Flanders
State of
the Art

20_035_1
FH Reports

On the accurate resolution of hydraulic jumps in OpenFOAM

Measurement techniques for the study of hydraulic jumps

DEPARTMENT
MOBILITY & PUBLIC WORKS

www.flandershydraulics.be

On the accurate resolution of hydraulic jumps in OpenFOAM

Measurement techniques for the study of hydraulic jumps

López Castaño, S.; Verelst, K.; van Hoydonck, W.

Legal notice

Copyright and citation

This publication should be cited as follows:

López Castaño, S.; Verelst, K.; van Hoydonck, W. (2024). On the accurate resolution of hydraulic jumps in OpenFOAM: Measurement techniques for the study of hydraulic jumps. Version 3.0. FH Reports, 20_035_1. Flanders Hydraulics: Antwerp

Document identification

Customer:	Flanders Hydraulic	Ref.:	WL2024R20_035_1
Keywords (3-5):	PIV, LiDAR, hydraulic jump		
Knowledge domains:	Hydraulics and sediment > Current velocities and patterns > In-situ measurements		
Text (p.):	20	Appendices (p.):	0
Confidential:	No	☒ Available online	

Author(s):	López Castaño, S.

Control

	Name	Signature	
Revisor(s):	Verelst, K.; van Hoydonck, W.	Getekend door:Kristof Verelst (Signature) Getekend op:2024-08-01 14:04:56 +02:0 Reden:Ik keur dit document goed Verelst Kristof Vlaamse overheid	Getekend door:Wim Van Hoydonck (Sign Getekend op:2024-08-05 09:57:00 +02:0 Reden:Ik keur dit document goed Van Hoydonck Wim Vlaamse overheid
Project leader:	López Castaño, S.	Getekend door:Santiago LOPEZ CASTA Getekend op:2024-08-09 15:30:10 +02:0 Reden:Ik keur dit document goed Lopez Castano Santiago Vlaamse overheid	

Approval

Head of division:	Bellafkih, A.	Getekend door:Abdelkarim Bellafkih (Sign Getekend op:2024-08-01 14:10:06 +02:0 Reden:Ik keur dit document goed Bellafkih Abdelkarim Vlaamse overheid

Abstract

This report forms part of the project “On the accurate simulation of hydraulic jumps using OpenFOAM”, where CFD analyses and calibrations are conducted for the study of classical and sloped hydraulic jumps. Complementary physical experiments are conducted in an effort to collect fine-grained data that could be used for verifying the numerical experiments. However, the development of PIV and LiDAR techniques is the primary objective of this report and not the presentation of a quantitative analysis for the experiments conducted. Following reports will detail the results obtained both in laboratory and numerical settings.

In the current study, the development of several techniques for the processing of images in PIV and the cleaning of LiDAR measurements are presented. Experimental campaigns are conducted in FHR, where the measurements conducted present problems that may be alleviated by the aforementioned techniques. Such techniques come as a result trial-and-error and from the revision of existing literature on subjects related to computer vision and statistical data analysis. Here it is intended to document the solutions to the different problems encountered during the experiments and to serve as a precursor for more accurate measurements. Development of software for PIV and LiDAR measurements is still underway, including the techniques discussed here.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

BGR	Blue Green Red
CSV	Comma-Separated Value Format
CFD	Computational Fluid Dynamics
EFD	Experimental Fluid Dynamics
FFT	Fast Fourier Transform
FHR	Flanders Hydraulics Research
LiDAR	Light Detection and Ranging
OpenCV	Open-source Computer Vision Library, version 4.3 (opencv.org)
PIV	Particle-Image Velocimetry
ROI	Region Of Interest

Latin symbols

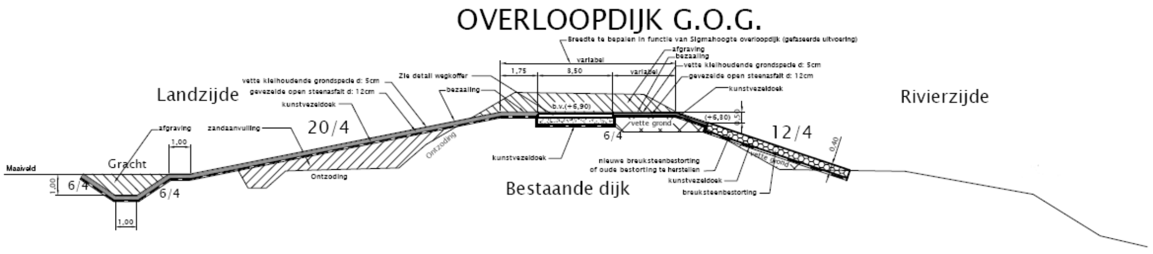
Fr	Froude Number	-
I	Intensity, as a char type	-
L	Geometric Scale	m
M	Inertial Scale	kg
Q	Discharge	l/s
q	Unit Discharge	l/s/m
Re	Reynolds Number	-
T	Temporal Scale	s

Greek symbols

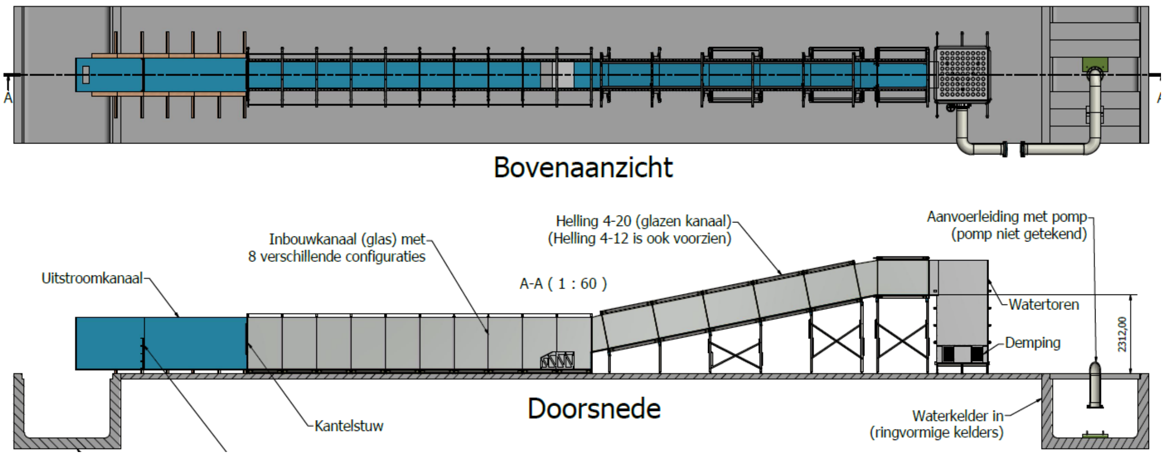
μ	Mean	-
ν	Kinematic viscosity	m^2/s
σ	Variance	-

1 Introduction

This document reviews the techniques used for the measurement of the velocity field and free-surface profile for the overflow over a dike and the subsequent hydraulic jump on model scale. Experiments are conducted using the overflow dike model of FHR, in order to test such techniques. Results and suggestions are reported here. Details on the design and construction details of the model dike are given in Vercruysse *et al.* (2018). The prototype dike and a simple overview of the model are shown in Figures 1(a)-(b) without mentioning any details about the studied configurations of the energy dissipation structure at the toe of the dike.



(a) Prototype-scale overflow dike with energy-dissipation structure at the toe in the land side.



(b) model-scale overflow dike with base configuration at the toe.

Figure 1 – Overflow dike at prototype and model scale.

