

Development of a Particle Image Velocity Data Processing Algorithm

Course Project for MCG5138F00: Introduction to Lasers/Optical Diagnostics

Ayham AlAkhras
Department of Mechanical Engineering
University of Ottawa
Ottawa, Canada
300207406

Abstract—This report covers the development of a particle image velocity (PIV) data processing algorithm coded in MATLAB. A brief introduction to PIV measurements is given. The algorithm is then broken down into its main components: (1) image processing, (2) interrogation window calculation, (3) particle number calculation, and (4) velocity calculation. The results of the algorithm are then presented and interpreted.

Keywords—Particle image velocity, MATLAB, image processing, cross correlation

I. INTRODUCTION

Particle image velocity (PIV) data processing is a crucial field for measuring flow fields and their velocities. To obtain PIV images, a flow is illuminated with a laser light sheet. The fluid is seeded with small reflective particles, typically fine TiO_2 powder, that are assumed to move follow the fluid perfectly due to their size. A double pulse laser and camera are synchronized and record two images in quick succession. The use of a laser allows for high intensity and short pulses, leading to minimal distortion due to particle movement. These images are typically taken less than $100 \mu\text{s}$ apart.

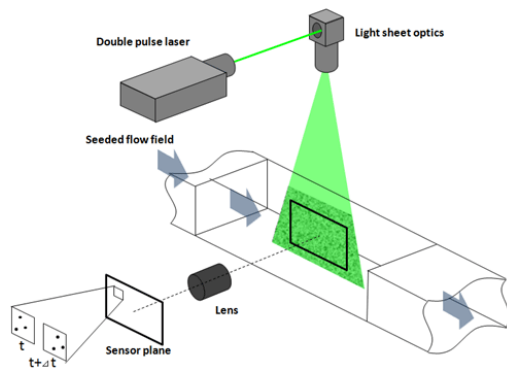


Figure 1: PIV setup [1]

Using these images, the particle velocity field can be found by dividing the image into smaller sections. These sections are called interrogation windows. Cross correlation is used to find the most likely displacement of each section of the image between two frames.

II. ALGORITHM

A. Image Processing

Once the images were imported and displayed, image processing was implemented to remove the background of the image and only include values greater than a brightness threshold. The background was removed using a user-defined function. The function made use of the *strel* function in MATLAB, which creates an object representing a flat morphological structuring element for each defined particle. A threshold filter is then defined and applied to remove all pixels with values below a threshold defined as half of the maximum intensity.

B. Interrogation Window Calculation

To obtain the interrogation window, two methods were used and compared. The first function, defined as *windowCalcCircle*, uses the image processing toolbox in MATLAB to locate circles in a black and white image. This method attempts to find bright (or dark) circles within the image within a range of radii in pixels. These circles are shown in red overlaid atop the processed image shown in Fig. 1.

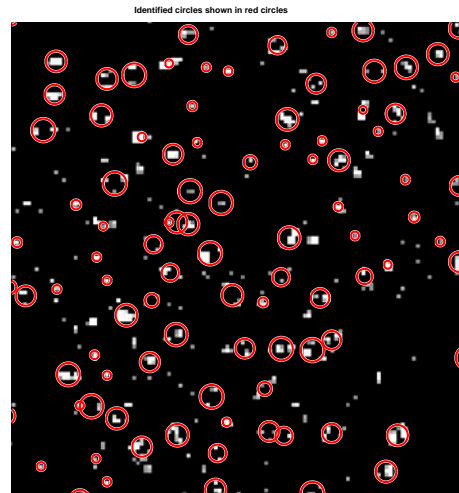


Figure 2: Circle detection

A major disadvantage of this method was the average radius of the particles. The MATLAB function, *imfindcircles*, issues a warning stating that algorithm accuracy is limited for radius values less than or equal to 5 pixels. The mean particle size was found by computing the average area of the circles found.

To combat this, another particle detection function, *windowCalcObject*, was developed. This algorithm utilizes the MATLAB function *bwconncomp*. This function locates pixels that are connected by the edges or by the edges and corners. This attribute is determined by the pixel connectivity as shown in Fig. 3. The function reports the connected pixels as objects.



conn — Pixel connectivity	
4 8 6 18 26 3-by-3-by-... -by-3 matrix of 0s and 1s	
Pixel connectivity, specified as one of the values in this table. The default connectivity is 8 for 2-D images, and 26 for 3-D images.	
Value	Meaning
Two-Dimensional Connectivities	
4-connected	Pixels are connected if their edges touch. Two adjoining pixels are part of the same object if they are both on and are connected along the horizontal or vertical direction. 
8-connected	Pixels are connected if their edges or corners touch. Two adjoining pixels are part of the same object if they are both on and are connected along the horizontal, vertical, or diagonal direction. 

