Image Processing Algorithm

The image processing algorithm presented here was specifically designed for studying particulate flow in wavy channels. Figure 1, displayed below, originates from a Particle Image Velocimetry (PIV) setup, employing a dual head laser to produce two images (referred to as Frame A and Frame B) representing the same particle distribution instance. Consequently, both Frame A and Frame B capture the same particle with the exact relative position. However, a notable distinction exists between the two images: Frame A exhibits superior focus on particles within the valley region, whereas Frame B sharply captures particles in the peak region. Therefore, to obtain an accurate estimation of the particle concentration distribution across the channel width, both image frames were processed. As both frames capture the same particle at the same relative position, it can be concluded that one image instance consists of two images, namely Frame A and Frame B.

The study window, representing the image frames, covers both sinusoidal walls of the wavy channel. Its length is approximately half of the sinusoidal wavelength, capturing the peak region of one wall and the valley region of the other.

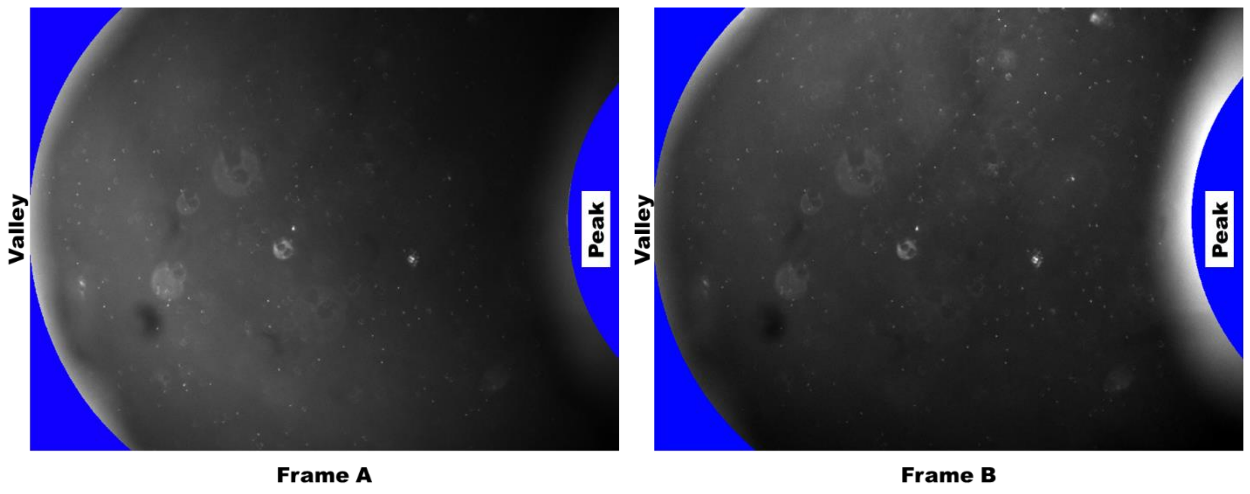
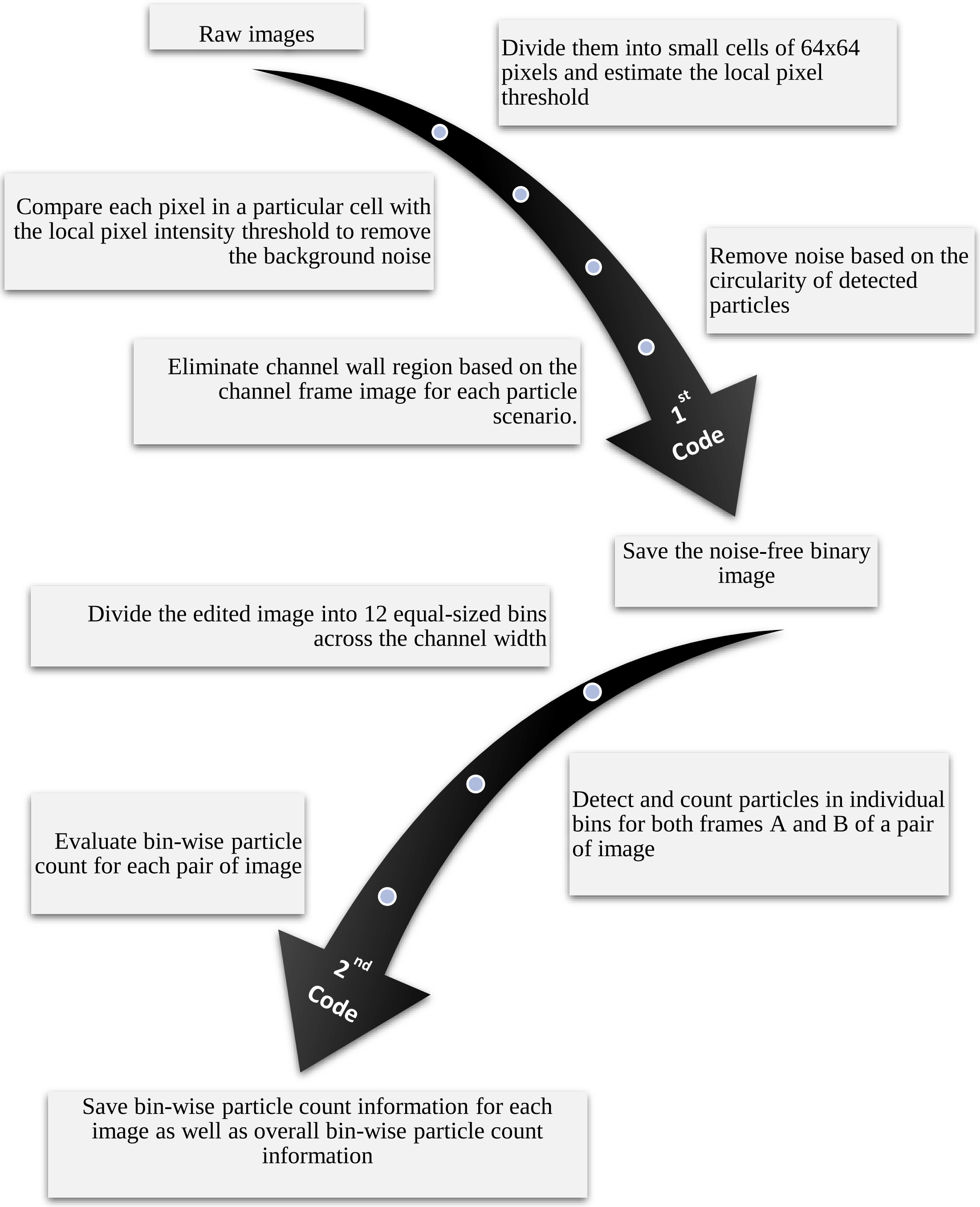


