COMPARATIVE ANALYSIS OF OPTICAL FLOW MEASUREMENT TECHNIQUE

AMAR NAZMI ZULKARNAIN B NIZAM AKBAR

MECHANICAL ENGINEERING

UNIVERSITI TEKNOLOGI PETRONAS

JANUARY 2017

Comparative Analysis Of Optical Flow Measurement Technique

by

Amar Nazmi Zulkarnain B Nizam Akbar
18521

Dissertation submitted in partial fulfillment of

The requirement for the

Bachelor of Engineering (Hons)

(Mechanical)

JANUARY 2017

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
32610 Tronoh
Perak Darul Ridzuan
Malaysia

CERTIFICATION OF APPROVAL

Comparative Analysis Of Optical Flow Measurement Technique

by

Amar Nazmi Zulkarnain B Nizam Akbar
18521

A project dissertation submitted to the
Mechanical Engineering Programme
Universiti Teknologi PETRONAS
In partial fulfillment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(MECHANICAL)

Approved by,

_____

(Dr Mark Ovinis)

UNIVERSITI TEKNOLOGI PETRONAS

BANDAR SERI ISKANDAR, PERAK

January 2017

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

_____

AMAR NAZMI ZULKARNAIN BIN NIZAM AKBAR

ABSTRACT

In 2010, an explosion occurred in the Macondo well in the Gulf of Mexico (GoM) that caused the oil rig to sink to the bottom of the ocean. At that time, there was no accurate method to estimate the flow rate of the hydrocarbon leakage. Timothy J. Crone used a noninvasive flow measurement technique, optical plume velocimetry to determine the flow rate using image analysis. There are several methods to estimate the flow rate measurement namely temporal cross correlation, frequency domain cross correlation, average square difference function, ASDF and average magnitude difference function, AMDF. The most important parameter in these methods is the time delay. However, Crone neglected the frequency domain cross correlation which may give a better flow rate estimation. The main motivation of this research is to do a comparative analysis of the four cross correlation algorithm as mentioned above. The four algorithms were used to estimate the flow rate where five simulations with five different flow rates were measured in order to determine the accuracy of each algorithm. Crone's experimental setup was replicated and a DSLR camera was used to record the flow. The mean flow velocities across the jet nozzle for simulation 1, simulation 2, simulation 3, simulation 4 and simulation 5 were 0.13m/s, 0.24m/s, 0.31m/s, 0.33m/s and 0.41m/s respectively. Of the four algorithms considered, the temporal cross correlation algorithm was the most accurate, with maximum percentage error of 25%. However, all methods still produced a non-zero crossing relationship between the estimated velocity and the actual velocity across the jet nozzle of -0.01 m/s, - 0.02 m/s, - 0.03 m/s and -0.06 m/s for temporal cross correlation, frequency domain cross correlation, average square difference function, ASDF and average magnitude difference function, AMDF respectively. The percentage error for the frequency domain cross correlation is lower compared to ASDF and AMDF, however the percentage error is quite high due to its inability to search for an optimal match of fluid flow images produced because of the dynamics of the fluid itself. AMDF method has the highest percentage error because this method is influenced by intensity variation of the images and background noise. Both ASDF and AMDF have limitations in terms of producing the accurate delay range. The temporal cross correlation algorithm was the best algorithm among the four algorithm used in estimating the flow rate.

ACKNOWLEDGEMENTS

My praises to The Almighty, Allah. I finished my final year project with his grace and mercy. I am very grateful to everybody who have directly or indirectly contributed to the completion of my research.

I would also like to express my deepest appreciation to my supervisor, Dr Mark Ovinis for giving me the opportunity to do my project under his supervision. Thank you very much to Mr Osman for his time and guidance. I will not be able to complete this project successfully without their guidance.

Besides that, I would like to convey my gratitude to both coordinators for the final year project, Dr Turnad and Dr Tamiru for providing the guidelines to complete the project successfully.

A million thanks to my family and all my friends who gave me the support to complete this project.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ASDF	Average Square Difference Function
AMDF	Average Magnitude Difference Function

CHAPTER 1: INTRODUCTION

1.1 Background

In 2010, an explosion occurred in the Macondo well in the Gulf of Mexico (GoM) that caused the oil rig to sink to the bottom of the ocean [1]. The disaster caused an oil leak as shown in Figure 1.1 and caused massive pollution to the marine environment. Usually a penalty is imposed on the company based on the amount of oil spilled into the ocean. In such disasters, determining the volume of oil spilled is important [2] for proper design of oil well interventions, calculation of the amount of dispersant that is required to reduce the risk of oil on the ocean surface and to know the amount of remaining oil in the reservoir.

Figure 1.1: Oil leak

At that time, there was no proven method for directly measuring the hydrocarbon discharge at the given temperature and pressure. There are two techniques to measure the flow, either using intrusive flow measurement technique such as flow meter or non-intrusive flow measurement technique such as Particle Image Velocimetry (PIV) or Optical Plume Velocimetry (OPV). Mcnutt et. al. [1] stated that intrusive measurement technique will fail due to icing caused by hydrates. Inserting a probe or sensor into a flow where the condition is high in pressure and temperature may damage the probe or sensor. Some of the examples of established flow measurement techniques are Laser Dopler technique, PIV and OPV [3]. Crone et. al. work on OPV and was able to yield an accurate flow estimates for the Deep Water Horizon oil spill incident. The technique was able to estimate the flow rate of a fluid flow

optically using temporal cross correlation technique. However the accuracy of the estimation differs depending on the choice of algorithm [4]. Time delay is one of the very important parameter in estimating the flow rate. Currently there are many methods to estimate the time delay such as temporal cross correlation, frequency domain cross correlation, average square difference function, ASDF and average magnitude difference function, AMDF. According to Jacovitti and Scarano, both the ASDF and AMDF based estimator outperform the direct cross correlation based estimator [5]. However, according to Aiordachioaie and Nicolau direct cross correlation method was more accurate compared to ASDF and AMDF. Besides, in Crone's work he only considers temporal cross correlation and neglect the frequency domain cross correlation which might give a better result. In this work, a comparative analysis of optical flow algorithm based on four different techniques which were temporal cross correlation, frequency domain cross correlation, average square difference function, ASDF and average magnitude difference function, AMDF were performed.