The model details will not be described entirely, only some of the relevant parameters and scaling principles will be discussed. The model dike's scale is 1:3 (scale-to-prototype). The flow discharge ranges from 16 to 270 l/s or the unit discharge ranges from 21.3 to 360 $l/s/m$. A downstream weir is used to control the water level at the sequent section of the flume. The elevation of the weir can be regulated, in order to produce different types of hydraulic jumps, as described in Ohtsu and Yasuda (1991). The discharge is kept constant using a frequency-drive axial pump connected to a closed circuit. The flow in the slope is not pre-aerated.

Note that the present model, according to the parameters just shown, is expected to suffer from scale effects. As it is common practice in free-surface flows, the Fr-similarity principle will be used for the up-scaling of the results obtained in this work.

In Table 4 the scale ratio λ is equal to the ratio between the prototype length scale (subscript 'p') and the model length scale (subscript 'm'), or L_p/L_m. By virtue of the kinematic similarity, we have that the prototype

Table 4 – Froude and Reynolds similarity principles (Rouse, 1946).

Parameter	Dimension	Froude	Reynolds
	Geometric Similarity		
Length	$[L]$	λ	λ
Area	$[L^2]$	λ^2	λ^2
Volume	$[L^3]$	λ^3	λ^3
Rotation	$[-]$	1	1
	Kinematic Similarity		
Time	$[T]$	$\lambda^{1/2}$	λ^2
Velocity	$[L/T]$	$\lambda^{1/2}$	λ^{-1}
Accel.	$[L/T^2]$	1	λ^{-3}
Discharge	$[L^3/T]$	$\lambda^{5/2}$	λ
	Dynamic Similarity		
Mass	$[M]$	λ^3	λ^3
Force	$[ML/T^2]$	λ^3	1
Stress/Press.	$[M/LT^2]$	λ	λ^{-2}
Work	$[ML^2/T^2]$	λ^4	λ
Power	$[ML^2/T^3]$	$\lambda^{7/2}$	λ^{-1}

discharge ranges:

$$Q_p = \lambda^{5/2} Q_m = \begin{cases} 3^{5/2} \times 16 \approx 250\, l/s \\ 3^{5/2} \times 270 \approx 4200\, l/s. \end{cases}$$

Notice that such range covers what is expected in common engineering practice. However, in relation to flow turbulence the ranges covered by the model are constrained by the Reynolds number:

$$Re_p = \lambda^{3/2} Re_m = \lambda^{3/2} \frac{q_m}{\nu} \approx \begin{cases} 110700 \\ 1900000, \end{cases}$$

where it is assumed that water is used both at the prototype and the model dikes. The unit dischage, q, is used for the calculation of the Reynolds Number, Re.

By choosing a 1:3 scaling we might be inaccurately representing some phenomena at the prototype scale directly linked to turbulence, such as: (1) energy dissipation – or head drop, (2) roller length in the hydraulic jump, and (3) aeration in the sloping channel. We must, at least, make sure that the flow at the model scale is indeed turbulent, and fully developed. In the latter case, we rest assured that the flow in the scale model is representative of *fully developed* turbulent flows at prototype scale, according to the scaling arguments just discussed.

This report is organized as follows: first, a short description of the experimental campaigns together with the different measurement techniques used will be given in section 1.1. Chapter 2 will describe how the data will be processed, that is, the methodology used to rationalize the data obtained from the different measurements. The processing of the measurements will be shown in chapter 3, and the conclusion will be presented in chapter 4.

1.1 Experimental campaign

In short, two measurement techniques will be used in this study: (1) 2D-PIV, and (2) LiDAR. The first refers to the processing of still images to extract velocity fields, more precisely Particle Image Velocimetry. In this work, bubbles will be used as tracers (looking from the side). The objective will be to develop an in-house code for the

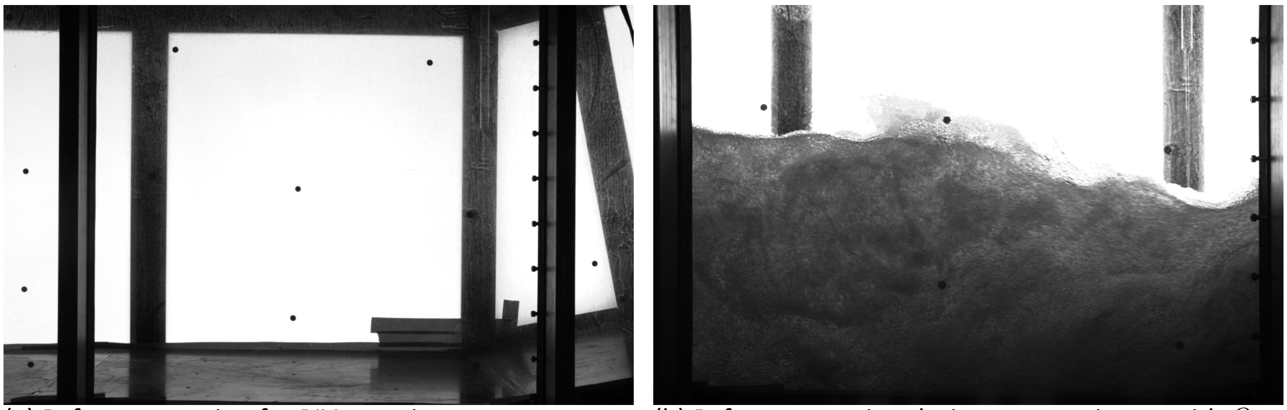

(a) Reference section for PIV experiments. (b) Reference section during an experiment with $Q = 150\,lt/s$.

Figure 2 – Reference section chosen for the collection of PIV images. Notice the markers on the glass indicating geo-referenced coordinates measured at such locations.

processing of images and velocity field eduction using PIV. The second technique, LiDAR, allows us to extract the free-surface profile from laser sheets (emitted from above) reflected back from the flow. Extracting the reading from the LiDAR is simpler, since the hardware returns polar coordinate measurements (r, θ) instead of the more general reflectivity values in dB, common for this kind of equipment.

Given the time constraints only two experiments were conducted. For the present study, the simplest configuration for the stilling basin at the toe of the dike was considered: a flat bed downstream of the toe of the dike. This case is also the most common in inland dikes serving rivers and on embankments for which overflow was not foreseen in the design. Such configuration is uncontrolled downstream, that is, the location of the hydraulic jump is not fixed and its location is function of the sequent depth and the discharge. However, we are interested in *Type B* jumps, or jumps where part of the roller is located upstream from the toe, given its complexity and high erosion potential. The location of the jump on the model is set by operating the weir at the outfall. A discharge of 100 l/s and of 150 l/s is used for the experiments, but results from PIV are shown only for the lower discharge.

On one hand, two camera sensors were used for the collection of images during the experiments. For reasons that will be clear later on, the technical details (vendor, lens, etc.) of each sensor are of no particular importance except that both sensors[1] are low-end single-exposure cameras, one of them has a maximum frequency of 155 Hz, while the other has a maximum frequency of 240 Hz. Data collection on the slower sensor was made using an external computer. For the other camera, videos were filmed and then images extracted from the resulting MP4 file. The cameras were set (almost) perpendicular and laterally to the flume, looking directly in the region where the toe of the hydraulic jump is expected to occur. The section chosen for filming and image-capturing is shown in Figure 2.

On the other hand, a 2D-LiDAR was placed above the flume (vertically on top of the toe of the dike) with the sensor looking through a vertical plane (containing the axis of the flume) and covering a range of 185°. The field of view covers most of the flume and the slope. The technical details, use, and data format of the apparatus are well described by Wolput (2021), and will not be repeated here.

