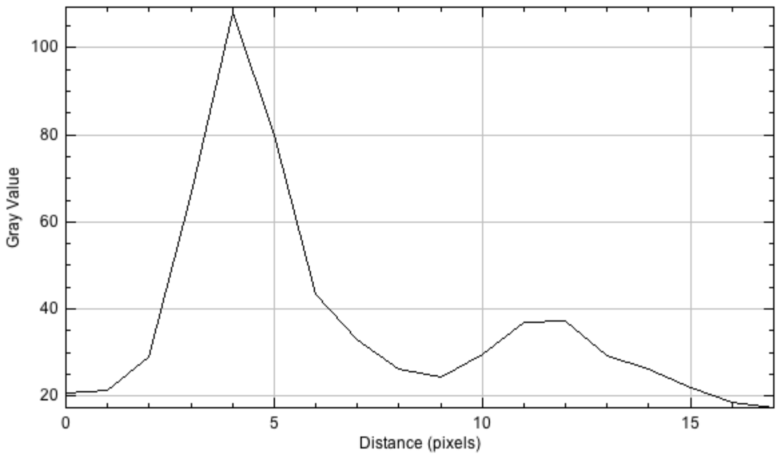
Note that the neural network has a fixed input size of 20 gray values. This means that resizing the vertical profiles is often necessary before passing them to the CNN. We therefore train our CNN with resized inputs in order to make it robust against such preprocessing operations.

Since the bubble radius is not known prior to vertical profile extraction, we also extract $n$ signals, where

$$r_{min} \leq n \leq L \tag{5.3}$$



(a) Enlarged image of a bubble from a low concentration image.




(b) Vertical bubble profile. The second peak is much dimmer than the first, but still bright enough to be distinguished from background noise.

Figure 5.1: A small air bubble from an image with low bubble concentration.

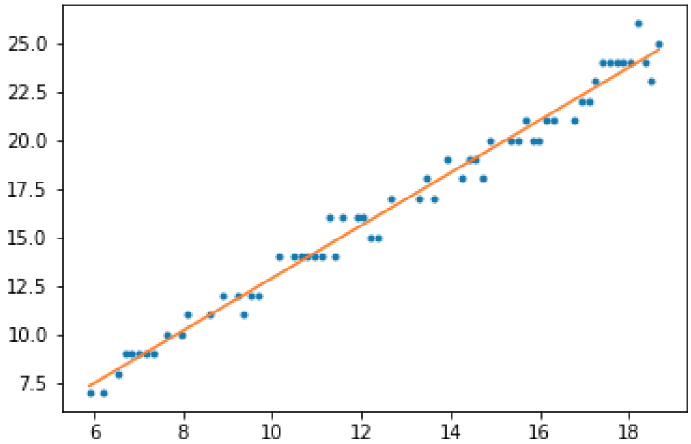



Figure 5.2: Simulated distance between two peaks as a function of bubble radius.

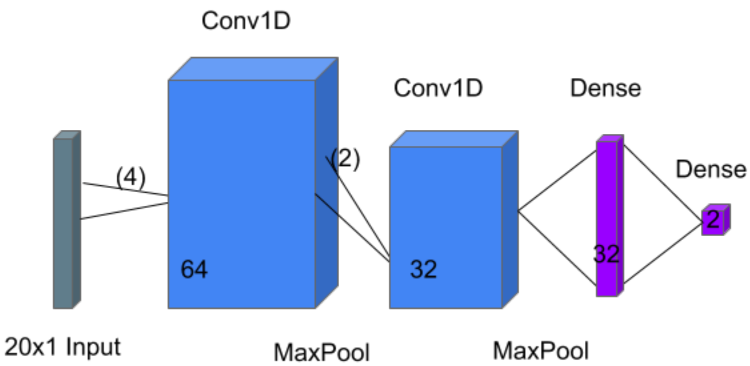



Figure 5.3: Architecture of BubbleNet. Numbers in parentheses refer to the kernel size. The rest refers to the layer size.

and then keep the signal with the highest bubble probability. Signals that are classified as background obviously do not get further considerations. $r_{min}$ is the smallest detectable bubble radius and it depends on the magnification factor. For the Aeolotron setup, we have $r_{min} = 90\mu m$.