1.2 Problem Statement

In Crone's work, only temporal cross correlation was considered and frequency domain cross correlation neglected, which may give a better flow rate estimation. The omission of frequency domain cross correlation to estimate the flow rate and the contradiction of the findings of the research done by Aiordachioaie et al. and Jacovitti et al. were the main motivation of this research.

1.3 Objective

The objectives of this project are:

1. To apply and compare the accuracy of the four algorithms which are temporal cross correlation, frequency domain cross correlation, average square difference function, ASDF and average magnitude difference function, AMDF for different flow rates.

1.4 Scope Of Study

The scopes of the study are:

a) To estimate the flow rate from five simulations where the nozzle mean flow velocities across jet nozzle ranges from 0.13m/s to 0.41m/s. When there is oil leakage underwater, a jet flow will be formed. The jet flow will be formed when the nozzle mean flow velocities across jet nozzle ranges from 0.13m/s to 0.41m/s. Thus this range of nozzle mean flow velocities across jet nozzle were implemented in order to simulate the real oil leak.

b) To apply the four algorithm techniques which include:
   - Temporal cross correlation
   - Frequency domain cross correlation
   - Average square difference function, ASDF
   - Average magnitude difference function, AMDF

CHAPTER 2: LITERATURE REVIEW

2.1 Optical Flow Measurement

There are several optical flow measurement techniques which include Particle Image Velocimetry, PIV and Optical Plume Velocimetry, OPV. PIV measure instantaneous flow fields by recording images of suspended seed particle in the flow at successive instants of time. In PIV fluid velocity information at an "interrogation region" is obtained from many tracer particles and it is taken as the most probable statistical value [6]. Figure 2.1 shows the principal layout of the PIV system.

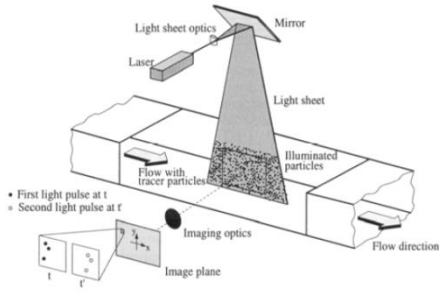


Figure 2.1: Principal layout of the PIV system

Optical plume velocimetry, OPV method is developed by Crone et al [3] and it is suitable for obtaining flow rate of fluid flow of plume shape as shown in Figure 2.2.

Figure 2.2: Fluid flow of plume shape

OPV is an optical flow technique using intensity data in the time direction [3]. Even though PIV was often used to calculate the image velocity field but it yields velocities that are significantly lower than expected when the fluid flow is plume shape [7].

2.2 Image Velocity Field

The image velocity field is a two dimensional vector field describing the motion of objects imaged within a video sequence [3]. There are two parameters that should be taken care of in order to obtain a high accuracy of image velocity field which includes distance of pixel separation and also time delay. The formula for image velocity field is shown in equation 1 in the appendix. The Figure 2.3 below shows the distance of pixel separation.

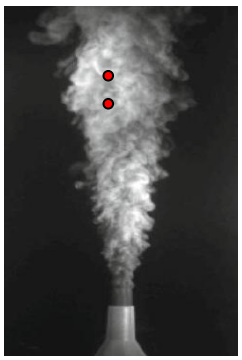


Figure 2.3: The distance between the two red dots represents the distance of pixel separation.

There are several methods to find the time delay. One of them is by using cross correlation method. Crone et al [7] used temporal cross correlation to find the time delay. Interpolated temporal cross correlation function of image intensity is calculated across the entire region of interest for pixel pairs separated by some distance vertically. The lag value corresponding to the maximum of each cross correlation function defines the time delay (in frames) required for flow features to transverse the distance defined by the pixel separation. Thus the image velocity (in pixel/frame) can be calculated at every pixel within the region of interest [7]. The region of interest for image velocity calculation is near the nozzle area because the mean velocity estimation everywhere in a fully developed pure jet which is at the nozzle area is proportional to actual flow velocity across jet nozzle which enables the mean velocity estimation being validated [3]. The current algorithm which is temporal cross correlation that has been used by Crone et al. will result in less accuracy of flow rate estimation when there is inclusion of image-velocity estimates from regions of transitional flow [3]. Thus other method will be implemented to estimate the image velocity with higher accuracy.

2.3 Percentage Error of Various Optical Techniques in the Measurement of Fluid Flow Rate

Table 2.1: Percentage error of various optical techniques in the measurement of fluid flow rate.

Techniques	Algorithm	Percentage Error
Large Eddy Tracking	Visual observation	20.3%
Particle image velocimetry, PIV	Spatial cross correlation	44.5%
Feature tracking velocimetry, FTV	Visual observation	47%
Digital Image velocimetry	Spatial cross correlation	7.5%

Table 2.1 shows the percentage error of various optical techniques in the measurement of fluid flow rate. All of the techniques used sequences of images as input signal and have high percentage error. Since the input are in the form of images it is very unlikely that accurate result can be obtained [8].

2.4 Frequency Domain Cross Correlation

The frequency domain cross correlation technique has been used in the medical field to determine the blood flow rate and velocity [9,10]. A two dimensional fast Fourier transforms (FFT) was performed on the image enclosed within the region of interest to convert the image from spatial domain to frequency domain. An inverse FFT was performed on the modified spectrum to obtain an image in the spatial domain that contained only high-frequency signal which corresponds to fast changes in the gray level such as edges.

2.5 Average Square Difference Function, ASDF and Average Magnitude Difference Function, AMDF

ASDF and AMDF methods also have been applied in a variety of applications such as in acoustics and radar communication to estimate the time delay between signals received at two spatially separated microphones [11]. Both algorithms are based on minimum error by seeking the position of the minimum difference between two signals. In these algorithms, the delay estimation process is reduced to a filter delay that gives minimal error. The ASDF method is based on finding the position of the minimum error square between two received signals and considering this position value as the estimated time delay. The AMDF method does not involve

multiplication and it is very useful in those applications where low computational complexity (fast estimation) is required [5].