Figure 3: Pixel connectivity [2]

The algorithm reported the particles as randomly colored pixels on a blue background. Using the obtained objects, a labeled matrix can be formed to obtain the mean particle size. This method reported better results from visual observation and was thus chosen for the algorithm.

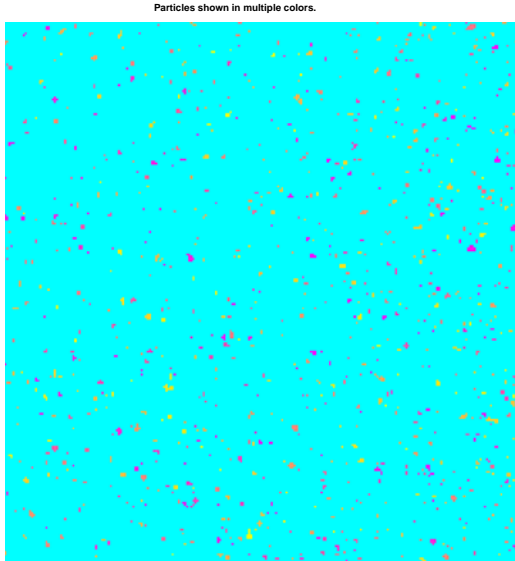


Figure 4: Particle detection

The particle density in the image is defined as:

$$\text{particle density} = \frac{\text{mean particle size} \times \# \text{ of particles}}{\text{total area}}. \quad (1)$$

Using the particle density, the area in which 10 particles exist on average throughout the image is found. This is known as the

interrogation window area, which can be used to obtain the square interrogation window side length.

C. Particle Number Calculation

A loop is then iterated to go through the image in increments equal to the interrogation window length in each direction. The user-defined function, *windowCalcObject*, was used to obtain the number of particles for each window of the image. A heatmap was then generated to intuitively represent the data.

D. Velocity Field Calculation

Lastly, another loop was made to iterate over the entire image in window length increments to obtain the velocity at each window. This loop made use of *normxcorr2* function in MATLAB to obtain the cross correlation at each window. The function was used on the original images to reduce errors. The computations in this loop were done to account for the shifting done by the cross-correlation algorithm to a local coordinate system separate from the one used by MATLAB's image processing and the shifting window in the reference image. Instead of searching the whole second frame for a match, the algorithm instead attempted to find cross correlation within ± 2 interrogation window lengths. This made the loop significantly faster and prevented false positives as the flow field does not change drastically between frames; therefore, the assumption that the interrogation window does not shift more than two interrogation window lengths at a time is sound.

III. RESULTS

All results were generated in MATLAB.

A. Reading Images, Maximum Light Intensity, and Mean Particle Size

Both images were displayed in MATLAB using the *imshow* function.

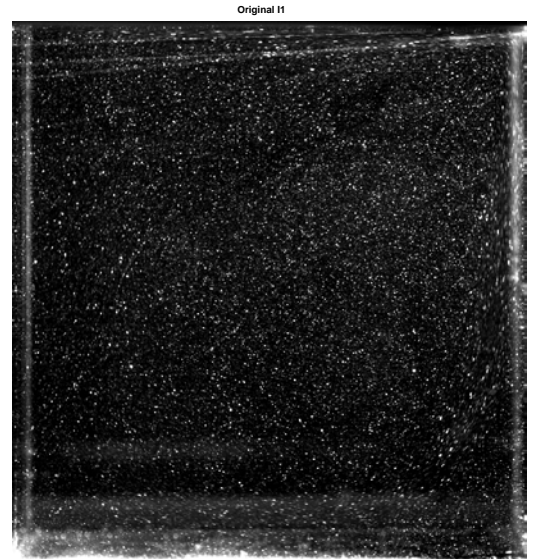


Figure 5: First frame

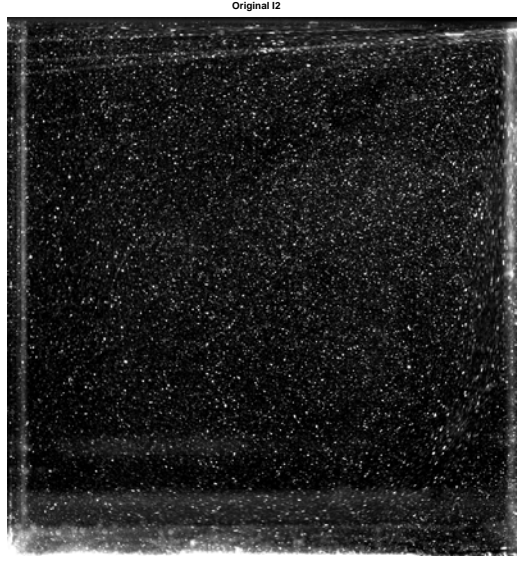


Figure 6: Second frame

Maximum light intensity was found to be 255 units, and the mean particle size with the used particle detection method was found to be 2.5 pixels per particle.