Figure 1. A pair of raw images (frame A on the left, frame B on the right).

To estimate the particle distribution across the channel width, the raw images acquired from the PIV setup underwent image processing using custom Python-based codes. To facilitate the image processing tasks, Python employs various libraries, including Pandas, NumPy, SciPy, and cv2. The codes were developed to process the raw images to detect, identify and count the particles and upload the data to a spreadsheet. The flow diagram of the image processing algorithm is shown in Figure 2.



Raw images
Divide them into small cells of 64x64 pixels and estimate the local pixel threshold
Compare each pixel in a particular cell with the local pixel intensity threshold to remove the background noise
Remove noise based on the circularity of detected particles
Eliminate channel wall region based on the channel frame image for each particle scenario.
1st Code
Save the noise-free binary image
Divide the edited image into 12 equal-sized bins across the channel width
Detect and count particles in individual bins for both frames A and B of a pair of image
Evaluate bin-wise particle count for each pair of image
2nd Code
Save bin-wise particle count information for each image as well as overall bin-wise particle count information

Figure 2. The flow chart of the image processing algorithm.

The image processing algorithm is divided into two parts with dedicated Python codes for each part. In the first part, the raw images are processed to eliminate noise, resulting in binary images free of noise. The second part utilizes these noise-free binary images to evaluate the particle distribution profile across the channel width.

The first Python code takes the raw images and divides each image into several smaller cells of size 64×64 pixels. Then the local average of pixel values is calculated for each cell. The local average of each cell is then multiplied by a multiplying factor determined through trial and error to get the local pixel intensity threshold. Then, each pixel in a cell is compared with the pixel intensity threshold specific to that cell. If a pixel's value is found to be lower than the cell's intensity threshold, it is set to zero (the lowest value on the gray scale). Otherwise, it is set to 255 (the highest value on the gray scale). This process generates a binary image with significant background noise removed, although some noise may still persist.

To further refine the binary image and remove additional noise, a circularity parameter is introduced. Since spherical particles were used in the experiment, most particles exhibit some degree of circularity despite pixel distortion caused by limitations of the experimental setup and image acquisition. Conversely, noise particles tend to lack circularity. Circularity (O) is defined as $\mathrm{O} = 4\pi A/P^2$, where A represents the area and P represents the perimeter of a particle. The appropriate circularity value is determined through a trial-and-error method. This circularity criterion aids in eliminating a significant amount of noise.

