

MATLAB® APP

Bubble Analyser

Instruction manual

By Francisco Reyes, Paulina Quintanilla and Diego Mesa

Bubble Analyser

Instruction manual

Bubble Analyser is a MATLAB® Application for processing and quantifying bubble images, normally taken for assessing froth flotation equipment.

This App has been developed by [Francisco Reyes](#), [Paulina Quintanilla](#) and [Diego Mesa](#).

For more information please visit our [GitLab repository](#).

The copyright of this document and Bubble Analyser rests with the authors and is made available under a GNU General Public License. Researchers are free to copy, distribute or transmit the document on the condition that they attribute it



Table of contents

Chapter 1 Installation	1
Using Bubble Analyser With Matlab®	1
Without Matlab®	1
Chapter 2 Usage.....	2
Opening The App.....	2
Using The App	2
Chapter 3 Using the default algorithm.....	10
Chapter 4 Using your own algorithms.....	12
App Interface	12
The Algorithm	12
The Configuration File	13
About the authors	14

This page has been intentionally left black

Chapter 1

Installation

Using Bubble Analyser with MATLAB®

You can simply download a copy of the latest version from the [GitLab repository](#) as a .zip (or equivalent) file. Unzip the file and keep the unpacked folder wherever you prefer—we recommend to put it in the default MATLAB® folder.

Once you have decided where to keep it, make sure you add this location to your MATLAB® path. For help on doing this, please refer to this [link](#).

Without MATLAB®

For Windows

1. Download “BubbleAnalyserApp.exe” from the [GitLab repository](#).
2. Double click on the file. This should open a virtual window run by MATLAB®.
3. Click “Next”
4. Choose installation folder, and then click on “Next”
5. As MATLAB Runtime is required, this program will be also installed on your computer.
6. Accept the license agreement, and then click on “Next”.
7. Click “Install”.
8. Wait until the installation is completed successfully.

For MacOS

1. Download “BubbleAnalyserApp.app” from the [GitLab repository](#).
2. Double click on the file. The app will be automatically saved in your “Applications” folder.
3. Use the app as indicated below.

Chapter 2 Usage

Opening the app

If you have MATLAB® installed in your computer and you have added the ‘Bubble Analyser’ folder to your Matlab path, you can run the APP using the following command in the command window

```
>> BubbleAnalyserApp()
```

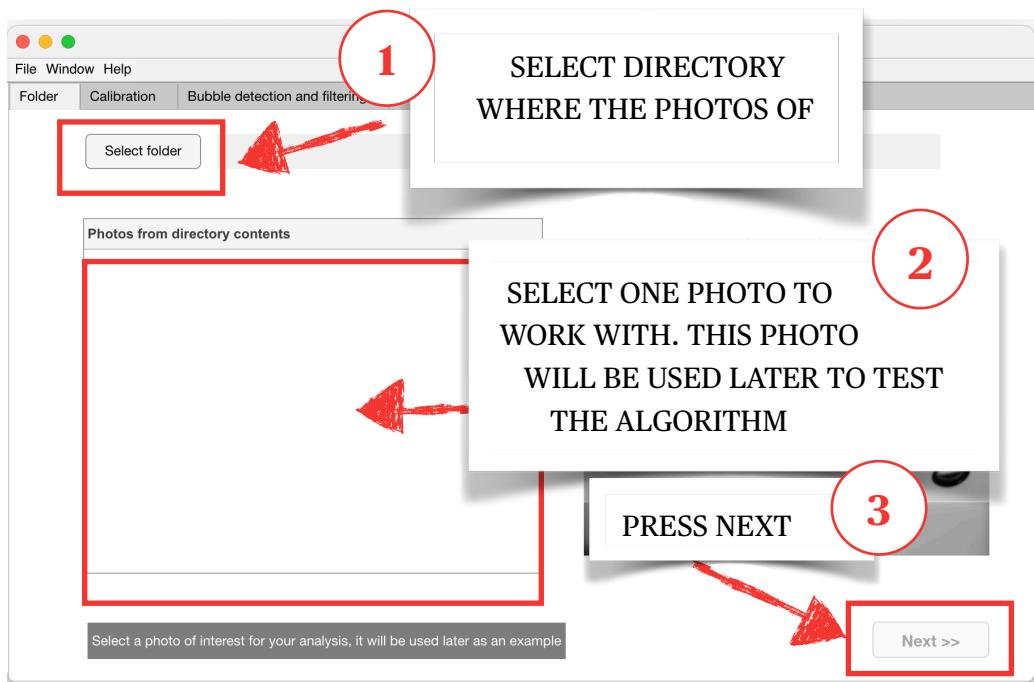
If you do not have Matlab installed in your computer, run the app by double clicking on the app icon.

Using the app

Bubble Analyser is divided in Tabs that guide the user through the process of selecting the images to process, calibration step, a step to batch process images and a final step for visualising the results. This section summarises each of these steps.