B. Interrogation Window Length

Interrogation window length was found to be 32x32 pixels, exactly the same as the recommended window length in literature [1].

C. Particle Count per Interrogation Window

Particle count per interrogation window was generated as a heatmap shown in Fig. 5.

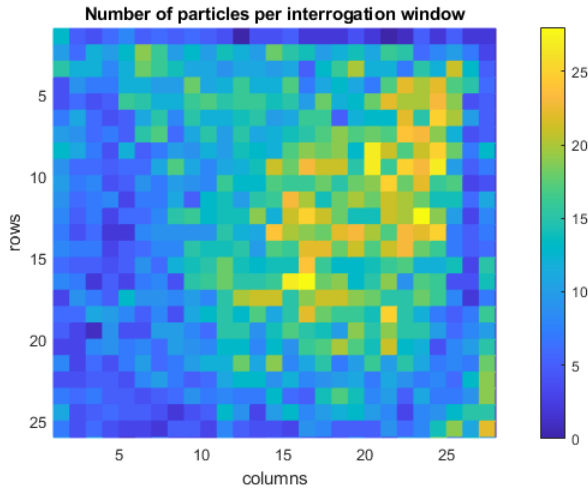


Figure 7: Particles per interrogation window

D. Correlation Plane

The correlation plane for the center interrogation window is shown in Fig. 6. The maximum correlation value is shown in yellow.

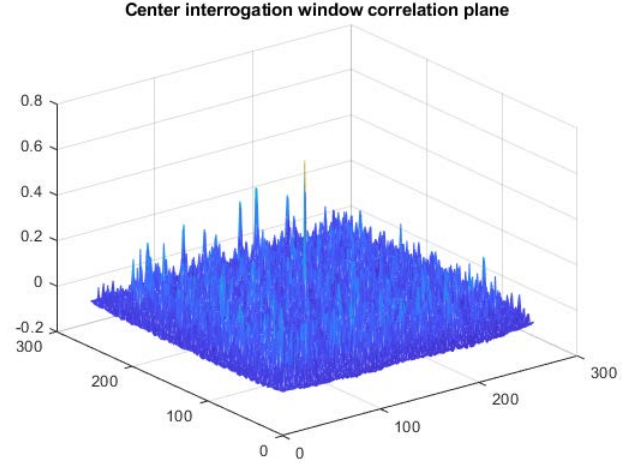


Figure 8: Correlation plane

E. Displacement Vector Field

The displacement vector field was plotted based on the maximum cross-correlation data obtained.

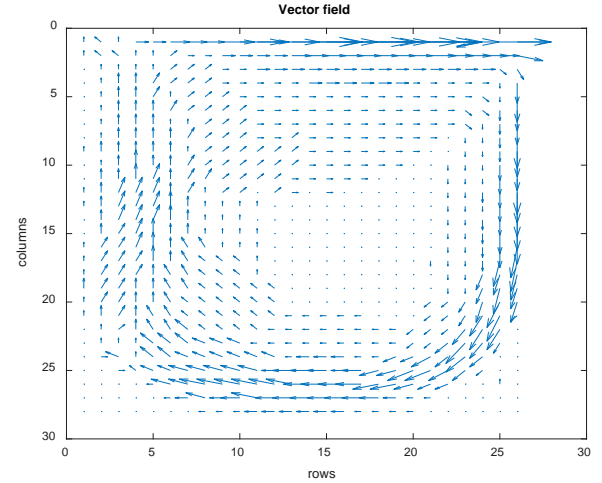


Figure 9: Velocity vector field

IV. DISCUSSION

The mean particle size, 2.5 pixels, was a reasonable value upon visual inspection of the particles in the image. Maximum light intensity, 255, allowed for effective filtering. Most particles appeared to be around 1 to 5 pixels in area. The interrogation window length was also accurate, as it exactly matches the 32x32 pixel recommendation in literature [1]. The particle count per interrogation window showed the variation in the flow field, as some areas had up to 28 particles while others had close to none. The correlation plane results were as expected, the images do not shift a large amount and there is a clear peak in the correlation. Lastly, the displacement vector

field was accurate when compared to the original images. When cycling through the images, a flow similar to the one depicted in *Fig. 9* can be seen. This flow appears to be a vortex with a stagnation at the center. All in all, the algorithm works well for the given settings

V. CONCLUSION

The development of this code was successful, and the results achieved were accurate. All measurements were reasonable and the final flow field matched visual analysis of the flow field.

However, the settings within the algorithm are highly delicate, and modification of these values for different situations is highly recommended.

REFERENCES

- [1] Z. Hong, "Particle Image Velocimetry (PIV) Data Processing Project Instructions," University of Ottawa, 2021.
- [2] MATLAB official documentation.