The wall region of the image, which does not contain any particles, is more prone to noise. To completely remove the wall section from the final binary image, the modified binary image (generated after circularity comparison) is compared with a manually generated channel frame image, specific to each particle scenario based on the raw images. Any pixels in the modified binary image corresponding to the wall in the channel frame image are eliminated, resulting in the final binary image with minimal noise.

Figure 3 displays a processed binary image alongside the corresponding raw image. It is evident that the processed binary image successfully preserves the majority of particles from the raw image while effectively removing most of the background noise.

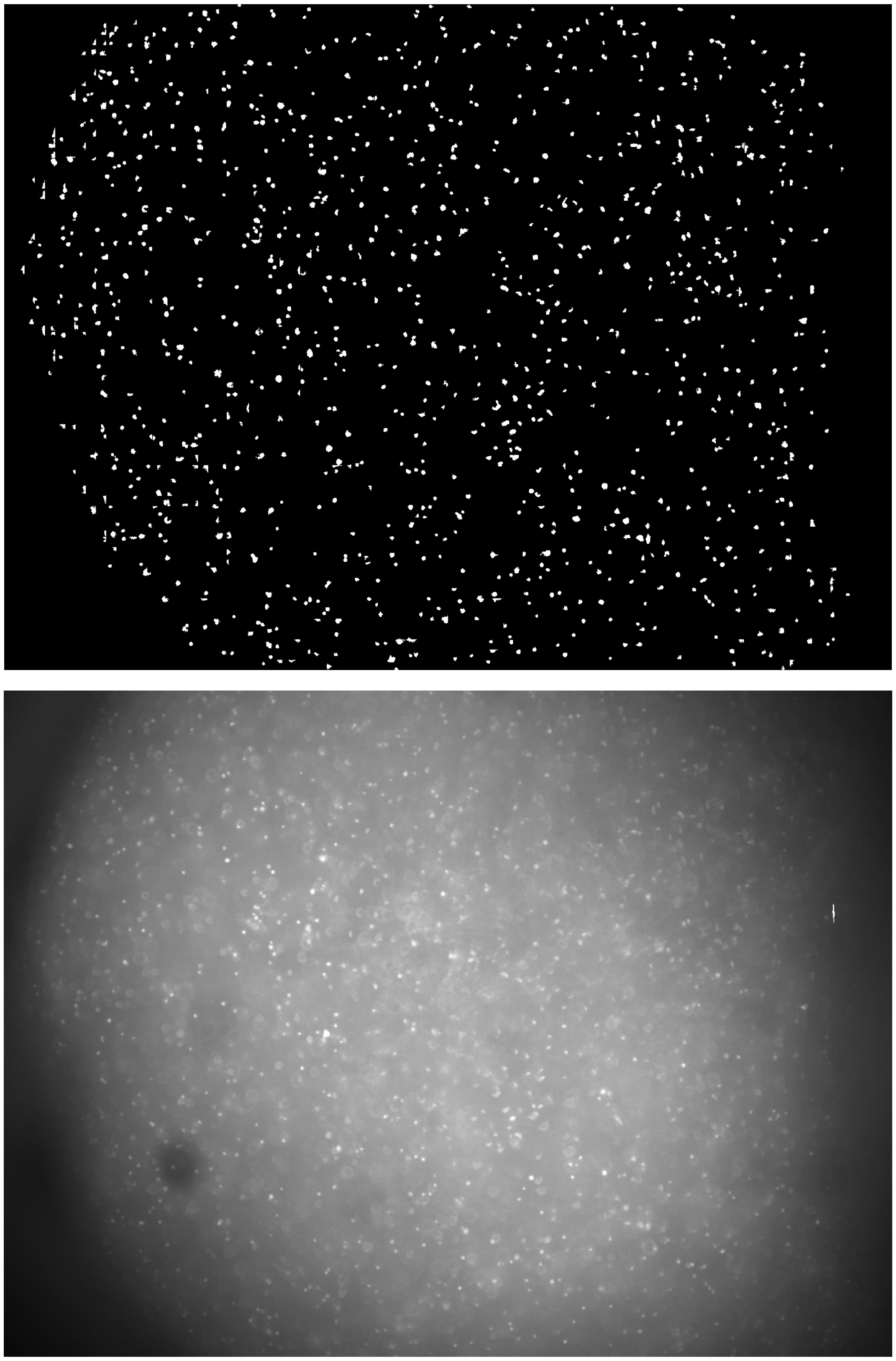
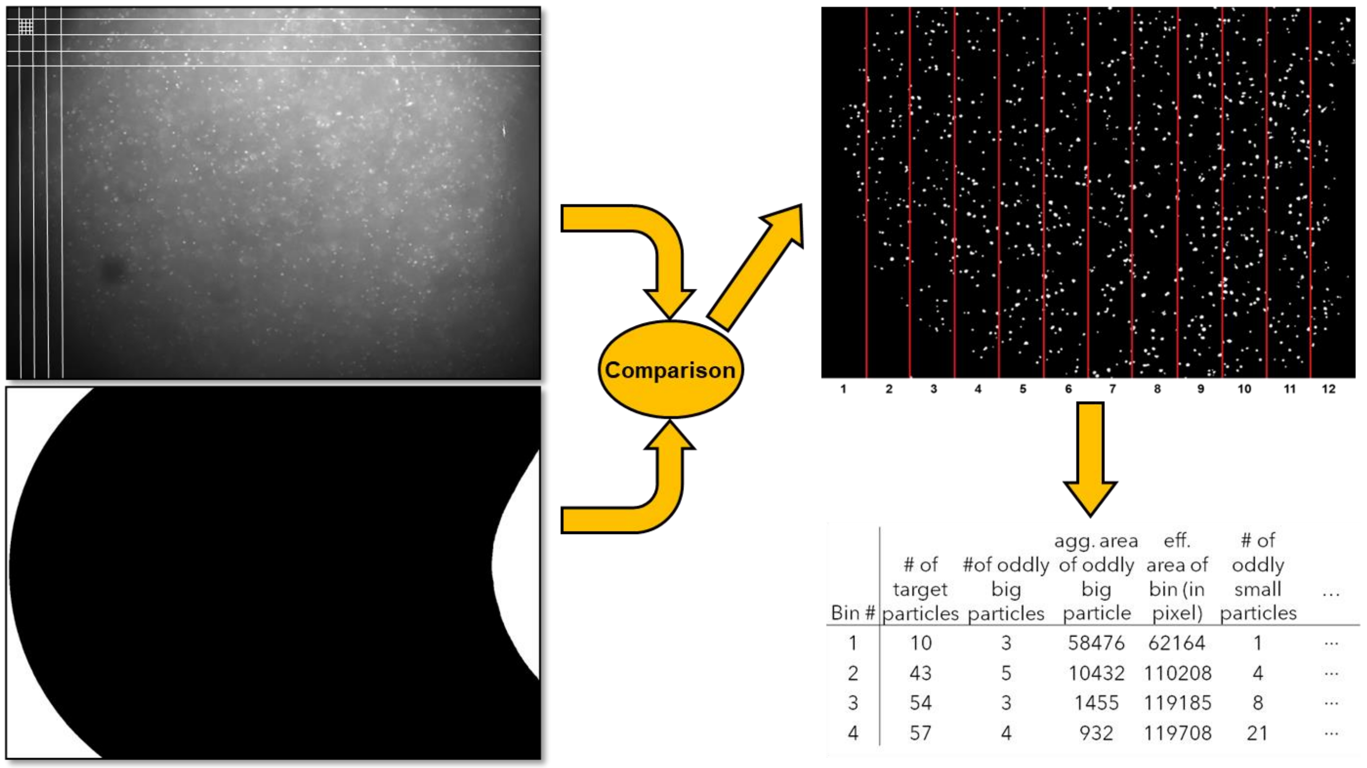