Among the four cross correlation techniques which are temporal cross correlation, frequency domain cross correlation, average square difference function, ASDF and average magnitude difference function, AMDF , the frequency domain cross correlation is expected to have highest accuracy in estimating the image velocity. By performing Fourier transform on the image, the signal can be represented in another way and the signal can be decomposed into a series of constituent trigonometric functions. Thus we are able to see, measure and modify the signal in a different way which might not be possible if it is in spatial domain. In a fluid flow of plume shape, most regions are edges which are not smooth. Therefore, the frequency domain cross correlation is suitable since an inverse FFT was performed on the modified spectrum to obtain an image in the spatial domain that contained only high-frequency signal which corresponds to fast changes in the gray level such as the edges.

CHAPTER 3: METHODOLOGY/PROJECT WORK

The Figure 3.1 below shows the conceptual flow diagram of the stages of image based jet flow measurement.

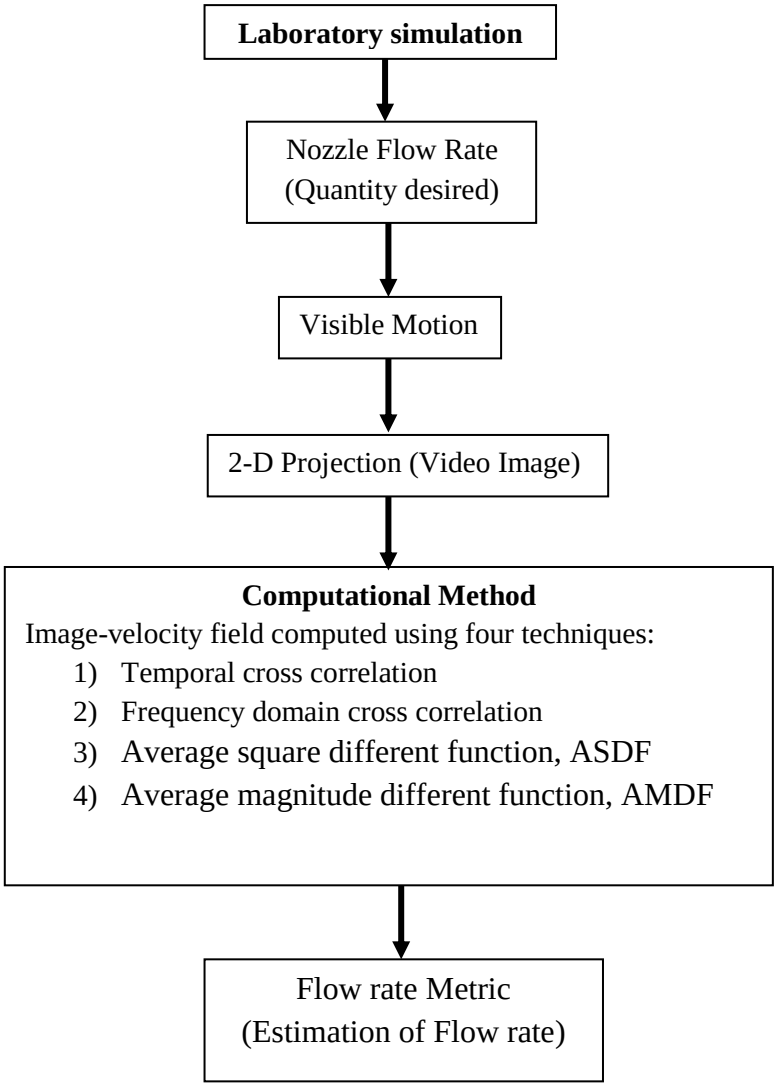


Figure 3.1: The stages of image based jet flow measurement.

In the Figure 3.1 the arrows represent the various physical, optical and computational transforms between the stages.

3.1 Laboratory Simulation

3.1.1 Nozzle flow rate

In this project, there will be five simulations with five different flow rates to investigate the accuracy of each algorithm in estimating flow rate. The mean flow velocities across jet nozzle for simulation 1, simulation 2, simulation 3, simulation 4 and simulation 5 were 0.13m/s, 0.24m/s, 0.31m/s, 0.33m/s and 0.41m/s respectively. The diameter of the nozzle used was 10 mm to create pressurized flow in order to obtain jet flow.

3.1.2 Visible Motion

In order to capture the visible motion, starch and black ink solution was used to simulate the hydrocarbon fluid. Starch solution is used to provide buoyancy flux.

3.1.3 Two-dimensional Projection Video Image

At this stage the laboratory apparatus will be constructed to simulate the flow of fluid with known flow rates. The digital single-lens reflex camera, DSLR will be used to capture the image sequences that contain two-dimensional projection of the three-dimensional flow field for the analysis. The simulated fluid motion was recorded for 20 seconds. The DSLR was able to record a frame rate of 50 fps for a total of 1000 frames with a resolution of 1280 x 720 pixels. Figure 3.2 shows the Crone's apparatus setup.

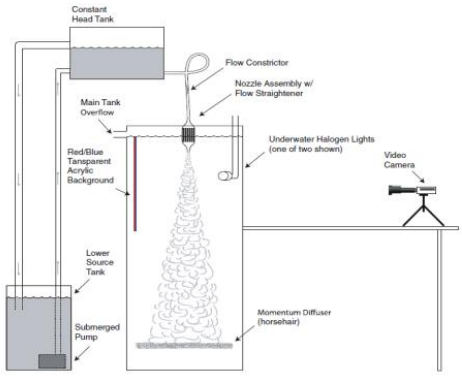


Figure 3.2: Crone's apparatus setup

The experiment set up consisted of three tanks; the lower source tank (600mm x 298mm x305 mm) which was filled with a mixture of starch, black ink and water,

the constant head tank (600mm x298mm x305 mm) and the main tank (400mm x300mm x600mm) which was filled with tap water until the nozzle was submerged in the water. A submersible pump was used to supply the mixture from the lower source tank to the constant head tank. Figure 3.3 shows the experiment set up which has been fabricated.