### Training Data

The most straightforward approach to generate training data is to annotate images by hand, preferably with an accuracy of no more than 2 pixels. However, this can quickly become very tedious work, especially when hundreds of thousands of images are needed to train a neural network with over $10^3$ parameters. Instead, we generated training data as follows:

1. Annotate 300 bubbles only by hand.
2. Train logistic regression classifier with 300 annotated and 300 simulated images using manually computed features.
3. Predict 100 000 bubble instances with logistic regression classifier
4. Use data augmentation to generate a total of 200 000 vertical profiles.
5. Extract 200 000 background (non bubble) images to balance data.

For step 2, we used following features:

- First peak's gray value $p_1$
- Second peak's gray value $p_2$
- $p_1/p_2$
- Peaks distance $d$

The annotated images are fairly similar in size, lighting conditions and camera settings (Gamma value, gain, exposure time etc..). The reason for that, is because we want to optimize the logistic regression classifier for a high recall, regardless of its precision. So at this stage, we only want to detect as few false positives as possible, even if that means neglecting potential true positives, which in turn lowers the precision. This is obviously not a good approach to classify bubbles, however, our goal at this stage is to merely generate accurately annotated data. After predicting $10^5$ true positive bubble instances from our measurements in step step 3, we processed these images further to generate more data. This included changing the gamma value, resizing (upsampling and downsampling), and vertical flipping.

Having generated more than $4 \cdot 10^5$, the neural network can now be trained.

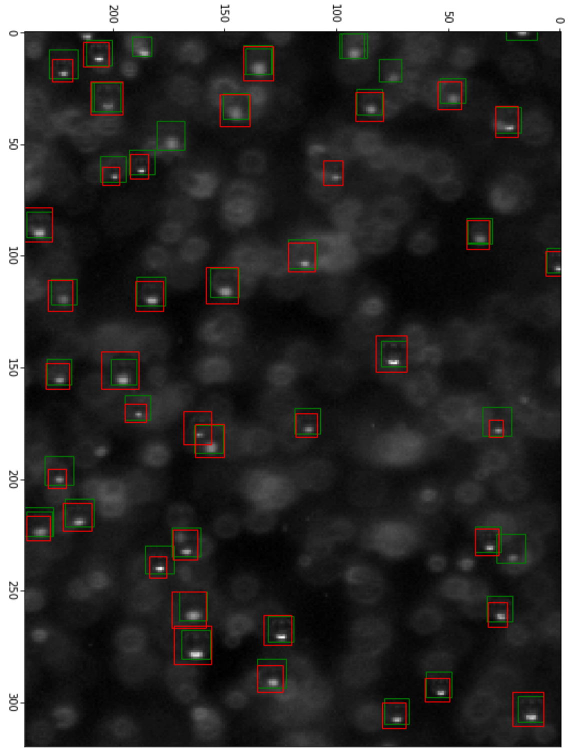


Figure 5.4: Testing of algorithm 1. Red: ground truth, green: predicted. Note how in this image, ground truth annotation was not done perfectly, so IoU results were likely slightly underestimated.

### Results

We first shuffled our data and split it into training (80%) and testing (20%). After training for 14 epochs, the testing accuracy converged against 96.8%, which is a very good result for a binary classifier.

For validation we used 20 more manually annotated small images, each containing from 10 to 20 bubbles. Figure 5.4 illustrates the performance of algorithm 1. The algorithm achieved IoU@0.5mAP = 91.5% and IoU@0.3mAP = 93%. Note that this is slightly worse than the classification accuracy result. This is due to the occasionally unprecise radius computation, which gets penalized by the IoU@p-mAP criteria.

## 5.3 BubbleCurves

This algorithm detects bubbles in images with high bubble concentrations and estimates their radii. In these images, bubble curvatures are much better distinguishable than on low bubble concentrations images. In algorithm 2, we first pick candidates, classify them and then determine their radii.