Tab 1: Folder

The initial Tab allows the user to perform mainly two actions. The first one is to browse and select the folder where the user has stored all the images that need to



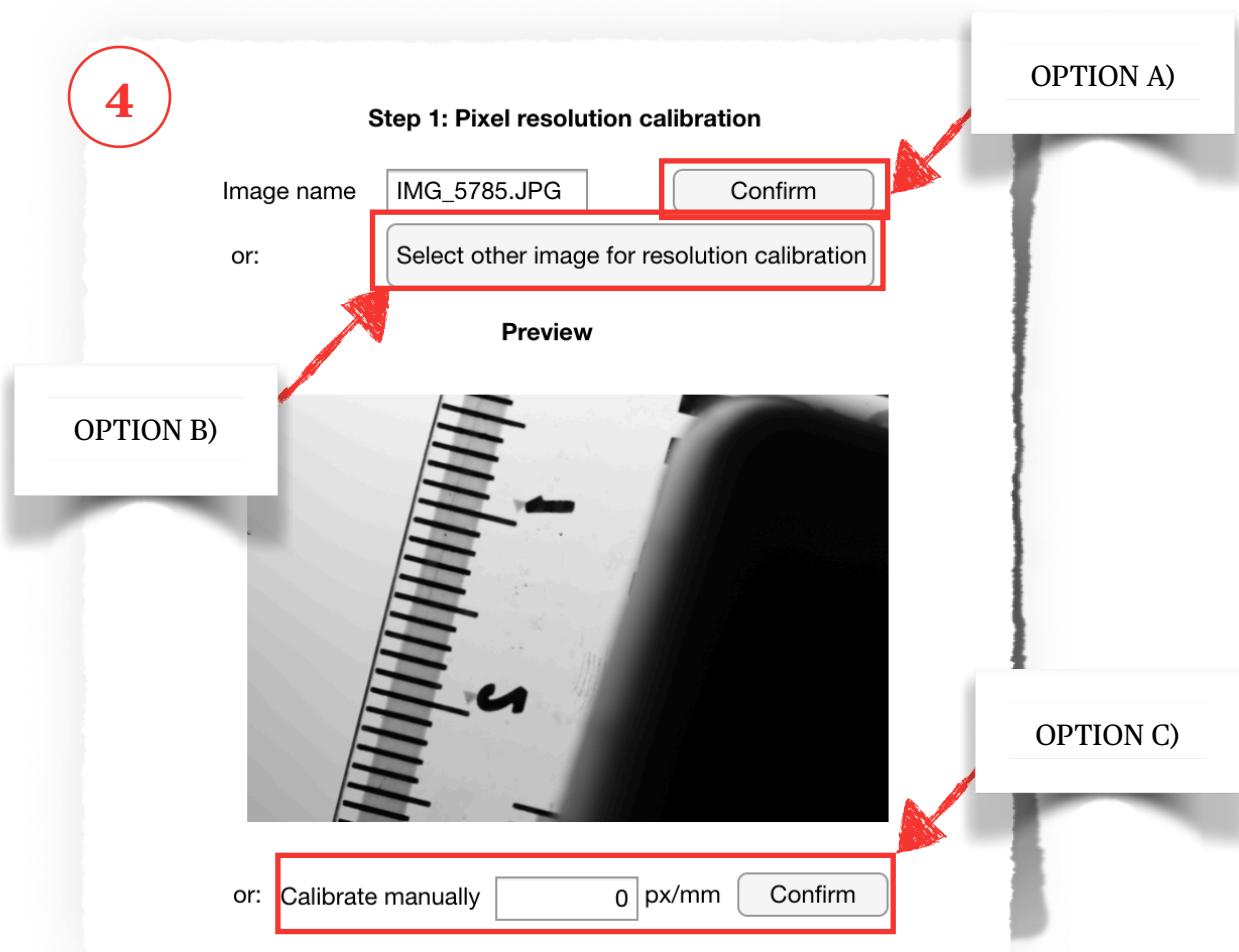
be processed. Bubble Analyser will create a subfolder (/working_files) where all the results and exported files will be saved