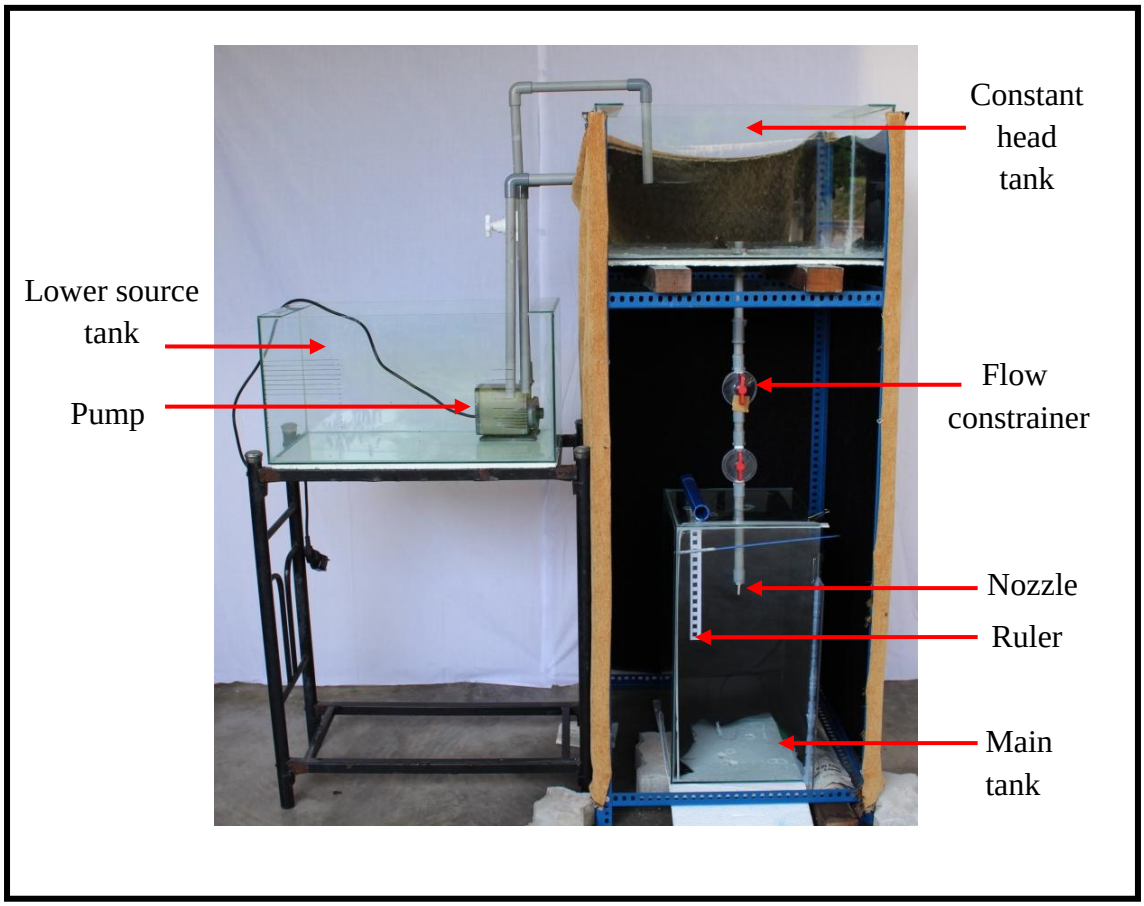


Figure 3.3: The experiment set up

By opening the flow constrainer, the simulated fluid flows from the constant head tank through the 10mm diameter nozzle to the main tank. The experiment was repeated 5 times for each flow rate. A ruler with alternating black and white squares of 0.01m x 0.01m on it was placed beside the nozzle as an indicator to determine the relationship between the pixel and the length. It was determined that twenty pixels

were equivalent to one centimeter. The raw data produced from the video were RGB (RedGreenBlue) images. The image for simulation 2 was shown in Figure 3.4.

Figure 3.4: An RGB image of the flow for simulation 2 extracted from a video recording.

Then the image was cropped to the size of 200 pixel x 250 pixel of the jet region. The cropped image was shown in Figure 3.5.

Figure 3.5: Cropped sample of RGB image

Once the video has been converted into sequences of images and being cropped, it was then converted into the grayscale. Then the sequences of images were converted into three dimensional pixel intensity matrix to be used in the computational step.

3.2 Computational Method

3.2.1 Image Velocity Field

The image velocity field is a two dimensional vector field describing the motion of object imaged within a video sequence. In this work, four algorithms which are temporal cross correlation, frequency domain cross correlation, average square difference function, ASDF and average magnitude difference function, AMDF will be applied to calculate the time delay. Once the time delay has been calculated the image velocity field can be obtained by using equation 1 in the appendix section. The pixel separation is 5 pixels so that the time delay is as large as possible without significantly reducing the correlation coefficient. The theories for each technique are as shown in the appendix.

3.2.2 Flow Rate Metric

To calculate the flow rate metric, all the values in the image velocity field that fell within a set region of the image will be averaged. In Crone's experiment, the set region covered the area of the image where the distance is 0.15m from the nozzle opening and he used 18 mm nozzle diameter. Thus the ratio of the distance from nozzle opening to nozzle diameter is about 8. Since in this experiment, the size of the nozzle used was 10mm, therefore the set region covered the area of the image where the distance is 0.08m from the nozzle opening. Then the mean time averaged image velocity (where the unit is in pixel/frame) will be converted into a scalar metric (where the unit is m^3/s) using equation 2 and equation 3 in the appendix section which then can be compared to the nozzle flow rate.

3.3 GANTT CHART AND KEY MILESTONE

FYP I (September 2016)

Project activities/ Week	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Searching information about optical flow														
Experiment set up								X						
Application of the algorithm											X			
Testing the algorithm using Crone's video												X		
Analyze the result and compare the accuracy of flow rate estimation														X

FYP II (January 2017)

Project activities/ Week	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Video data collection for different flow rate		X												
Testing the algorithm with different flow rate									X					
Analyze the data and compare the accuracy of flow rate estimation														X

X- Key Milestone

Figure 3.6: Gantt chart and key milestone

CHAPTER 4: RESULT AND DISCUSSION

4.1 The Measured Flow Rate from Experimental Work

Table 4.1 shows the measured flow rates of simulated fluid for the five cases of flow constrainer opening. T_i is the time taken for the simulated fluid in the lower source tank to be displaced by 2cm.