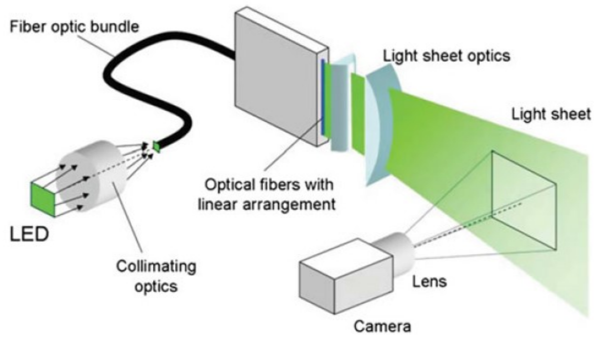
[1]For completeness, one of the cameras is a A71 Samsung cellphone and the other is a uEYE 160Hz sensor.

2 Theoretical considerations for PIV and LiDAR

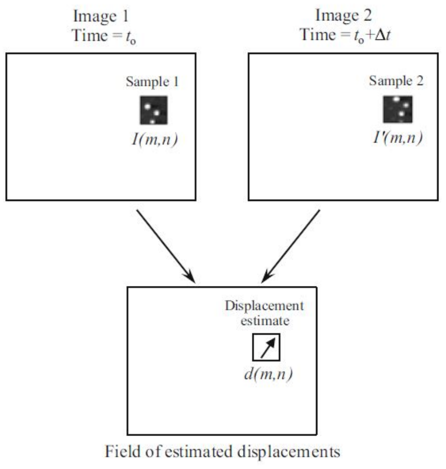
This chapter discusses the fundamentals of 2D-PIV and 2D-LIDAR. Section 2.1 describes the basics of PIV and LIDAR. The camera calibration for the PIV measurement is discussed in section 2.2. Some of the discussion on PIV is based on the book of Raffel *et al.* (2018).

2.1 Basics of PIV

Particle-Image Velocimetry (PIV) pertains the calculation of velocity fields using pairs of images captured with sensors (could be cameras, laser sheets, or other). Such eduction makes use of the *Lagrangian* framework: the displacement of inert markers within the flow and across the two images, spaced a certain time interval, are used to calculate the velocity field. Thus, PIV is concerned with the detection and tracking of such markers within the fluid flow.



(a) Typical PIV arrangement with illumination sheet.



(b) Basic input and output from PIV analysis.

Figure 3 – Basic setup and results obtained from PIV.

The basic experimental setup is a single sensor positioned orthogonally to the region of the flow where we are interested in measuring the velocity field. For best results, an illumination sheet (be it a laser pulse or LED) is set to illuminate the plane of interest for the sensor and to highlight the markers. Note that with a single sensor one can only measure image-planar displacements of the markers; more sensors are needed to measure ”depth”. A sketch of a 2D-PIV setup is shown in Figure 3.

The evaluation of the markers’ displacements is made through a spatial cross-correlation of pixel intensities[2] across both images. The auto-correlation maps need to be normalized, in order to avoid peaks with same values

[2] A greyscale (black-and-white) image is a M-by-N matrix where each entry (pixel) represents an intensity I ranging from zero (white) to 255 (black).

across different samples. The normalized auto-correlation C^* of pixel intensities is defined then as:

$$C^*(x,y) = \frac{C(x,y)}{\sqrt{\sigma_1}\sqrt{\sigma_2}}, \tag{1}$$

$$C(x,y) = \sum_{i=0}^{M}\sum_{j=0}^{N}[I_1(i,j)-\mu_1][I_2(i+\Delta_i, j+\Delta_j)-\mu_2], \tag{2}$$

$$\sigma_1 = \sum_{i=0}^{M}\sum_{j=0}^{N}[I_1(i,j)-\mu_1]^2, \tag{3}$$

$$\sigma_2 = \sum_{i=0}^{M}\sum_{j=0}^{N}[I_2(i,j)-\mu_2]^2, \tag{4}$$

where Δ, σ and μ are pixel displacements, variance, and the mean, respectively. The analysis consists in splitting the images in samples (or interrogation windows) where the auto-correlation is calculated and where the displacements are estimated by connecting the 'peaks' (or maxima) of the auto-correlation map. In summary, the analysis of a pair of intensity images can be divided in the following steps:

Step -1 Calibrate sensor/camera, determine its pose, and un-distort images.

Step 0 Enhance or remove features from the two images.

Step 1 Divide the images in interrogation windows, compute μ and $\sigma^{1/2}$.

Step 2 Subtract the mean from each window.

Step 3 Compute the auto-correlation for each window.

Step 4 Divide the auto-correlation by the standard deviation of each image.

Step 5 Determine displacements by detecting 'peaks' in the auto-correlation maps.

Step 6 Remove 'rogue´ vectors, or outliers, from the sampled field.

From step 1 to step 5 the procedure is pretty much standard, and algorithms can be found in any textbook on measurement techniques for EFD or even the internet. It is important to mention that the auto-correlation is calculated in spectral space, via FFT, in order to reduce the complexity[3] of the calculations. This trick comes from Parseval's identity, where one can reduce a convolution of two complex valued fields (in this case auto-correlation of two images) to the simple multiplication of their Fourier transforms.

However, steps -1, 0, and 6 are not as thoroughly discussed and is oftentimes not paid much attention. The reason for this stems from the quality of the sensors and the setup used for 2D-PIV. Typically, such experiments are conducted with high-quality cameras installed in already calibrated frames. In short, this means the experimenter has a relationship between pixel displacement per frame and unit length per unit time. Software such as OpenPIV require such relationship as an input. That's not the case in general where one may end up using low-quality sensors (like cellphone/drone cameras) in non-laboratory settings. Plus, one may need to relate the world's frame of reference to the xy-pixel plane of the intensity image in order to locate the velocity field in space. Furthermore, in the "real world" bad illumination and uneven light scattering[4] are the norm, hence the need for additional treatment. Lastly, depending on the type of the markers, it might be necessary to use intensity-capping techniques to enhance the images.

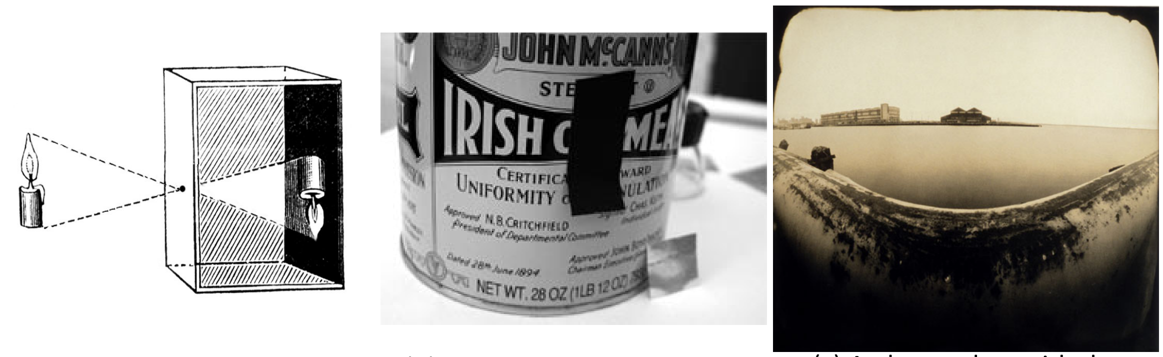
[3]Complexity understood as number of operations (flops) required for a certain routine to be completed. The complexity in this case is reduced from $O((N\times M)^4)$ to $O((N\times M)\log(N\times M))$.

[4]To be precise, light scattering in the Mie regime.