**Algorithm 2** BubbleCurves

**Input** Image with a high bubble concentration $G$.
**Output** List of rectangles $Rec$, List of depths $Dep$
Obtain local maxima $loc\_max$ in $G$
**for** Each $lm$ in $loc\_max$ **do**
    Extract image window $W$ around $lm$ with size $10 \times 10$.
    **if** RFC($W$) **then**
        get bubble curvature $cur$
        Compute radius $r$ and center $c$ from $cur$ using a circle fit.
        Compute real radius $r'$ using equation 5.1
        Compute depth $d$ using equation 5.5
        append($Rec$, toBoundingBox($r'$, $c$))
        append($Dep$, $d$)
    **end if**
**end for**

### 5.3.1 Bubble Classification

In a similar way to algorithm 1, we pick our candidate bubbles based on local maxima in the input image. Then we extract a window with fixed size of $10 \times 10$ pixels around the local maximum. Orientation from structure tensor, the eigenvalues of the structure tensor (see section 5.3.2) as well as the eigenvalues of the Hessian matrix are used as features to classify bubbles using a random forest classifier.

The choice of these features is justified as follows. We expect bubbles to show a strong orientation inwards around the local maximum. Therefore we compute the orientation using the structure tensor and expect orientation to be vertical. The eigenvalues of the structure tensor show whether this orientation is significant or not. Finally, with the eigenvalues of the hessian matrix we can determine the curvature around the local maximum, which is expected to be concave.

As for classification, we chose a random forest classifier (section **??**) because it is well suited for determining thresholds by using decision trees that are particularly robust to inclusion of irrelevant features, which is important when the window size is larger than the considered bubble.

### 5.3.2 Radius from Orientation

The idea behind determining the bubble radius is to start from a point on the bubble edge and then "walk" along this edge and keep track of the path. After gathering enough points along this path, we perform a circle fit to determine the bubble's radius and center.

#### Orientation

Figure 5.5 shows the result of applying the structure tensor to a single bubble. We note that orientation arrows describe the orientation as expected around the strongest peak in the image. Since it is only possible to compute the double angle using the structure tensor, the direction of the arrows is not important. In fact, we see a direction flip around the middle of the image. When considering the eigenvalues, we note that both eigenvalues are close to zero around the bubble center and on the upper part of the bubble (figure 5.5a and 5.5b). This means that orientation around this area is homogeneous (ref theory struct tensor EV interpretation) so results form these area are likely not valid. This is more apparent in figure 5.6, where orientation on the upper side is different from the one around the brightest peak in the image. For these reasons, we will only use orientation along the bubble edge up to the middle of the bubble, i.e. where the larger eigenvalue is significantly larger than zero.

#### Curve from Orientation

Assuming that our starting point is part of the bubble's lower edge, we first get the orientation at the current pixel and then sample points on a line segment along this orientation. The sampling method is taken from (cite max bopp): A square is drawn around each point on the sampling line, and then a weighted average of the gray values within the rectangle is performed, where weights are intersection areas between neighboring pixels and the drawn rectangle. This sampling step is illustrated in figure 5.8 After sampling, we perform a Gaussian fit along the fit line, where we initialize the mean with the current position. Next, we add the fit result to our curve and move one step to the right. Figure 5.7 summarizes how a bubble curve is extracted using orientation.

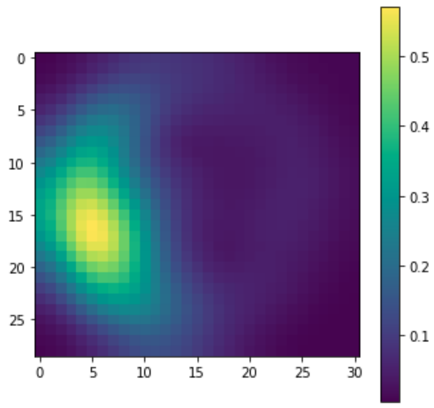
Finally, we use a least squares fit to find the center $c = (a, b)^T$ and the radius $r$ from the circle equation

$$\sqrt{((x-a)^2 + (y-b)^2)} = r^2 \tag{5.4}$$

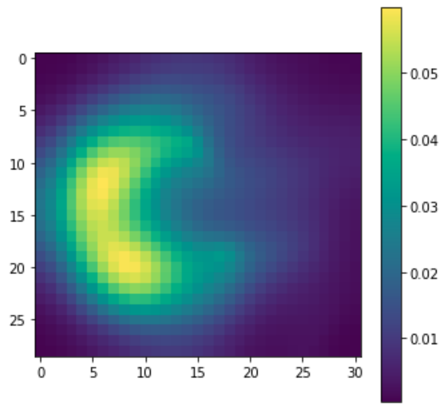
The fit and sampling results are shown in figure 5.9