Figure 3. Illustrating a processed image (top) and corresponding raw image (bottom).

The final binary image serves as the input for the second Python code. In this second code, the binary images are divided into twelve equal-sized bins across the channel width. A particle is assigned to a specific bin if the centroid of the particle lies within that bin. To determine the particle count of a bin for a particular image instance, the particle counts of both images Frame A and Frame B, corresponding to that bin, are compared. The maximum value of particle count between Frame A and Frame B is considered as the final particle count for the particular bin in that image instance. For example, for an image instance, if the number of detected particles in bin number 3 (say) is bigger for image Frame A compared to image Frame B, the particle count of bin number 3 is based on image Frame A for that image instance.

Particle distribution information from each image instance is recorded in a spreadsheet. To obtain the overall particle distribution for a specific particle scenario in the channel, an average is calculated across all 300 image instances. The resulting overall particle distribution is then uploaded to a separate spreadsheet within the same Excel file and exported for further analysis. Figure 4 shows a preview of the image processing algorithm.



Bin #	# of target particles	#of oddly big particles	agg. area of oddly big particle	eff. area of bin (in pixel)	# of oddly small particles	...
1	10	3	58476	62164	1	...
2	43	5	10432	110208	4	...
3	54	3	1455	119185	8	...
4	57	4	932	119708	21	...

Figure 4. A preview of image processing algorithm.