2.2 Camera calibration: Basics of Image formation

2.2.1 Distortion

Images taken using cameras are distorted. One of such distortions is known as ”fisheye” or ”barrel” distortion, produced by the necessity of setting an equal (radial) distance between the film and the pinhole (sensor) of the camera. This is the case for the pinhole cameras (Camera Obscura) we used to make from aluminum cans, as depicted in Figure 4. Notice from the image below the significant radial distortion on the pier, which causes it to be curved. Also notice the tangential distortion on the building on the left: it seems to shrink as one goes to the borders of the photo. Another kind of distortion is the flipping of the image as it enters the pinhole (sensor): for such no corrective measures are needed as modern digital cameras do that automatically by writing the pixels on memory from the bottom up.



(a) Basics of a pinhole camera. (b) An actual pinhole camera made from an aluminum can. (c) A photo taken with the camera in (b).

Figure 4 – pinhole camera and source of distortions *(source: google.com)*.

Modern cameras may suffer from the same radial and tangential distortions as our primary school pinhole cameras, hence corrections for such defects may be needed in order to make quantitative calculations. Radial distortions may be represented as Taylor expansions around the radius r:

$$x_0 = x\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right),$$
$$y_0 = y\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right),$$

where (x_0, y_0) are the (distorted) image’s (row, column) indices (or coordinates), and (x,y) are the undistorted or corrected coordinates. A similar approach can be taken for describing the tangential distortions, by re-projecting the image on a second order polynomial:

$$x_0 = x + \left[2p_1 xy + p_2(r^2 + 2x^2)\right],$$
$$y_0 = y + \left[2p_2 xy + p_1(r^2 + 2y^2)\right].$$

Notice that these correction factors are intrinsic to the camera (sensor) one is using, that is they don’t change unless you change cameras[5] or lenses. The five distortion parameters defining the camera are then the following:

$$\text{distortion} = (k_1\ k_2\ p_1\ p_2\ k_3)\,.$$

Note that this is not the only way of describing the camera’s intrinsic parameters. For the present purposes, however, such parametrization will be assumed since some Computer-Vision libraries, such as OpenCV, use them.

[5]This is the reason why we don’t delve into the technical specifications of the cameras/lenses used. A mathematical description of its intrinsics is all we need.

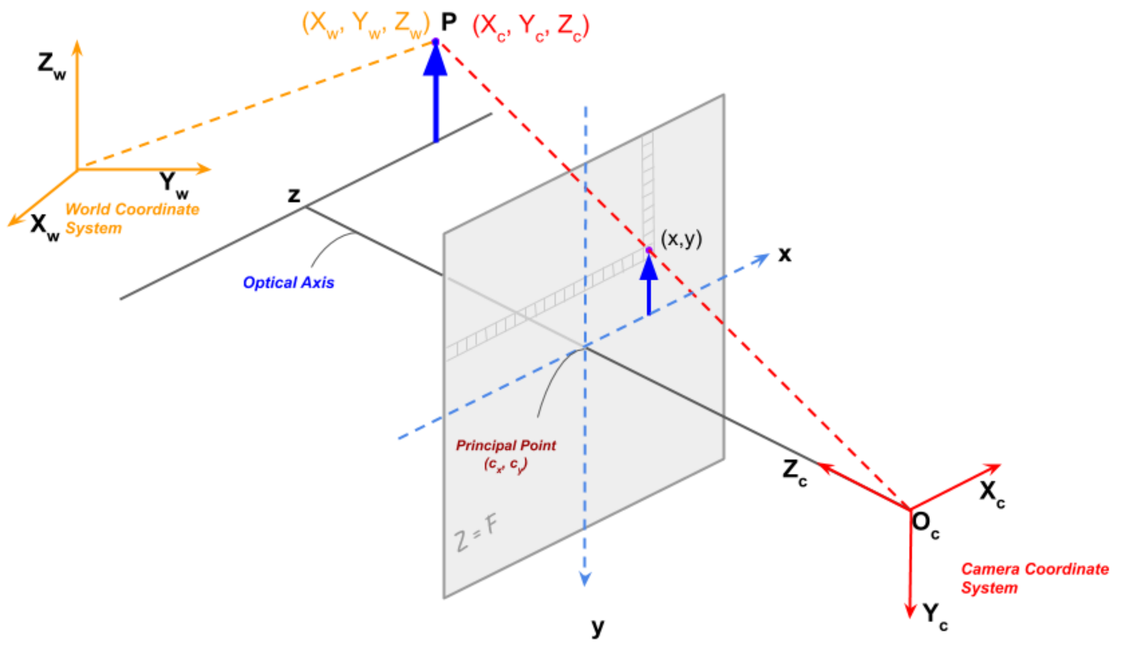


Figure 5 – Graphical representation of the change of coordinates. The point $p = p(x, y)$ is located on the image plane, and its coordinates are in pixels.

2.2.2 Distortion Transformation: Camera Matrix

Although the distortion coefficients don't change for the camera, this doesn't mean they remain invariant under change of coordinates. Said plainly, one has to scale such coefficients as one focus/zooms the camera. This can be easily understood if we take the pinhole photograph, shown in Figure 4(c), and do the mental exercise of zooming into the two-story house (initially undistorted) along the optical axis of the pinhole: Will the house distort as one gets closer with the camera? Of course it will. Why? Because the house is "bigger" now (is scaled).

Without going much into details the camera matrix, which scales the distortion parameters, is function of the focal length (f_x, f_y) of the camera and the location of the principal point (see Figure 5) in the image plane (c_x, c_y):

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix},$$

Thus, if one is interested in recovering the three dimensional coordinates P_c, with respect to the pinhole, from a certain point p in the undistorted image (in pixels) one can use the following transformation:

$$\mathbf{p} = \mathbf{K}\mathbf{P}_c.$$

2.2.3 Camera Pose: Coordinate Transformation

The objective is to transform the camera's coordinate system onto the world's coordinate system. Let a point p be in the undistorted image plane, corresponding to a point P_w in the world's coordinate system, and P_c in the Camera's coordinate system.

As a visual aid for what it comes next, a sketch of the coordinate transformation is shown in Figure 5. The homogeneous transformation of coordinates is given by the following expression:

$$s\,\mathbf{p} = \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{P}_w, \tag{5}$$

where **R**, **t**, and s are the rotation matrix, a translation vector, and a scaling factor, respectively. Note that in homogeneous coordinates an additional dimension to the coordinates is added in order to represent a family of ALL (even with infinite length) arbitrary vectors that ”point to the same place”. That is, a vector (x, y, z) can be represented as $(p/w, q/w, r/w, w)$ where w is any real number. This allows for the matrix $[\mathbf{R}|\mathbf{t}]$ (3x4) to pre-multiply an homogeneous vector P_w (4x1).

It is important to know that the scaling factor s has nothing to do with zooming, or ”making things bigger/smaller”. When the point **P** does not lay on any of the planes crossing at least two of the world’s coordinate system’s axes, then s is different from unity and needs to be found.