#### Results

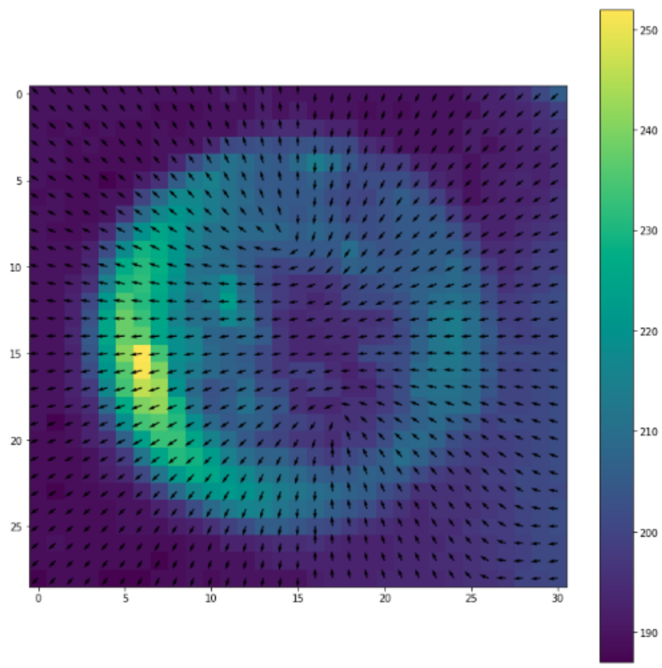
The final result is shown in figure 5.10. Although the circle matches the bubble's shape well, it is clear that the radius has been underestimated. Similarly to algo-




(a) Larger eigenvalues of the structure tensor




(b) Smaller eigenvalues of the structure tensor




(c) Orientation from structure tensor

Figure 5.5: Structure tensor eigenvalues and orientation on a single bubble from a high bubble concentration image.

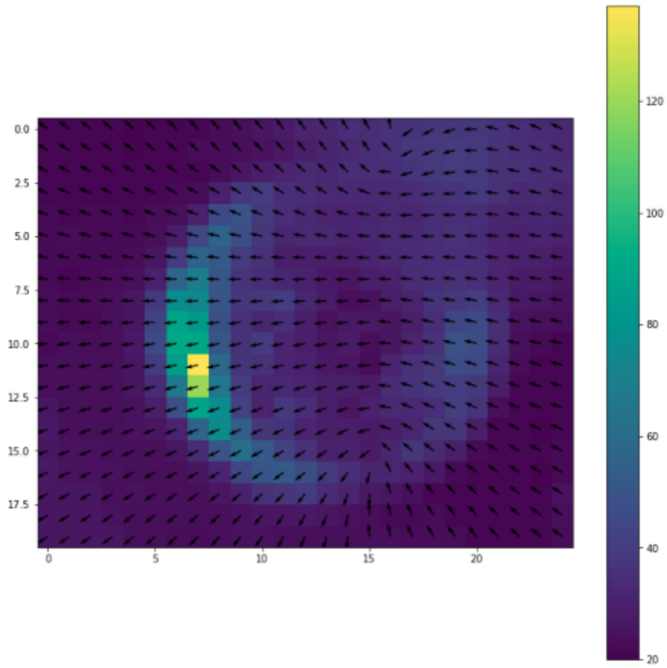



Figure 5.6: Applying the structure tensor on a slightly smaller and differently lit bubble. Orientation from the upper part is not meaningful.