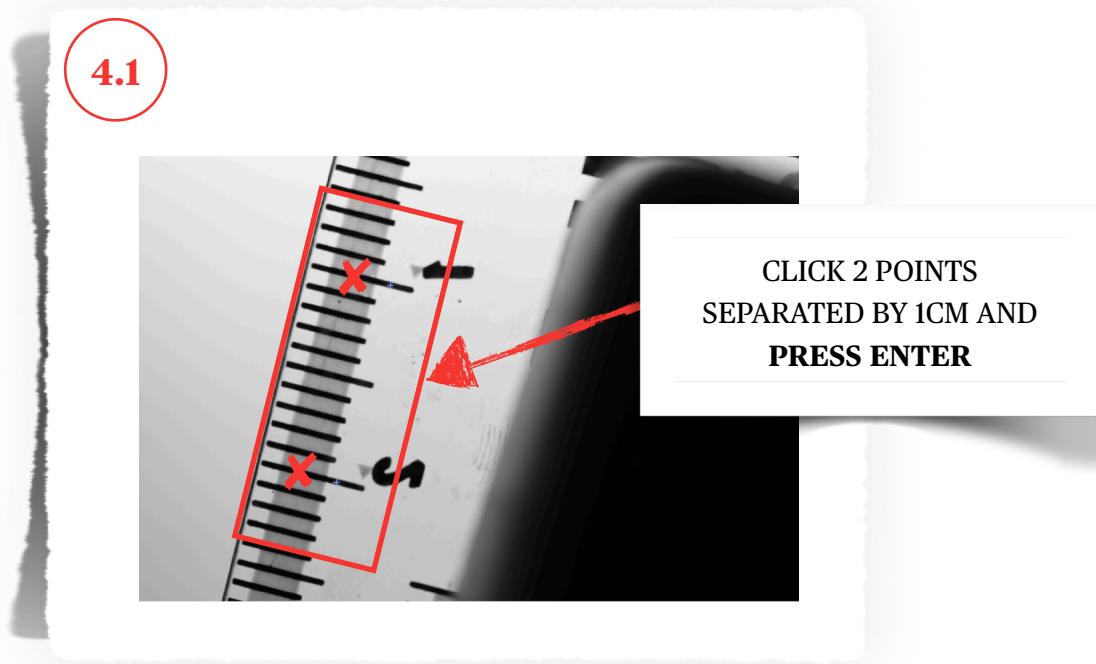
Once the folder is selected, Bubble Analyser will allow the user to perform the additional action. The user will be able to navigate the folder and preview the different images, to double-check the selected folder. Action that is performed by clicking on the “Next” button.

Tab 2: Calibration

An essential step for quantifying bubble size is to define the pixel resolution, this is how many *digital* pixels translate into a *physical* millimetre. There are three options for **pixel resolution calibration**:

- A) Confirm the image that was selected by default from your working directory (Tab 1 Folder).
- B) Select other image for pixel resolution calibration.
- C) Enter the calibration number manually (e.g. 183 px/mm).





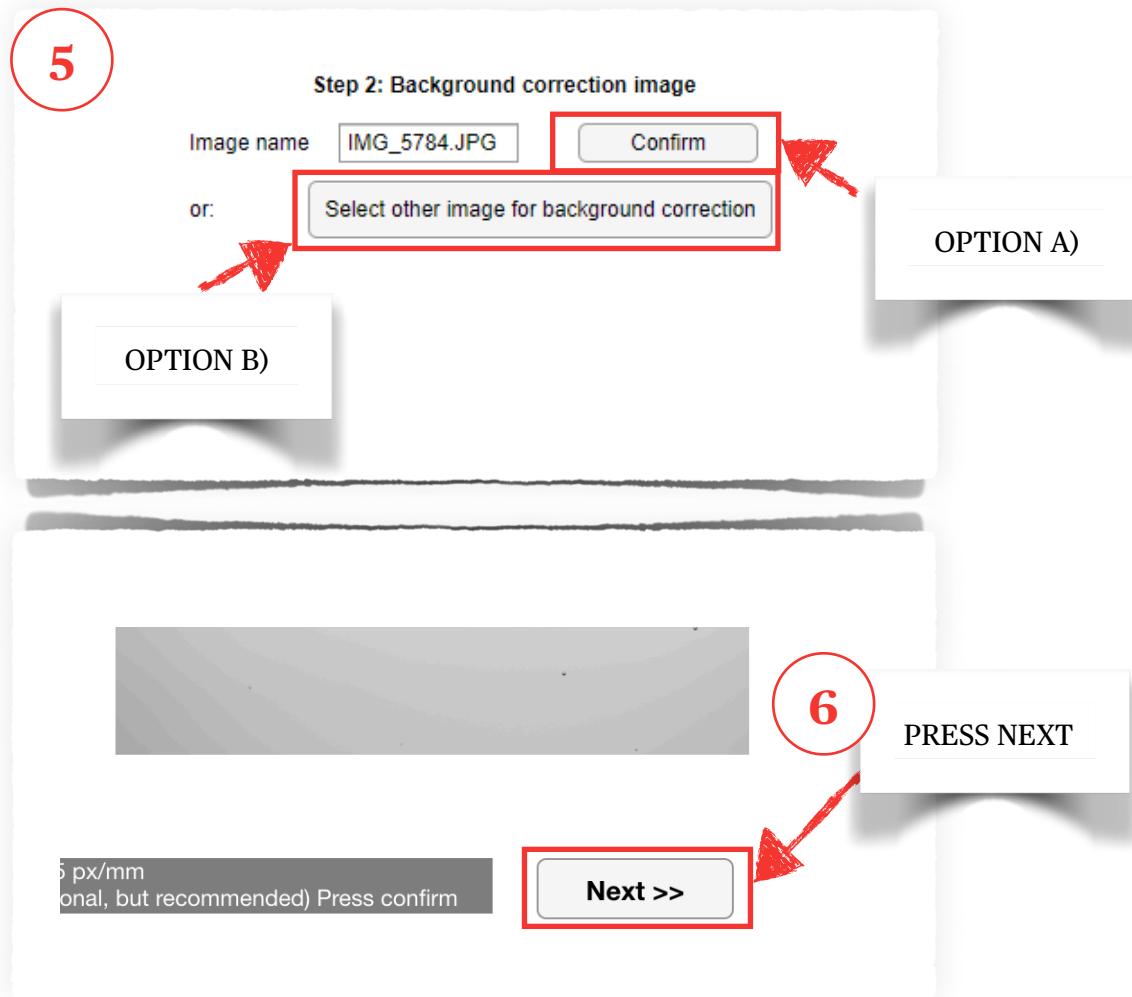
For options A) and B) a new window will be open to determine two points separated by 1 cm.