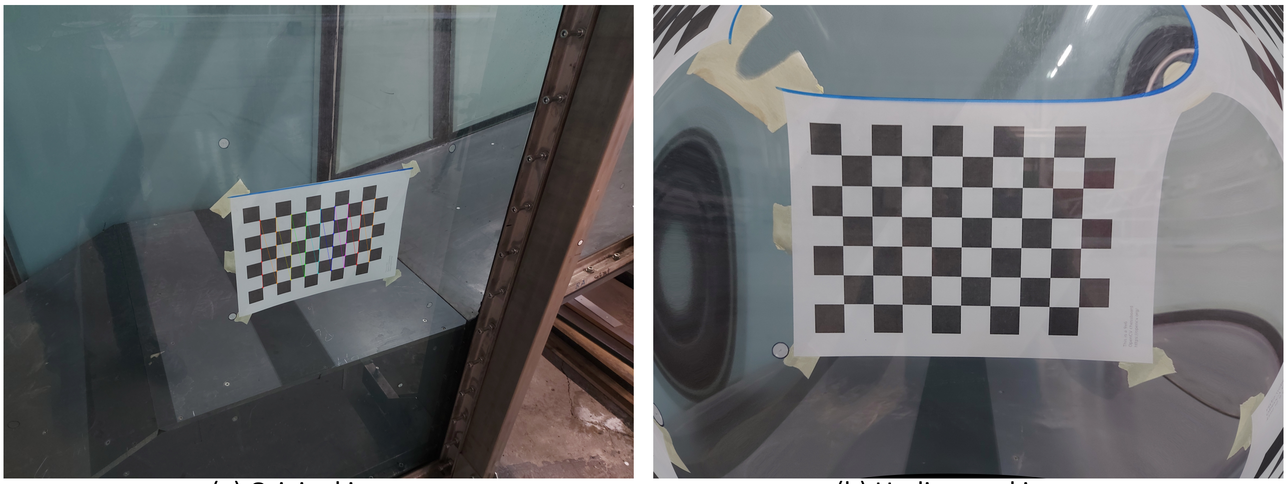

(a) Original image. (b) Undistorted image.

Figure 6 – Image transformation (distortion and pose) using OpenCV.

Finally, as a test for all the theory just discussed in this section, we produce the calibration for the camera in a Samsung A70 cellphone. By means of the various subroutines contained in OpenCV, the camera calibration and un-distortion of images is conducted and the results shown in Figure 6. The code presented in Listings 2.1 are the camera intrinsics, rotation matrix and translation vector of the camera/pose used to un-distort the image just shown.

Listing 2.1 – Camera intrinsics and pose using a Samsung Galaxy A70 sensor.

```
<?xml version="1.0"?>
<opencv_storage>
<camera_matrix type_id="opencv-matrix">
  <rows>3</rows>
  <cols>3</cols>
  <dt>d</dt>
  <data>
    3.4061945856932407e+03 0. 2.2588205259897909e+03 0.
    3.3939950891165668e+03 1.8360728114692490e+03 0. 0. 1.
  </data></camera_matrix>
<distortion_coefficients type_id="opencv-matrix">
  <rows>1</rows>
  <cols>5</cols>
  <dt>d</dt>
  <data>
    2.8939230574747399e-01 -6.8192035262471684e-01
    1.1103757192316941e-02 -1.0109344428553331e-02
    -2.2644743462372081e+00</data></distortion_coefficients>
<rotation_vector type_id="opencv-matrix">
  <rows>7</rows>
  <cols>1</cols>
  <dt>"3d"</dt>
  <data>
    4.1095181950816423e-01 5.0626570027045359e-01
    -1.5217660013498033e+00 5.6205479906540234e-02
    9.9216475132382784e-02 -1.5718596835308818e+00
    -4.6231606573054124e-01 6.4342700983759027e-01
    -1.4398004907207580e+00 -4.5461295586244471e-01
    1.6502402139810465e-01 -1.5478877363001502e+00
    -7.0071705259357756e-02 5.8651764481360273e-02
    -1.5551036048251083e+00 7.1278921288504465e-02
    9.7848211166912141e-02 -1.5575190621538342e+00
    7.8560340166416576e-01 1.3783334818479678e-01
    -1.5967962825730679e+00</data></rotation_vector>
<translation_vector type_id="opencv-matrix">
  <rows>7</rows>
  <cols>1</cols>
  <dt>"3d"</dt>
  <data>
    -5.8648411996546464e+01 2.8770949620325993e+01
    5.8782329648163341e+02 -3.9092950003906573e+01
    3.4446358358190516e+01 8.7980443424625037e+02
    -3.8903152366185111e+01 -1.5918744868882595e+01
    5.9868126638060050e+02 -5.7101076372048155e+01
    1.1562921845890799e+01 4.6743260849021567e+02 4.4353258318037440e+01
    -1.0907419724229962e+02 1.4829818316728517e+03
    -8.0040674933184704e+01 3.3111192688092579e+01
    3.1778824141761345e+02 -6.3754005856348350e+01
    4.1051110666337145e+01 6.1365963607864342e+02
    </data></translation_vector>
</opencv_storage>
```

2.2.4 Image Enhancing: Capping

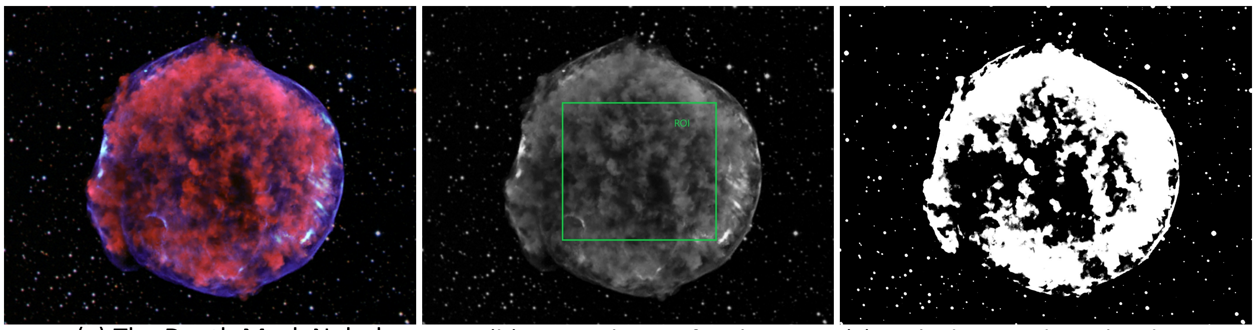
Capping is a technique that differs somewhat from filtering, or sharpening, and thresholding of images. The procedures and use of the latter tools for image enhancing are well documented, and will not be discussed here. However, capping is a technique based on heuristic arguments and is not necessarily useful in all cases. Basically, the technique *caps* the intensity matrix according to certain upper/lower limits, and then proceeds to normalize the image according to the newfound extrema. By doing this one can 'de-saturate' a poorly illuminated image, for instance. Although this technique is not useful for our specific case, the Author finds it important to leave some documentation about it in case a more thought out experiment is conducted. The algorithm is pretty simple:

Step 1 Calculate the minimum/maximum thresholds used for capping the intensity map, for a ROI in the map. For PIV it is common to clip the intensity in the following manner:

$$\mu(I_{\text{ROI}}) - \sqrt{\sigma(I_{\text{ROI}})} < I(i,j) < \mu(I_{\text{ROI}}) + \sqrt{\sigma(I_{\text{ROI}})}.$$

Step 2 Cap the image.

Step 3 Normalize the image according to the new minima/maxima.



(a) The Death Mask Nebula *(source: space.com)*. (b) A ROI chosen for the calculation of statistics. (c) High-density dust clouds in the nebula enhanced.

Figure 7 – Intensity-capping example.

As an example imagine we want to enhance some feature in an over-saturated image, like the dust clouds in the nebula shown in Figure 7(a). Clearly the over-saturation comes from deep space (which is black). If one proceeds to cap the intensity matrix using an appropriate ROI –see Figure 7(b), then the resulting enhanced features are shown in Figure 7(c). Notice that the procedure was done over the greyscale version of the original image: one can trivially extend the method for each channel in a BGR (color) image.

2.2.5 Removing outliers

The procedure of removing 'rogue' vectors, or outliers, from the resulting displacement field is very similar to what it was discussed in section 2.2.4. One needs to basically cap the vector field $u(x,y)$ in the following fashion:

$$\mu(u) - 3\sqrt{\sigma(u)} < u(x,y) < \mu(u) + 3\sqrt{\sigma(u)}.$$

According to the formula an outlier occurs less than 0.3% of the times, assuming the error in the PIV to be *diffusive* (as in a Random Walk). Such assumption may have some validity when processing noisy images. It obviously depends on the type of noise.