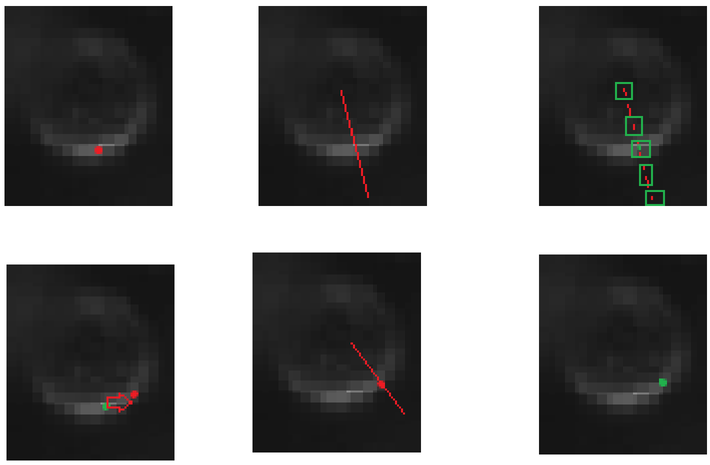


Figure 5.7: Extracting curve from bubble using orientation and neighborhood sampling.

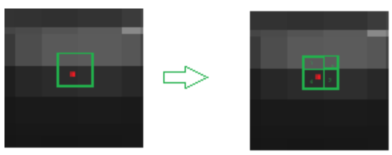
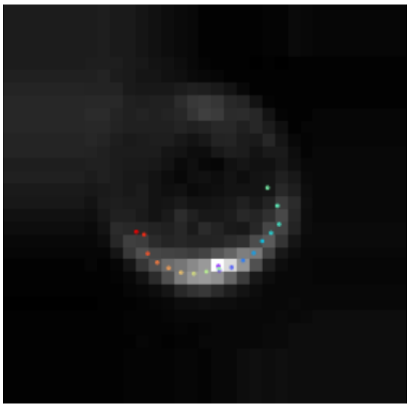
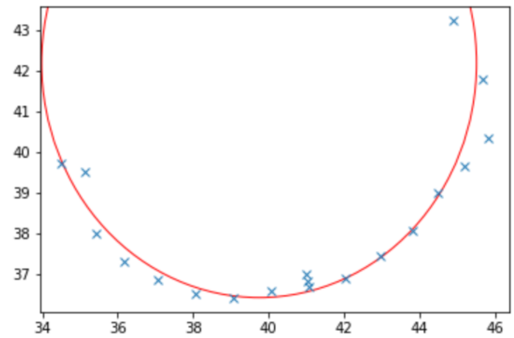


Figure 5.8: Gray value sampling along sampling line



(a) Sampling result.




(b) Circle fit result.

Figure 5.9: Sampling and fit Note that the circle works well, even though only half a circle is extracted from the bubble at most.

Figure 5.10: Final result of algorithm 2 applied to a single bubble

rithm 1, radius calibration is addressed in section 5.4.2

Using a validation set of 30 small images, each containing 30 to 40 bubbles, this algorithm achieved IoU@0.5mAP = 89.3%, which is slightly worse than algorithm 1. This is in part due to the high rate of overlap between bubbles, making curvature extraction less precise.. Nevertheless, the algorithm is reliable enough to be usable for bubble detection in high concentration images.

## 5.4 Calibration

### 5.4.1 Depth of Field

Determining the distance between a bubble and the focal plane can be deduced from bubble blurriness. If we consider a vertical profile of the bright lower peak (for both low and high bubble concentration images), we note that the peak's width increases with increasing distance from the focal plane (figure 5.11a). We therefore describe blurriness with the peak's width, given by the sigma parameter of a Gaussian fit around the peak. Theoretically, we would expect a linear dependency between blurriness and distance to focal plane. However, our light source has a certain size that has to be accounted for, which gives bubble peaks a certain minimum width at focus.

Furthermore, blurriness depends on the radius. For instance, larger bubbles tend to be visible for larger distances from the focal plane (up to 10 centimeters), whereas small bubbles are only visible for no more than 2 centimeters. Figure 5.11b shows measured blurriness (in terms of $\sigma$) as a function of radius and distance from focal plane.