Table 4.1: Measured flow rates for each case of the flow rate

Simulation	Time, T_i (s)	Average Time, (s)	Volumetric Flow rate (m^3/s)	Flow velocity (m/s)	Image-velocity, Velocimetry (pixel/frame)
1	344	345	1.03×10^{-5}	0.13	5.2
	346				
	343				
2	183	185	1.93×10^{-5}	0.24	9.6
	185				
	184				
3	147	146	2.44×10^{-5}	0.31	12.4
	143				
	148				
4	133	135	2.64×10^{-5}	0.33	13.2
	137				
	136				
5	108	110	3.25×10^{-5}	0.41	16.4
	111				
	112				

4.2 Temporal Cross Correlation

4.2.1 Image-velocity Field

Figure 4.1 shows a set of typical 20-s time averaged image velocity field for the five simulations, 1 to 5, computed using the temporal cross correlation technique. It can be seen that the velocity of fluid flow near the nozzle was very high which was as expected. As the nozzle flow rate increases, the image velocity estimation also increases. The image velocity field reduces as it gets further away from the nozzle.

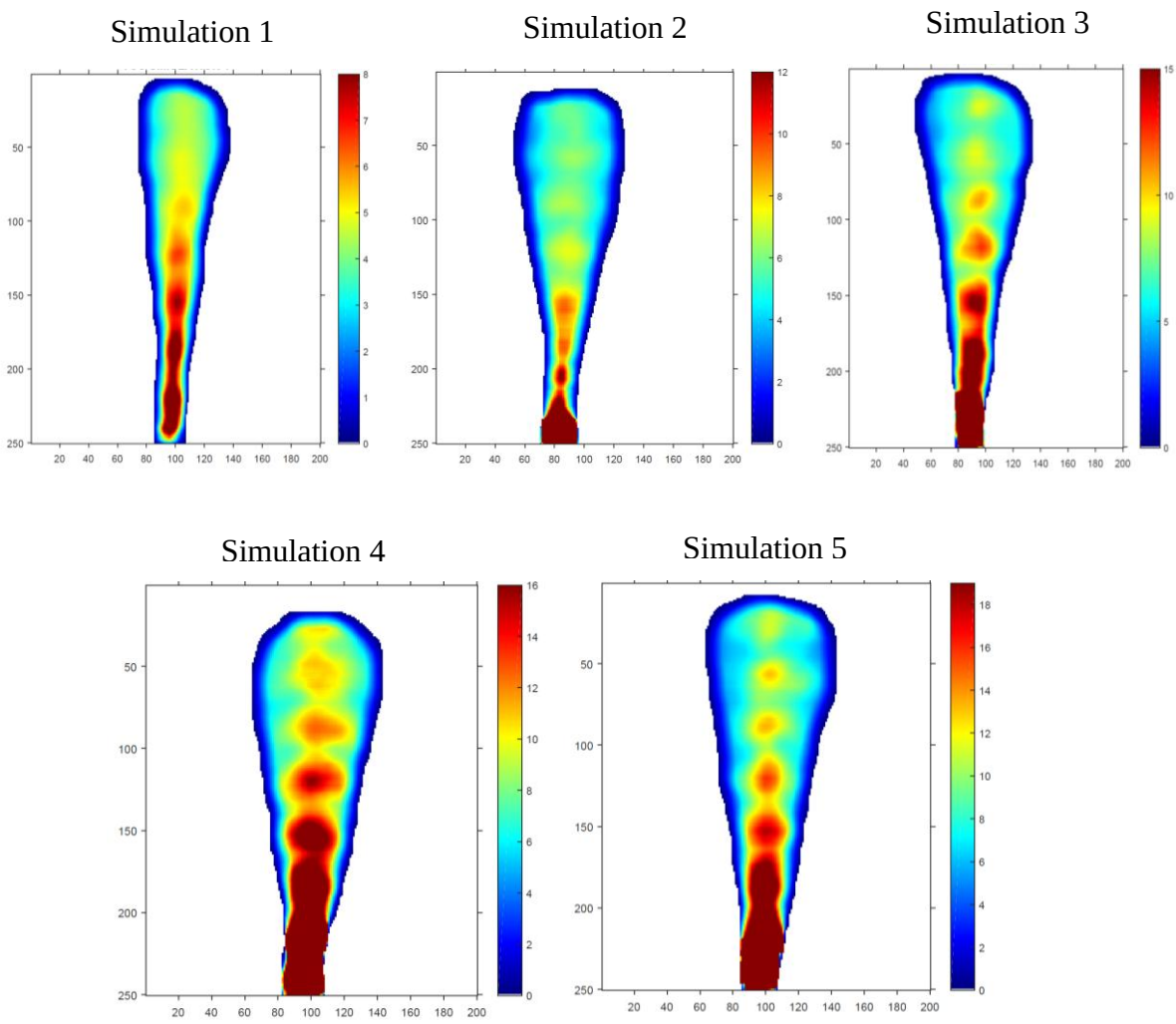


Figure 4.1: Typical 20-s time averaged image velocity field computed using the temporal cross correlation technique

4.2.2 Flow Rate Metric

Table 4.2 shows the estimated flow rate metric of the simulated fluid for the five simulations, 1 to 5, computed using the temporal cross correlation technique. As can be seen in the table the percentage error for each simulation especially for simulation 2, and simulation 3 was quite high but the mean velocity value are still close to the expected result. The maximum error was 25%.

Table 4.2: Estimated flow rate metric of the simulated fluid computed using the temporal cross correlation technique.

Simulation	**Mean image velocity (pixel/ frame)**	**Estimated velocity (m/s)**	**Estimated Flow rate Metric (m^3/s)**	**Percentage error (%)**
1	5.35	0.14	1.05×10^{-5}	7
2	7.36	0.20	1.44×10^{-5}	16
3	9.22	0.23	1.81×10^{-5}	25
4	12.96	0.32	2.54×10^{-5}	3
5	14.53	0.40	2.85×10^{-5}	2

This method is very suitable for input images that have an overall high brightness and high contrast, otherwise this method will lead to undesirable side effect such as reduced contrast and some noise can be easily introduced into the fused image, which will reduce the resultant image quality. Consequently the percentage error will be high [12].