Optional, but recommended, there is also the possibility to define a background correction image. This refers to an image with no bubbles, so the background illumination can be captured and corrected for.

Select a **background correction image** and press confirm. There are two options:

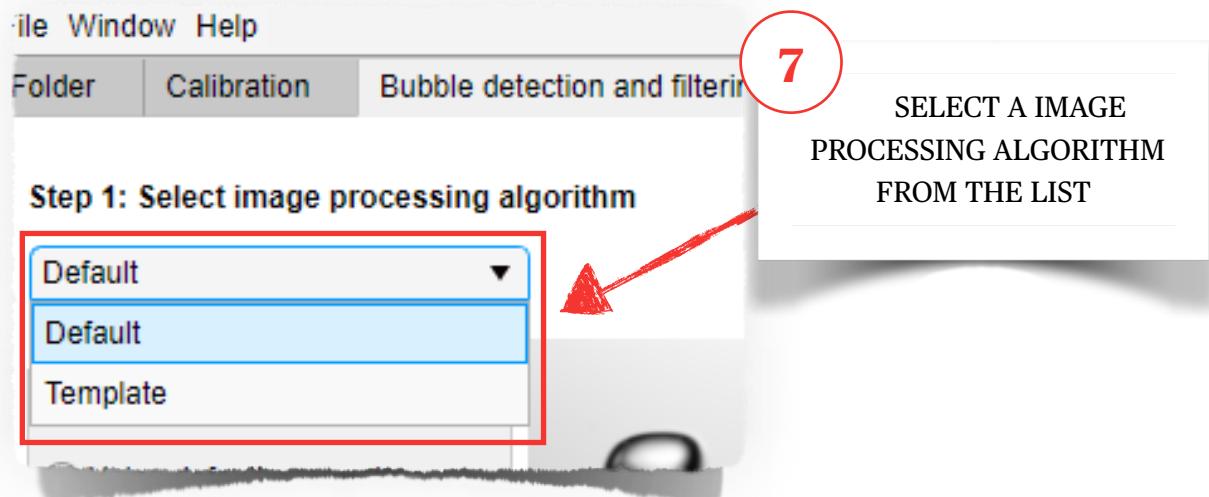
- A) Confirm the image that was selected by default from your working directory (Tab 1 Folder).
- B) Select other image for background calibration.

As mentioned before, selecting a background correction is optional, so the user is free to skip this step by just pressing the “Next” button once this becomes available after confirming a pixel resolution.



Tab 3: Bubble detection and filtering

Bubble Analyser comes with a default image processing algorithm (*BV_quantification*), but it can also incorporate others. Please refer to Chapter 3 for a detailed description on how to set up your own algorithm in Bubble Analyser.



The image processing algorithm has parameters that can be either:

- A) The default values, or
- B) your own chosen values.

8

OPTION A)

Step 2:
Image processing sandbox

Using default parameters
 Choosing my own parameters

Confirm parameters

OPTION B)

Step 2:
Image processing sandbox

Using default parameters
 Choosing my own parameters

Parameter	Value
Morphological_...	5.0000
Connectivity	8.0000
Marker_size	10.0000
resample	0.5000
Max_Eccentricity	0.8500
Min_Solidity	0.9000
min_size	0.1000