Therefore, we choose a fourth degree polynomial to describe the blurriness-depth dependency and a linear function to describe the radius dependency. The resulting calibration equation is

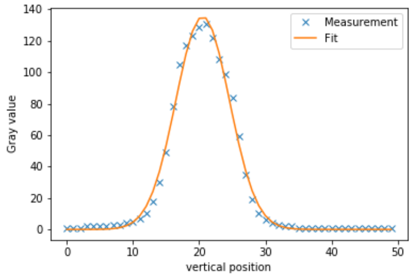
$$f(r, d) = b \cdot r + a_1 \cdot d + a_2 \cdot d^2 + c \tag{5.5}$$

Where $a_1 =, a_2 =, b =, c =$ were used for Aeolotron setup. Figure 5.11c shows the two dimensional calibration function alongside the measurements.

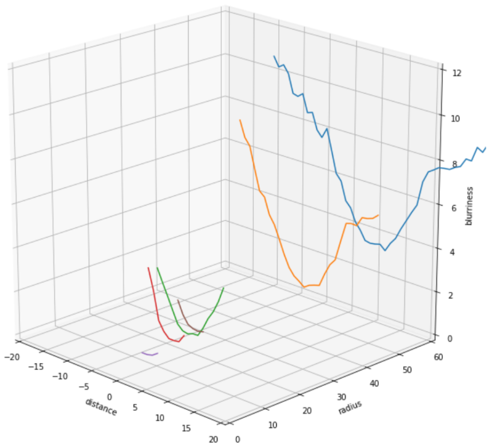
Note that only measurement results up to a certain threshold $\sigma_{th}$ were kept and used for the fit function 5.5.

### 5.4.2 Radius

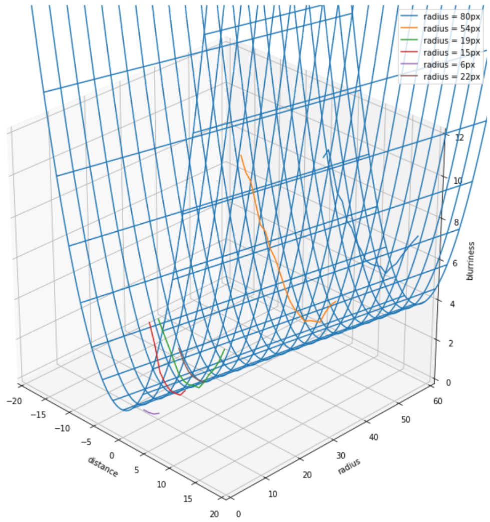
The measurement technique developed in this work uses a light source to light up bubbles from below and captures images with a camera placed at 90° to the light source. The resulting images are characterized by a lower peak (or bright lower curvature for high concentration bubbles) as discussed in section **??**. From these images alone, it is not clear whether the observed edge corresponds to the real bubble




(a) Vertical peak profile and Gaussian fit with $\sigma = 4.04$




(b)




(c) Sampling result.

Figure 5.11: Depth calibration

edge or not. Also, algorithm 2 clearly underestimates the observed radius (figure 5.10).

In order to correct for this effect, we performed calibration measurements as described in section 4.5.2.

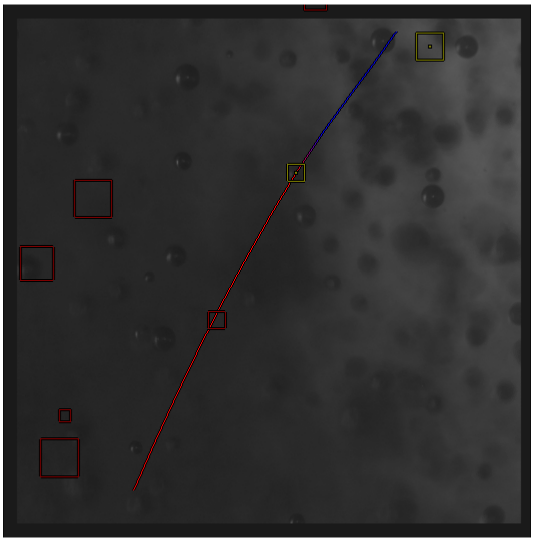
We first need to identify back lit bubbles with those lit from below. So we take images with back lit bubbles only, and track a few bubbles of different sizes and depths and compute their radii. From the tracked back lit bubbles, we can roughly estimate the position of the same bubble lit from below. Figure 5.12 illustrates the principle behind identifying bubbles from different images.