Figure 4.2 shows the values of the estimated velocity plotted against the actual velocity for the temporal cross correlation technique. Also plotted is linear least square regression. Over the range of the flow rate, the relationship of estimated velocity to actual velocity is linear. The non-zero-crossing regression intersects the horizontal axis at the actual velocity of - 0.01 m/s. This intercept value will be used as a measure of the relative bias imparted by each technique, where a larger negative intercept suggests a greater bias.

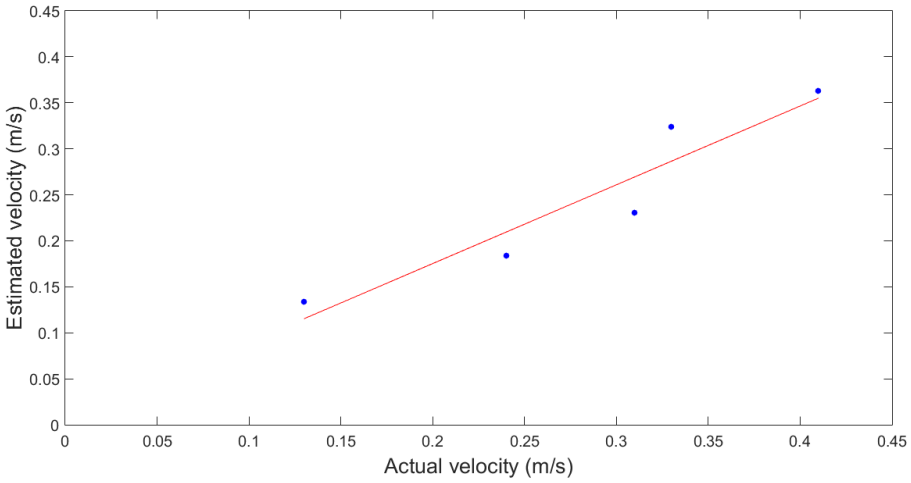


Figure 4.2: Estimated velocity from the five simulations compared to actual velocity for the temporal cross correlation technique

4.3 Frequency Domain Cross Correlation

4.3.1 Image-velocity Field

Figure 4.3 shows a set of typical 20-s time averaged image velocity field for the five simulations, 1 to 5, computed using the frequency domain cross correlation technique. From the result of the image velocity field there was a small bias, where for the low flow rate there was a small portion of the area which was quite far from the nozzle and has higher mean image velocity. However, for the higher flow rate, the result generated was close to the expected result where the image velocity was higher at the nozzle and reduces as it got further from the nozzle area.

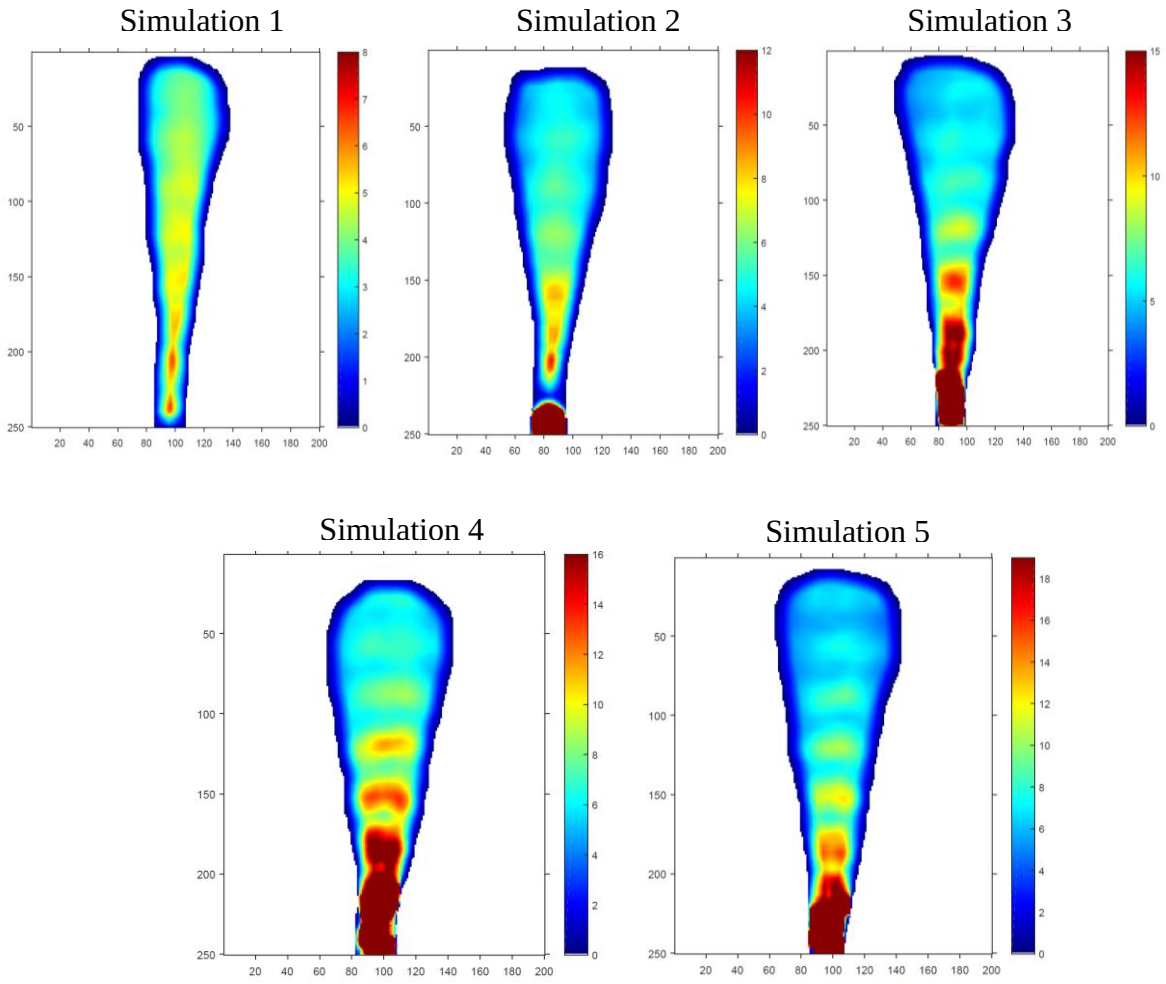


Figure 4.3: Typical 20-s time averaged image velocity field computed using the frequency domain cross correlation technique.