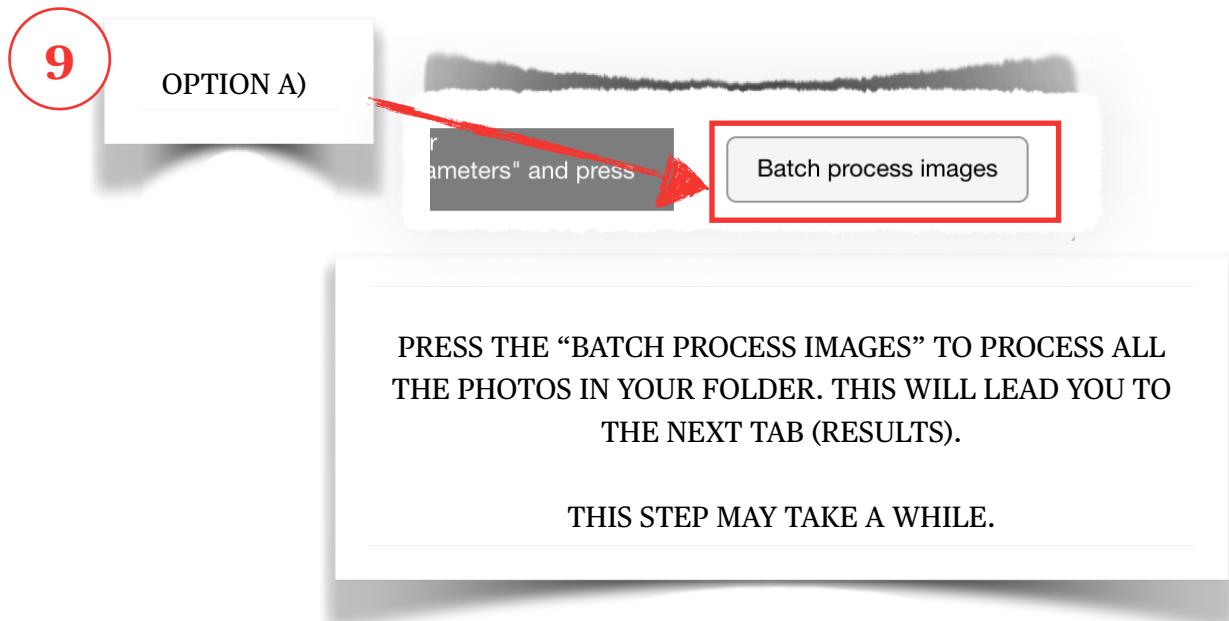
<< Back

TO TRY THE SELECTED PARAMETERS,
PRESS CONFIRM TO PROCESS THE

MODIFY PARAMETERS AND PRESS
“CONFIRM PARAMETERS” BUTTON

After processing the sample image, you can either:

- A) Press “Batch process images” button to process all the images in the folder
- B) Modify the parameters and try them in either the same or another sample image.



Step 2:

Image processing sandbox

Using default parameters
 Choosing my own parameters

Try new parameters

Parameter	Value
Morphological_...	5.0000
Connectivity	8.0000
Marker_size	10.0000
resample	0.5000
Max_Eccentricity	0.8500
Min_Solidity	0.9000
min_size	0.1000

OPTION B)

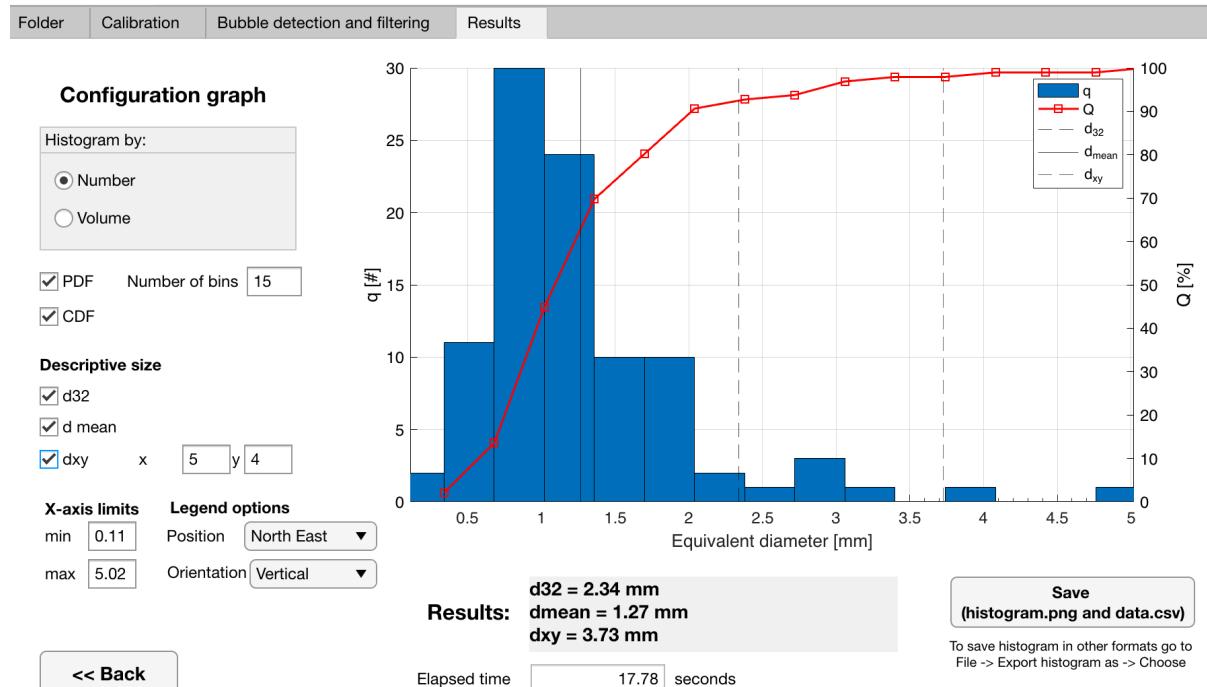
Once you choose the final parameters...

< Prev. Img Next Img >

OPTIONAL: CHOOSE ANOTHER SAMPLE IMAGE TO ANALYSE

Tab 4: Results

The final step in Bubble Analyser is made available after Batch Processing all images. In this Tab the bubble size (equivalent diameter) distribution is shown as a cumulative (red) or density function (blue bars). Both of these can be calculated using a by-number or by-volume basis.