2.3 Processing of LiDAR data

Data acquisition from the 2D-LiDAR equipment involves a polar-to-cartesian coordinates conversion. Furthermore the data from the LiDAR comes in CSV format. The methodology thus involves the aforementioned conversion and the extraction of several columns of data from the file. The snippet shown in Listings 2.2 summarizes the post-processing algorithm. Note that some of the conversion factors are non-standard, and given by the vendor.

Listing 2.2 – pre-processing data from LiDAR using NumPy.

```
name00 = 'LMS5xx_FieldEval_PRO (SN 20500041)'
name0 =  name00
                  '.ScanData.aDataChannel16[0].aData'
name1 = name00
                 '.ApplRange.aRange[0].diStartAngle'
name2 = name00
                 '.ApplRange.aRange[0].diStopAngle'
name3 = name00
                 '.ApplRange.aRange[0].udiAngleRes'
name4 = name00
                 '.ScanData.aDataChannel16[0].DataChannelHdr.uiAngleRes'
for csvfile in csvfiles:

    Din=pd.read_csv(os.path.join(folderin ,csvfile),sep=';')
    Dm_S=Din.filter(like=name0)

    diStartAngle=Din[name1][0]/1e4
    diStopAngle=Din[name2][0]/1e4
    udiAngleRes=Din[name4][0]/1e4

    angle=np.zeros(len(Dm_S.columns))
    shift={}
    shift['Angle']=90
    shift['Xshift']=0
    shift['Yshift']=0
    shift['Xswap']=1
    shift['Yswap']=1

    for nr,col in enumerate(Dm_S.columns):
        colnr=col.split('.')[-1]
        colnr=colnr.replace('aData[','')
        colnr=colnr.replace(']','')
        colnr=int(colnr)
        angle[nr]=diStartAngle+colnr*udiAngleRes+shift['Angle']

    Dm_Scleaned=Dm_S/1000
    Dm_Scleaned=Dm_Scleaned.replace(0,np.nan)
    Dm_Scleaned[Dm_Scleaned>8]=np.nan
    Dm_X=Dm_Scleaned*np.sin(angle*np.pi/180)*shift['Xswap']
        +shift['Xshift']
    Dm_Y=Dm_Scleaned*np.cos(angle*np.pi/180)*shift['Yswap']
        +shift['Yshift']
    Dm_Xm=Dm_Scleaned.mean(axis=0)*np.sin(angle*np.pi/180)
        *shift['Xswap']+shift['Xshift']
    Dm_Ym=Dm_Scleaned.mean(axis=0)*np.cos(angle*np.pi/180)
        *shift['Yswap']+shift['Yshift']

    X=Dm_X.values #Final results
    Y=Dm_Y.values
```

Sometimes, it might be necessary to reconstruct bad readings (NaN) from sampled points. Here, as a simple second-order extrapolation[6] of the calculated water depth is done in the following manner:

$$h_i = (3/2)h_{i-1} - (1/2)h_{i-2},$$

where h_i and i are the the equispaced water depth readings and the running index, respectively. Given that LiDAR emit pulses every 0.33° over an arc of 185°, the error associated with extrapolating heights on unsampled

[6]Interpolations may produce further bad readings if two or more consecutive readings are bad readings.

points is expected to be negligible given how close to each other the samples are.

3 Results

3.1 PIV

This section will show the progress made so far in the development of a code for processing PIV images that is suited for the needs of FHR. As mentioned in previous chapters, the intent is to be able to deploy low-end cameras in the field for PIV/PTV measurements. This implies that we have to come up with a way to determine the camera pose and calibration as fast and as simple as possible. Notice also that determining the camera pose by georeferencing several points in the field is rather expensive, thus one may be inclined instead to determine the camera pose by *detecting patterns* and determine the world's coordinates from such pattern (like we did with the checkerboard shown in Figure 6).

However the results here shown have only considered Steps 1 to 5 from the PIV analysis. Given that the amount of data collected and processed is rather high (around 1 Tb of data in total), this chapter will only focus on results that support the arguments held in this work.

3.1.1 Camera speed and PIV: it matters

Tests were conducted with a low-end sensor with adjustable framerate (up to 160 Hz), and used another camera with fixed framerate (240 Hz). Samplings at 60, 120, 155, and 240 Hz were taken at different times for the two discharges (100 and 150 l/s). Since the bubbles, product of the self-areation induced by the hydraulic jump, are used as markers it is important to assess whether the auto-correlation is able to deal with such dense sample (at each framerate). Such rating will be *heuristic*, that is, a visual examination of the displacement vectors is made and these should exhibit reasonable flow patterns.

A sequence of images and their respective displacement vectors, at different framerates, are shown in Figure 8. Since the tests were made at different times, it is not possible to do a vis-à-vis comparison between framerates. However, the results are still useful in determining the importance of framerate when using bubbles as markers. As one moves from Figure 8(a)-to-(d) it becomes apparent that the presence of outliers (or rogue vectors) reduces. Such reduction is quite evident when passing from 60 Hz to 120 Hz, however more coherence in the flow is seen as one goes to higher framerates. One can readily do an estimation, based on Stokes' drag theory, of the *time lag*[7] τ_P produced if one assumes the mean bubble size to be of 5 mm:

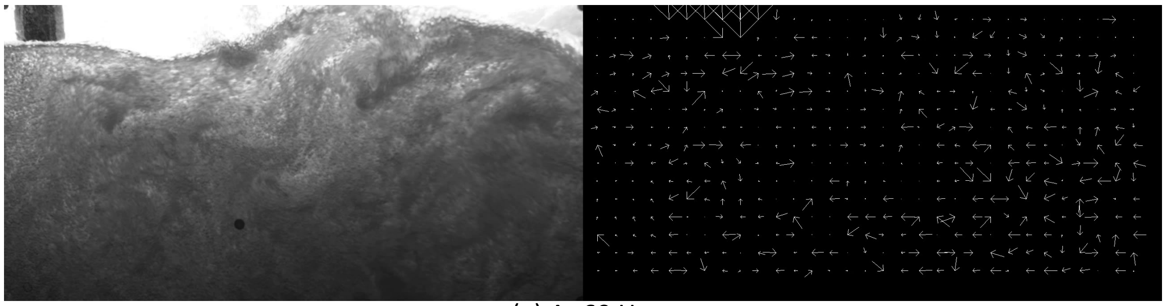
$$\tau_P = d_P^2 \frac{\rho_P}{18\mu_l} \approx 0.0012s,$$

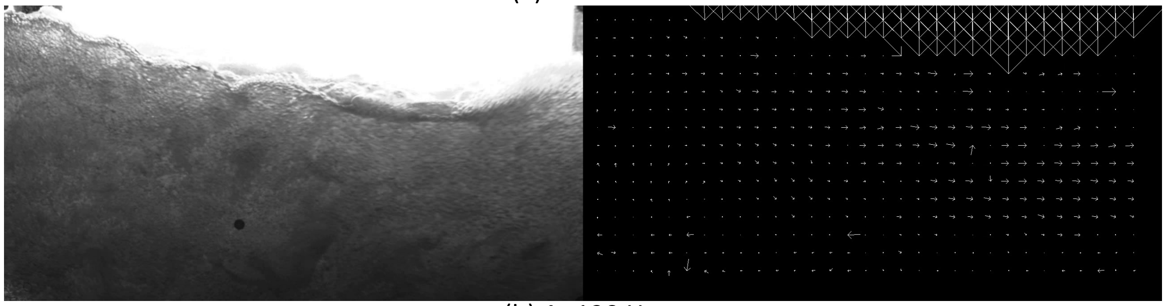
where d and ρ are the diameter and density of the bubble P, and μ_l the dynamic viscosity of the surrounding fluid. This means that in order to neglect the influence of buoyancy in our PIV measurements, we have to work with framerates higher than 1000 Hz. In literature (Raffel *et al.*, 2018), it is suggested to work with the following limit:

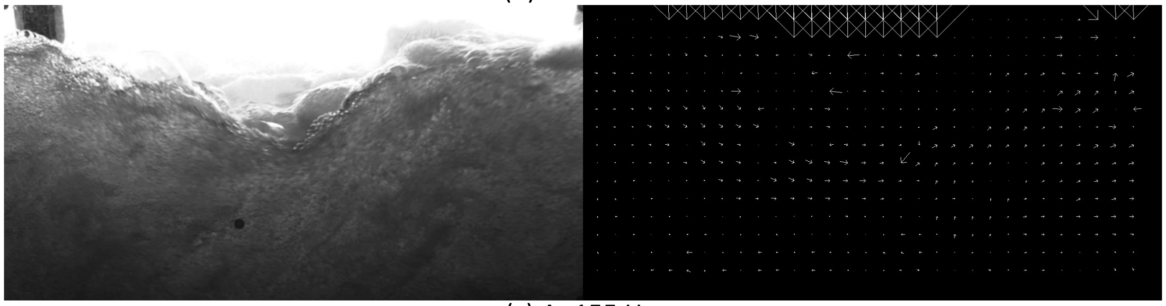
$$\tau_P \times \text{framerate} > 0.1,$$

which lowers the framerate requirements down to around 100 Hz. Interestingly enough, our heuristic analysis is proving the validity of such assumption.

[7]The time it takes for a buoyant marker to respond to a certain flow feature


(a) At 60 Hz.


(b) At 120 Hz.


(c) At 155 Hz.

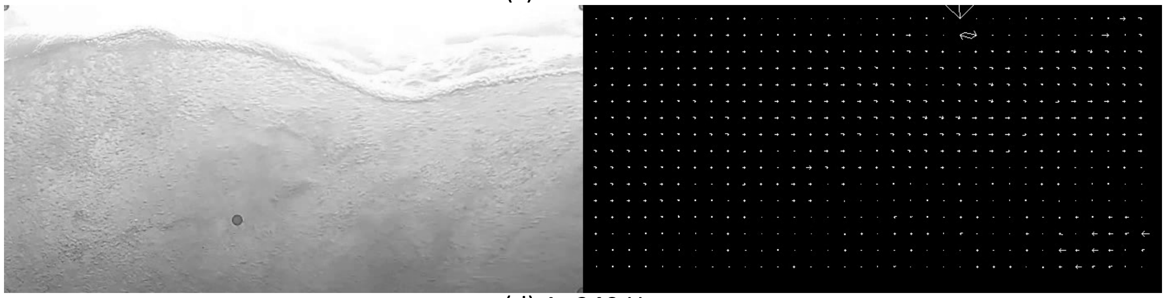

(d) At 240 Hz.

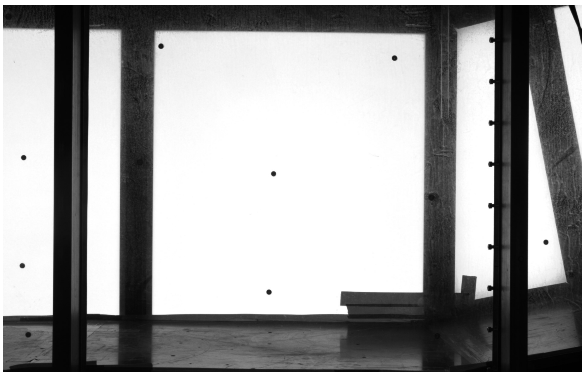
Figure 8 – PIV calculation of peak displacements at different camera framerates, at $Q = 150\, l/s$. The rogue vectors usually tend to exhibit very large displacements or awkward direction.

3.1.2 Affine Transformation vs. Camera Calibration

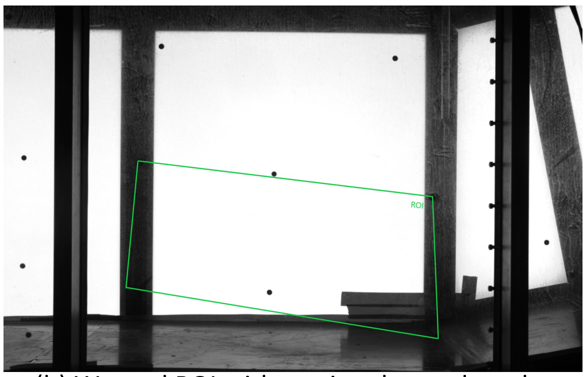
It is important to mention that an *affine* transformation, instead of a camera calibration, was made for the images taken from the sensors. That is, a ROI is selected where a 2D-to-2D transformation of the image is done so as to obtain a "bird view"[8] over the region chosen. Notice that said transformation involves pixels only: neither camera distortions nor coordinates are taken into consideration. This technique is valid as long as the chosen ROI is not warped, and the camera looks perpendicularly to the plane being sampled.

The affine transformation performed for one of the various experiments is shown in Figure 9. Note that the ROI is warped because the georeferenced markers are located in somewhat arbitrary positions. This transformation fixes the eye orthogonal to the selected view, but doesn't preserve neither perspective nor considers camera

[8]Some authors prefer to call this an "orthogonalization".



(a) Empty section with georreferenced markers.



(b) Warped ROI with vertices located on the markers.

(c) Transformed image.

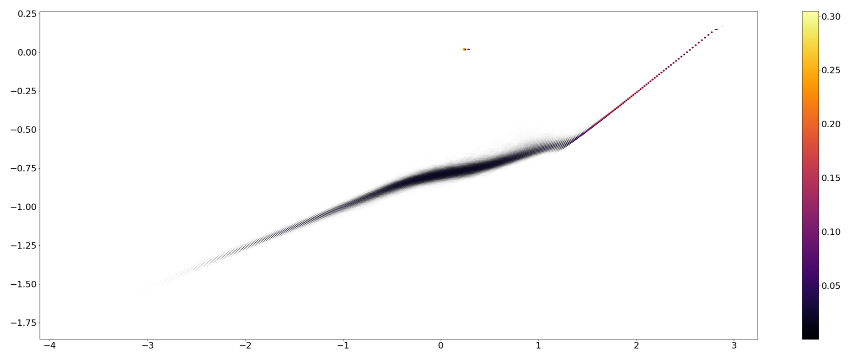
Figure 9 – Affine transformation using georeferenced markers. The coordinates themselves are not used in the transformation.

distortion. If one looks closely to the marker in Figure 9(c) it is obvious that the initially round marker is now *warped*, that is, its shape is not preserved under the transformation. If we extend this argument to the results shown in section 3.1.1, then it becomes clear that such distortion is also present there. The problem with the present approach is that it is not possible to accurately determine the *error* (or warping) because there is simply no relation (or mapping) between the real world coordinates and the 2D transformation.