4.3.2 Flow Rate Metric

Table 4.3 shows the estimated flow rate metric of the simulated fluid for the five simulations, 1 to 5, computed using the frequency domain cross correlation technique. Overall, the percentage error was very high for all range of flow rate and the highest percentage error was 48%. This may be due to the low quality image where it has high brightness which added the noise to the image and can cause high percentage error during analysis.

Table 4.3: Estimated flow rates metric of simulated fluid computed using the frequency domain cross correlation technique.

Simulation	**Mean image velocity (pixel/ frame)**	**Estimated velocity (m/s)**	**Estimated Flow rate Metric (m^3/s)**	**Percentage error (%)**
1	3.59	0.08	7.06×10^{-6}	30
2	5.61	0.14	1.10×10^{-5}	41
3	6.38	0.15	1.25×10^{-5}	48
4	8.88	0.22	1.74×10^{-5}	32
5	9.58	0.23	1.88×10^{-5}	41

In fluid flow there will be region that experience discontinuity. Thus, the performance of this algorithm is sometimes unpredictable, giving error. The aperture effect causes the fluid flow to have an ambiguous transformation field and consequently this method will produce high percentage error [13].

Figure 4.4 shows the values of the estimated velocity plotted against the actual velocity for the frequency domain cross correlation technique. The non-zero-crossing regression intersects the horizontal axis at the actual velocity of - 0.02 m/s which is a larger negative intercept value compared to when using temporal cross correlation technique. This shows that, the result is more biased when using frequency domain cross correlation technique.

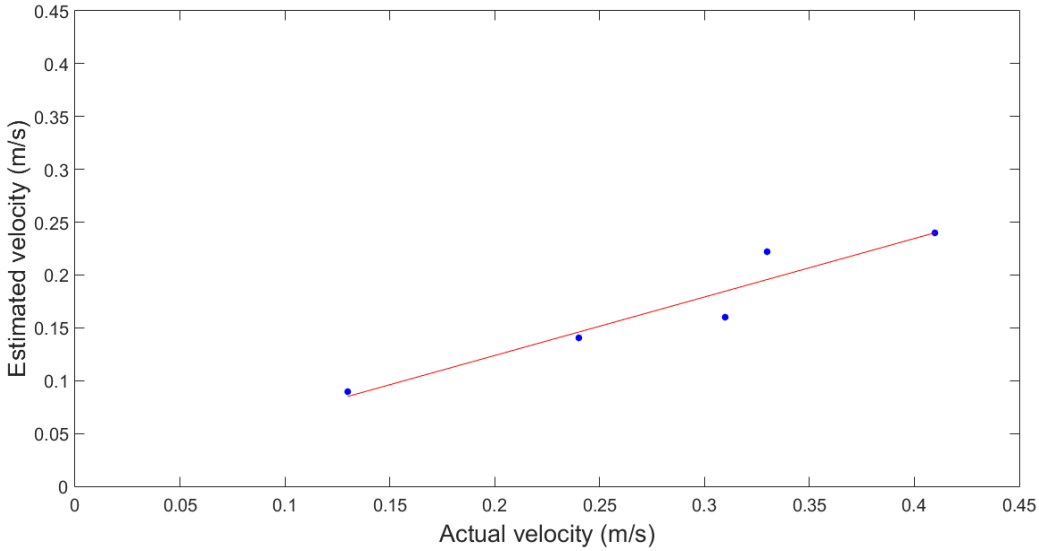


Figure 4.4: Estimated velocity from the five simulations compared to the actual velocity using the frequency domain cross correlation technique

4.4 Average Square Difference Function, ASDF

4.4.1 Image-velocity Field

Figure 4.5 shows a set of typical 20-s time averaged image velocity field for the five simulations, 1 to 5, computed using the average square difference function technique. From the result of the image velocity field, there was a small bias where for the low flow rate there was a small portion of the area which was quite far from the nozzle and has a higher mean image velocity. However, for the higher flow rate the image velocity was higher near the nozzle and reduces as it got further from the nozzle area which was as expected.