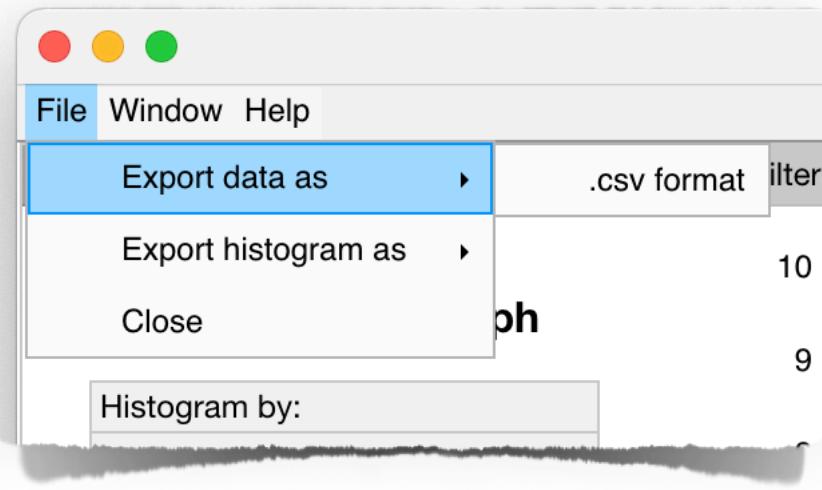
Additionally a set of descriptors can be added to the plot. These are the Sauter bubble diameter (d_{32}), the average (d_{mean}) diameter, and a user-defined descriptor defined as:

$$d_{xy} = \frac{\sum d_i^x}{\sum d_i^y}$$



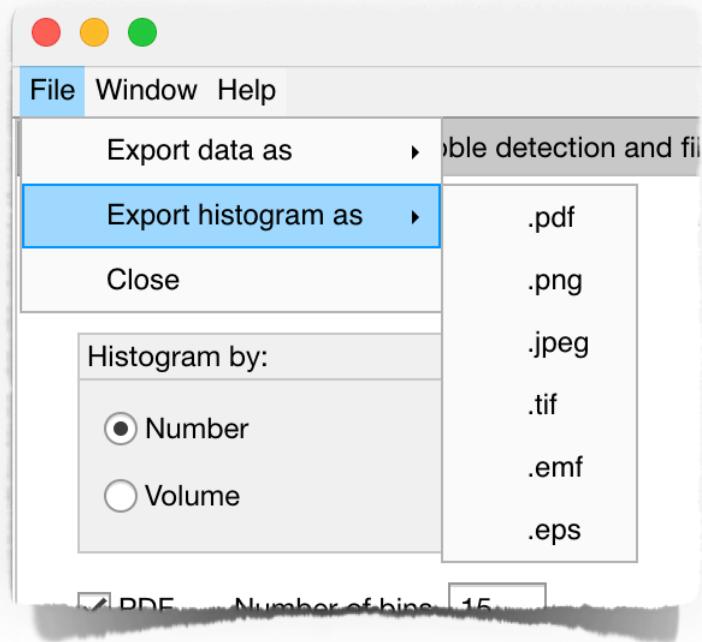
Bubble Analyser will try to adjust the x-axis limits automatically, however, this does not always produce a satisfactory result. For this reason the user can define these limits manually.

If the user is satisfied by the results, these can be exported in the form of a CSV file that will contain a list of all the bubbles detected with their equivalent diameter and area, in mm.



The plot area can also be exported as an image, the supported formats are:

- PDF
- PNG
- JPEG
- TIFF
- EMF
- EPS

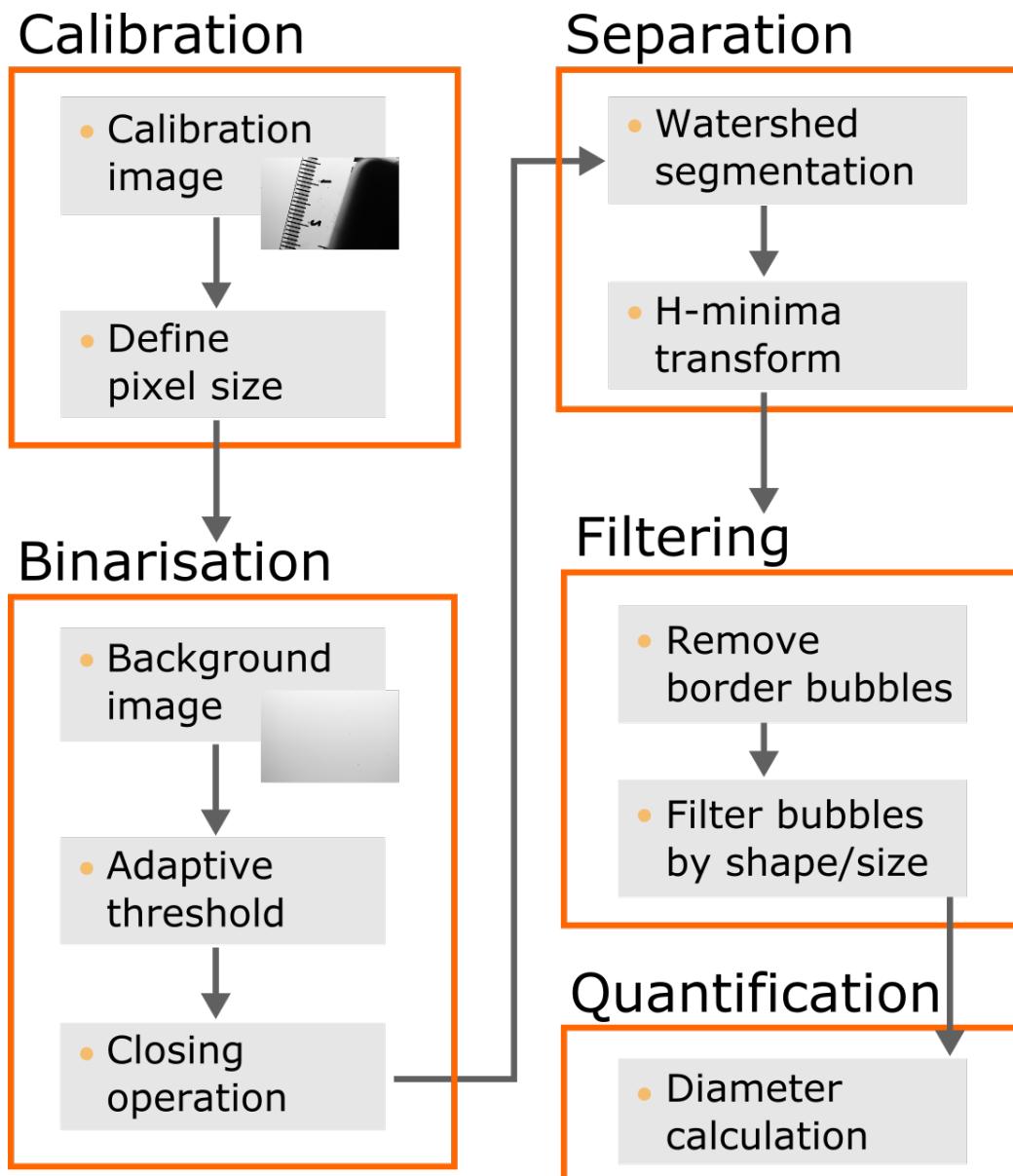


All of the exported files (data as CSV and plots as images) will be saved in the subfolder created in Step 1 (Tab 1 Folder).

Chapter 3

Using the default algorithm

Bubble Analyser's code includes a Matlab routine based on a sequence of actions for segmenting bubbles, separating them from one another and applying filters to remove unwanted bubbles from the analysis (e.g., those not fully in the field of view). The Figure below shows the key steps taken when processing the images.



Tab 2 in the app

The algorithm initially requires a calibration step in which the user needs to provide the image resolution (in number of pixels per mm). This calibration can be done by either entering the resolution manually, or analysing a photography with a 1 cm scale object (e.g., a ruler).

The algorithm also benefits from the user including a background correction image, this is an empty image (no bubbles) to capture spatial differences of lighting. Having this kind of image allows the application of an adaptive thresholding algorithm that will take into account the inherent differences in illumination present in each photograph. If there is no available background correction image, an Otsu threshold algorithm is used to segment the bubbles from the background.

Tab 3 in the app

The binarisation of the images is followed by a closing morphological operation (a disk-shaped structural element is used with a size defined by the user. This step is necessary due to the reflection present in most of the bubbles that makes their centre appear lighter than the borders. Once bubbles are segmented, an additional step is required to ensure individual bubbles are accounted for. Due to the 2D nature of photography, bubbles will overlap and will be segmented as a unit, that will later be measured as a big bubble instead of two smaller ones. A watershed algorithm, with some post processing to avoid hyper-segmentation based on the h-minima transform, is used to perform the task of splitting overlapping bubbles. Because bubbles' size can vary with operating conditions, Bubble Analyser allows the user to change a parameter (marker size) to determine the best value for the user's data. The selection of these parameters was done by visual inspection of the segmentation and results via the app. A final step is used to filter out some bubbles. Unavoidably, some bubbles will not appear wholly in the photograph. These bubbles need to be removed as their size will be skewed towards smaller diameters. Additionally, the watershed algorithm for separating overlapping bubbles can sometimes fail to do so. Those groups of bubbles can be removed based on their shape and/or size. Bubble Analyser comes with a filter for Solidity and Eccentricity.

Tab 4 in the app

After those steps, all of the bubbles' equivalent diameters are calculated and plotted as a distribution. Bubble Analyser allows the user to plot these distributions on a by-number or by-volume base. Descriptive parameters such as mean bubble diameter and Sauter mean diameter are also calculated and plotted.

The graphs shown in Bubble Analyser can then be exported as an image or PDF file. The raw data, each bubble with its equivalent diameter and area can also be exported as a CSV file. The total processing time of all images (elapsed time) is also displayed in the Results tab in the app. This allows comparing the performance of Bubble Analyser with another software or comparing different algorithms within the Bubble Analyser app.