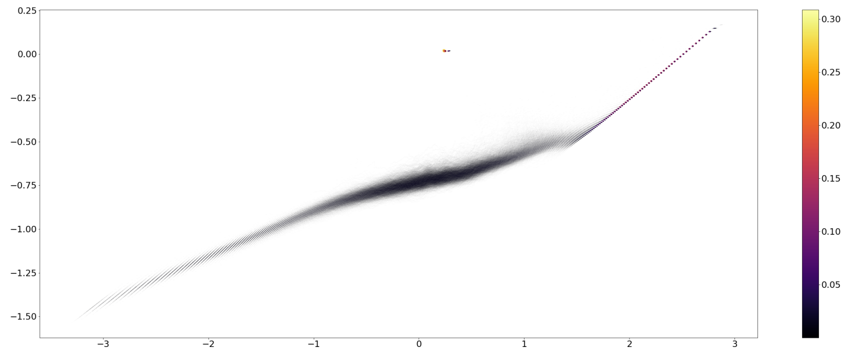
3.2 Water surface profile: LiDAR

For this particular study, samplings were taken for 10 minutes at a frequency of 0.33 Hz. Such arrangement produces a large amount of data, as it can be observed in the scatter plot depicted in Figure 10 with running water and without. Note that very often the LiDAR produces spurious output at around 0.3 m in the horizontal axis.

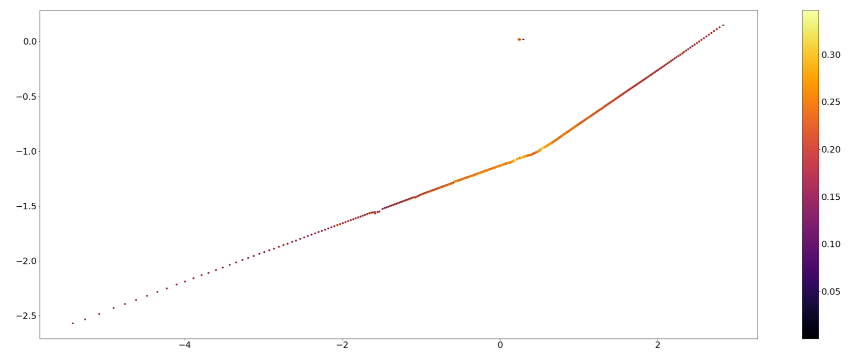
The outliers can be detected by calculating the local average of the neighbouring points and excluding values that are outside the 99.7 % band. The value set in place of the outlier is extrapolated using a first-order upwind polynomial, as explained in section 2.3. The same treatment is followed for invalid (NaN) readings. The *filtered* data is presented in Figure 11. Notice there that the flume's bottom coincides with the mean free-surface on the otherwise supercritical section of the flow. It is also important to mention that the data presents an important amount of scatter: one may attribute such to the false detection of water droplets and Mie scattering.



(a) Data readings at $Q = 100\,l/s$.

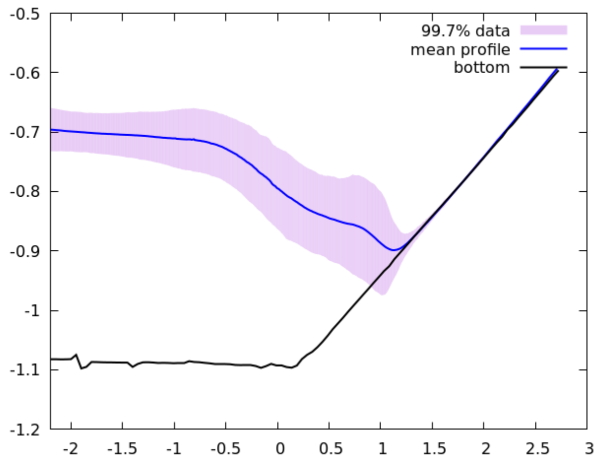


(b) Data readings at $Q = 150\,l/s$.

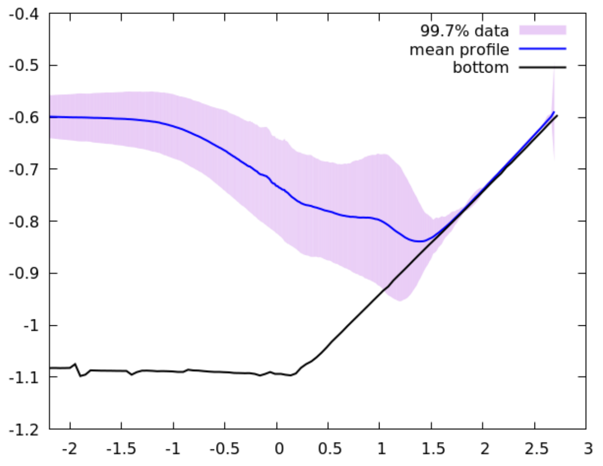


(c) Data readings when dry.

Figure 10 – Raw data transformed from polar to cartesian coordinates obtained from LiDAR for a period of 10 minutes, at different discharges. Notice that the slanting of the profile is due to the slanting of the sensor.



(a) Filtered data readings at $Q = 100\, lt/s$.



(b) Filtered data readings at $Q = 150\, lt/s$.

Figure 11 – Data obtained from LiDAR for a period of 10 minutes, at different conditions, and then filtered.

4 Conclusions

This report forms part of the project “On the accurate simulation of hydraulic jumps using OpenFOAM”, where CFD analyses and calibrations are conducted for the study of classical and sloped hydraulic jumps. Some inconveniences were found during PIV and LiDAR processing of the measurements obtained during the experimental campaigns. Much of these errors respond to the challenges in installing equipment in large-scale models and to the use of low-end equipment for such complex scenario. Once such challenges are overcome the results therein obtained will be used for comparison with CFD studies conducted under the framework of this project.

On the PIV side it was found that framerate plays a large role in the quality of the results, regarding the false detection of correlation peaks. A simple ‘back-of-the-envelope’ calculation, supported by the experiments themselves, confirmed that a minimum framerate of 100 Hz is needed to get somewhat reasonable displacement fields. Additionally, it was found that affine (2D-to-2D) transformation of images using georeferenced markers may not preserve the shape of the objects being photographed, this invalidating any displacement field that may be determined via PIV.

For LiDAR, it is clear that the water surface needs to be bubbly in order to be picked up by the sensor. The section upstream of the hydraulic jump is transparent to the LiDAR, hence is not seen on the measurements.

A repository with the codes herein developed, both for LiDAR and PIV, are available at:

`https://wl-subversion.vlaanderen.be/svn/repoSpNumMod/PIV`

4.1 Future work

Given the present challenges and questions that still remain open, the following routes are suggested for further research:

- A camera calibration method, using a PnP (Point-and-Perspective) methodology, is suggested for future use, using OpenCV. Said methodology may allow for a more accurate location of the air bubbles within the 2-D pixel plane of the images.
- Future campaigns must explore a way of aerating the flume on the slope. By having an accurate representation of the free surface it is possible to use basic relations of hydraulics which may allow to determine the character of the hydraulic jump.

References

Ohtsu, **I.; Yasuda**, **Y.** (1991). Hydraulic jump in sloping channels. *Journal of Hydraulic Engineering 117 (7)*: 905–921 pp.

Raffel, **M.; Willert**, **C. E.; Scarano**, **F.; Kähler**, **C. J.; Wereley**, **S. T.; Kompenhans**, **J.** (2018). Particle image velocimetry: a practical guide. Springer

Rouse, **H.** (1946). Elementary mechanics of fluids.

Vercruysse, **J.; Verelst**, **K.; Mostaert**, **F.** (2018). Sigmaplan –Energie dissipatie aan teen overloopdijk Gecontroleerde Overstromings Gebieden. 0.1. *WL Rapporten (in progress)*. Antwerpen, BE

Wolput, **B.** (2021). Instructie gebruik LMS511 2D LiDAR SICK. 1.0. *WL Memo (in Dutch)*. Antwerpen, BE

DEPARTMENT **MOBILITY & PUBLIC WORKS**
Flanders Hydraulics

Berchemlei 115, 2140 Antwerp
T +32 (0)3 224 60 35
F +32 (0)3 224 60 36
waterbouwkundiglabo@vlaanderen.be
www.flandershydraulics.be