Determining the radius of bubbles lit from below is done using algorithm 1. As for the corresponding back lit bubbles, the radius is determined as follows: First, we compute the derivative image and then estimate the position of the bubble center using the characteristic bright peak around the bubble center. We then extract several horizontal profiles that go through or are very close to the bubble center. Next, we compute the local maxima from the extracted profiles and apply a Gaussian fit around them in order to determine the local maximum with a subpixel accuracy. The bubble diameter is then given by the distance between the two local maxima that are farthest apart from each other. Figure 5.13 illustrates the described steps above.

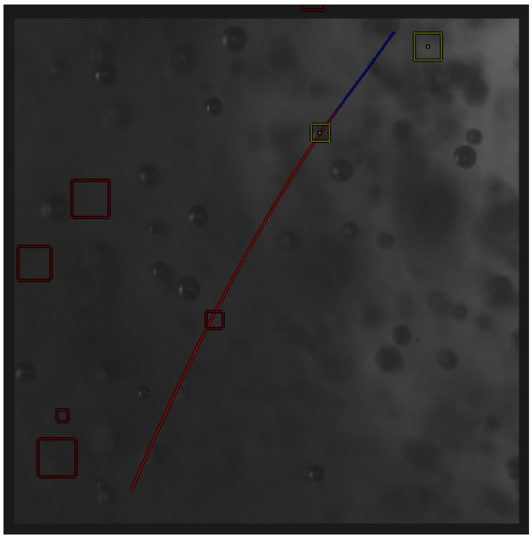
Figure 5.14 shows the radius determined by algorithm 1 of bubbles lit from below as a function of the radius of back lit bubbles. The error bars are standard deviations, i.e. statistical errors arising from multiple radius computations of the same bubble at different depths and different locations. The resulting radius calibration equation is

$$r' = a \cdot r + b \tag{5.6}$$

Where $a = 0.95$ and $b = 4.54$ were used for the Aeolotron setup.



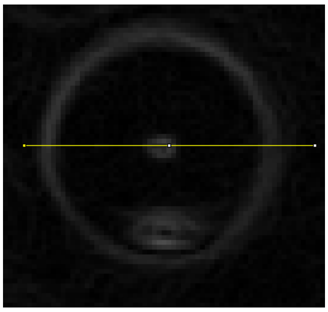
(a)



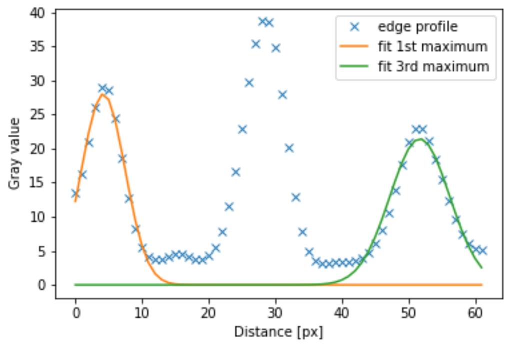
(b)

(c)

Figure 5.12: Identifying a bubble from images with different lighting. Images were acquired at 200 FPS. Image 5.12c was taken after 5.12a and before 5.12b. Matching bubble in 5.12c is marked in red



(a) From derivative image we extract horizontal profiles




(b) The distance between the farthest local maxima gives the diameter

Figure 5.13: Determining the radius of a back lit bubble

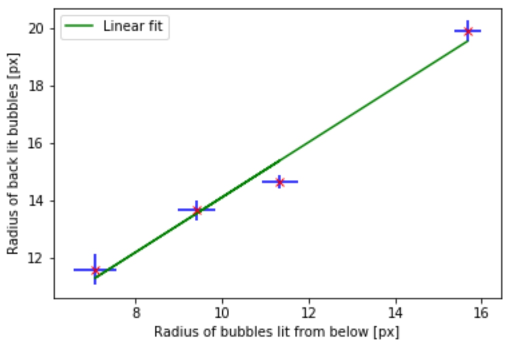



Figure 5.14: Measured radius as a function of real radius.

# 6 Summary and Discussion

# Appendix

# A Lists

## A.1 List of Figures

## A.2 List of Tables

Erklärung:

Ich versichere, dass ich diese Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den 30.09.2018 .......................................