Chapter 4 Using your own algorithms

APP Interface

Bubble Analyser can apply user-defined image processing algorithms by a simple “*interface*” that allows the app to interact and apply such algorithms with ease. If the user wants to use their own algorithms they only need to provide two files: an M-file with the algorithm and a .config file with the parameters required.

Both files must have the same name (different extension, obviously) and need to be saved in the quantification folder of Bubble Analyser so the app can access them and correctly display them in the interface. Once these files are added to the quantification folder Bubble Analyser will add the algorithm’s name in the drop-down menu of Step 7 and will provide the user the chance to edit the default values of the algorithm’s parameters in the table shown in Step 8.

The algorithm

The M-file is required to perform the task of taking individual photos and creating an output that is suitable for Bubble Analyser to show in Step 4. The structure required by Bubble Analyser can be summarised below:

```
function [D, L_image] = Default(img, params)
% Inputs:
%   img: image, either rgb or grayscale
%   params: parameters needed for certain operations, default values set
%           by the corresponding .config file, but can be edited by the user
%
% Outputs:
%   D: number array with the equivalent diameter, in mm, of each bubble
%       detected and segmented
%   L: labelled image resulting from the image processing algorithm
```

Bubble Analyser will call this M-file and provide it with the individual image (`img`) and required parameters. In turn, Bubble Analyser is expecting to receive two outputs (`D`, `L_image`). `D` is a vector array that contains the equivalent diameter for each bubble detected in the image provided. `L_image` is a label image corresponding to the image provided and showing the segmented and kept (after filtering) bubbles.

Bubble Analyser will call this algorithm as many times required for batch processing the images (Step 9). The equivalent diameters for each image is thus collated

The configuration file

Recognising that many image processing algorithms have parameters that need to be adjusted depending on the image resolution, object size, etc. Bubble Analyser provides a way for the user to connect their algorithms to the app so the user can change them in an interactive way via the app.

For achieving this connection the user must provide a .config file (matching the file name of the algorithm). This file has the following syntax:

- Lines starting with a # symbol are ignored, use them for commenting your parameters
- Empty lines are also ignored
- Each line has to contain 3 items separated by commas
 - Name of the parameter (with no spaces, no symbols and starting with a letter)
 - Default value of the parameter
 - Range of possible values

```
==== Parameters Config File ====
#Parameters are described by its name (no spaces), its value, and
its range.
#All three items separated by commas
#For input validation use the range option:
# - square brackets define a continuous range between the values
(inclusive)
# - curly brackets define a discrete range between the values
(inclusive)
# - brackets define a set of possible values listed inside

==== PARAMETERS ====
param_1 default_value, range
...
```

The range of possible values has a particular syntax as well:

- Two values between square brackets, e.g., [1 42]
 - That will define a continuous range between those two values, including those numbers
- Two values between curly brackets, e.g., {1 42}
 - That will define a discrete range between those two values, including those values, e.g., 1, 2, 3,..., 41, 42
- Two values between brackets, e.g., (1 28 121 42)
 - That will define a set of possible values, listed in-between the brackets

About the authors



Dr Francisco Reyes
Research Fellow
JKMRC, Sustainable Minerals Institute
University of Queensland
Australia
f.reyes@uq.edu.au

Francisco obtained his Bachelor of Science in Electrical Engineering from the School of Engineering at Universidad Católica de Chile (PUC) in 2010, where he was also awarded his Master of Science in Electrical Engineering in 2012. Francisco later left Chile for the United Kingdom to pursue a PhD in the Advance Minerals Processing Research Group at Imperial College London. His research focused on the analysis and quantification in mineral processing using X-ray micro tomography images, collaborating with the Universities of Manchester and Nottingham. Upon receiving his PhD, he gained valuable experience as a Research Associate. He joined the Advanced Process Prediction and Control Group (APPCo) within the JKMRC, in 2019.



Paulina Quintanilla
PhD candidate
Royal School of Mines
Imperial College London
United Kingdom
p.quintanilla18@imperial.ac.uk

Paulina obtained his Bachelor of Science in Chemical Engineering from Universidad Técnica Federico Santa María (USM), Valparaíso, Chile, in 2017, where she was also awarded her Master of Science in Chemical Engineering the same year. Since 2018, Paulina is pursuing a PhD in the Advance Minerals Processing Research Group at Imperial College London. Her research is focused on the development of dynamic flotation models for predictive control, collaborating with USM, Santiago, Chile.



Dr Diego Mesa
Research Associate
Royal School of Mines
Imperial College London
United Kingdom
d.mesa@imperial.ac.uk

Diego obtained his Bachelor of Science in Mining Engineering from Universidad de Chile in 2013 where he was also awarded his Master of Science in Extractive Metallurgy in 2015. Diego worked as a metallurgical consultant in Chile for over a year before moving to the United Kingdom, where he obtained his PhD in the Advanced Minerals Processing Research Group at Imperial College London. His research is focused on the quantification of the effects that different design modifications have on froth flotation phenomena. Nowadays, Diego is a Research Associate at Imperial College London, where he continues his research related to froth flotation, collaborating with several partners associated to the Fine Future project, funded by the European Union.