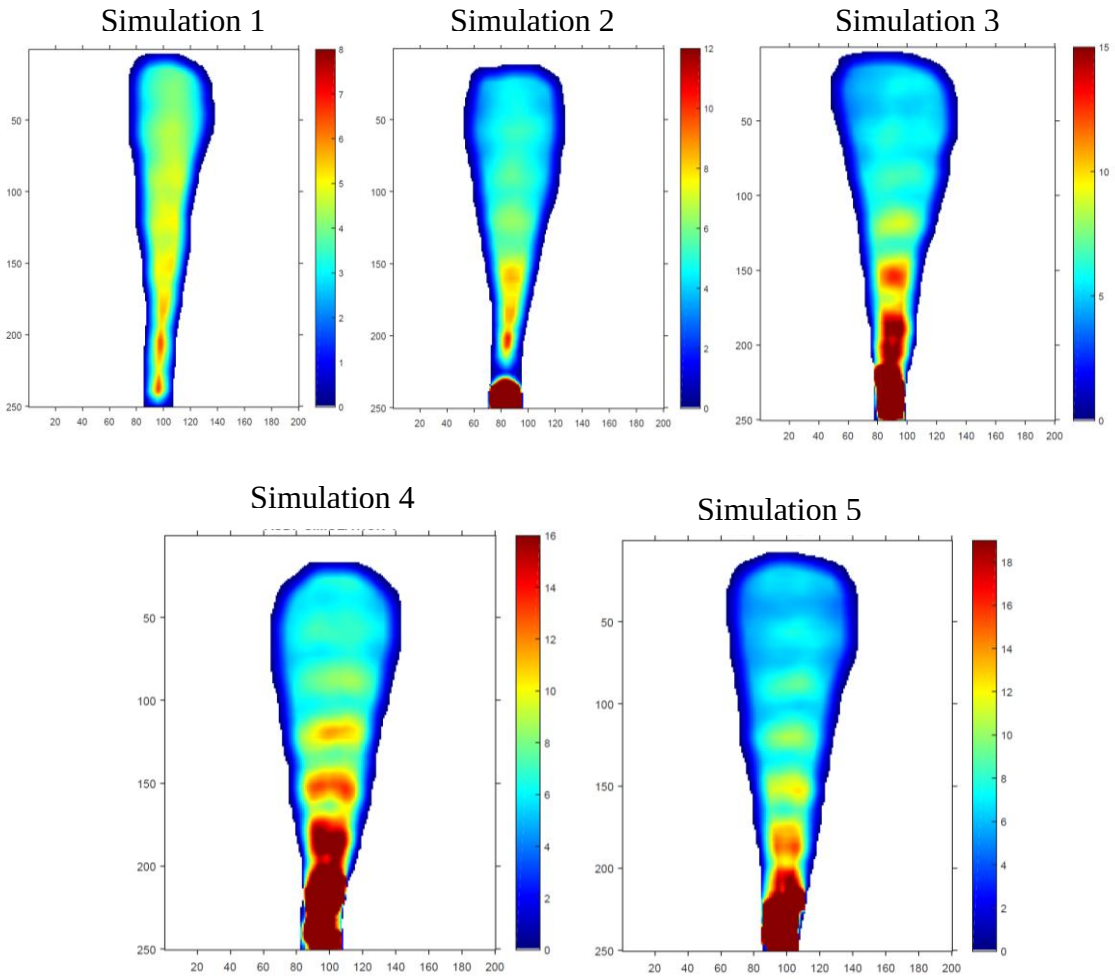


Figure 4.5: Typical 20-s time averaged image velocity field computed using the average square difference function technique, ASDF.

4.4.2 Flow Rate Metric

Table 4.4 shows the estimated flow rates metric of the simulated fluid for the five simulations, 1 to 5, computed using the average square difference function technique. Overall the percentage error was also very high for all ranges of the flow rate and the highest percentage error was 49%. The possible sources of errors will be further mentioned in the discussion.

Table 4.4: Estimated flow rates metric of simulated fluid computed using the average square difference function technique, ASDF

Simulation	**Mean image velocity (pixel/ frame)**	**Estimated velocity (m/s)**	**Estimated Flow rate Metric (m^3/s)**	**Percentage error (%)**
1	3.60	0.09	$7.08x10^{-6}$	31
2	5.62	0.15	$1.12x10^{-5}$	42
3	6.39	0.16	$1.27x10^{-5}$	49
4	8.90	0.23	$1.76x10^{-5}$	33
5	9.59	0.24	$1.90x10^{-5}$	42

In ASDF technique the fluid flow of the sample from different lag will interfere the true delay time even in zero noise situation. However in the real situation it is impossible to eliminate noise and this produce large percentage error. The variance of ASDF only tends to zero if two situations are satisfied; when noise is zero and when absolute different of the reference and delayed signal is zero. Both these situations are difficult to achieve, consequently there will be a large error [5].

Figure 4.6 shows the values of the estimated velocity plotted against the actual velocity for the average square difference function technique. The non-zero-crossing regression intersects the horizontal axis at the actual velocity of - 0.03 m/s which is a larger negative intercept value compared to when using frequency domain cross correlation technique. This shows that, the result is more biased when using average square difference function technique.

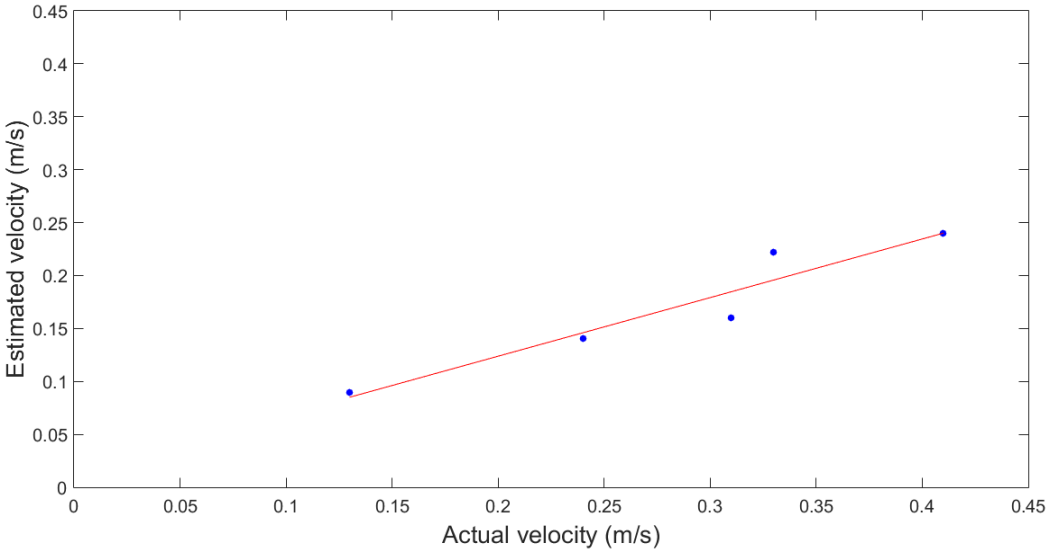


Figure 4.6: Estimated velocity from the five simulations compared to actual velocity using the average square difference function technique, ASDF

4.5 Average Magnitude Difference Function, AMDF

4.5.1 Image-velocity Field

Figure 4.7 shows a set of typical 20-s time averaged image velocity field for the five simulations, 1 to 5, computed using the average magnitude difference function technique. Among the four techniques, this technique produces the biggest bias result where the image velocity field near the nozzle is quite low and is not close to the expected result as can be seen in simulation 1 and simulation 2. The average magnitude difference function technique was not suitable for low flow rate. However, as the flow rate increases the mean image velocity becomes higher at the nozzle area.

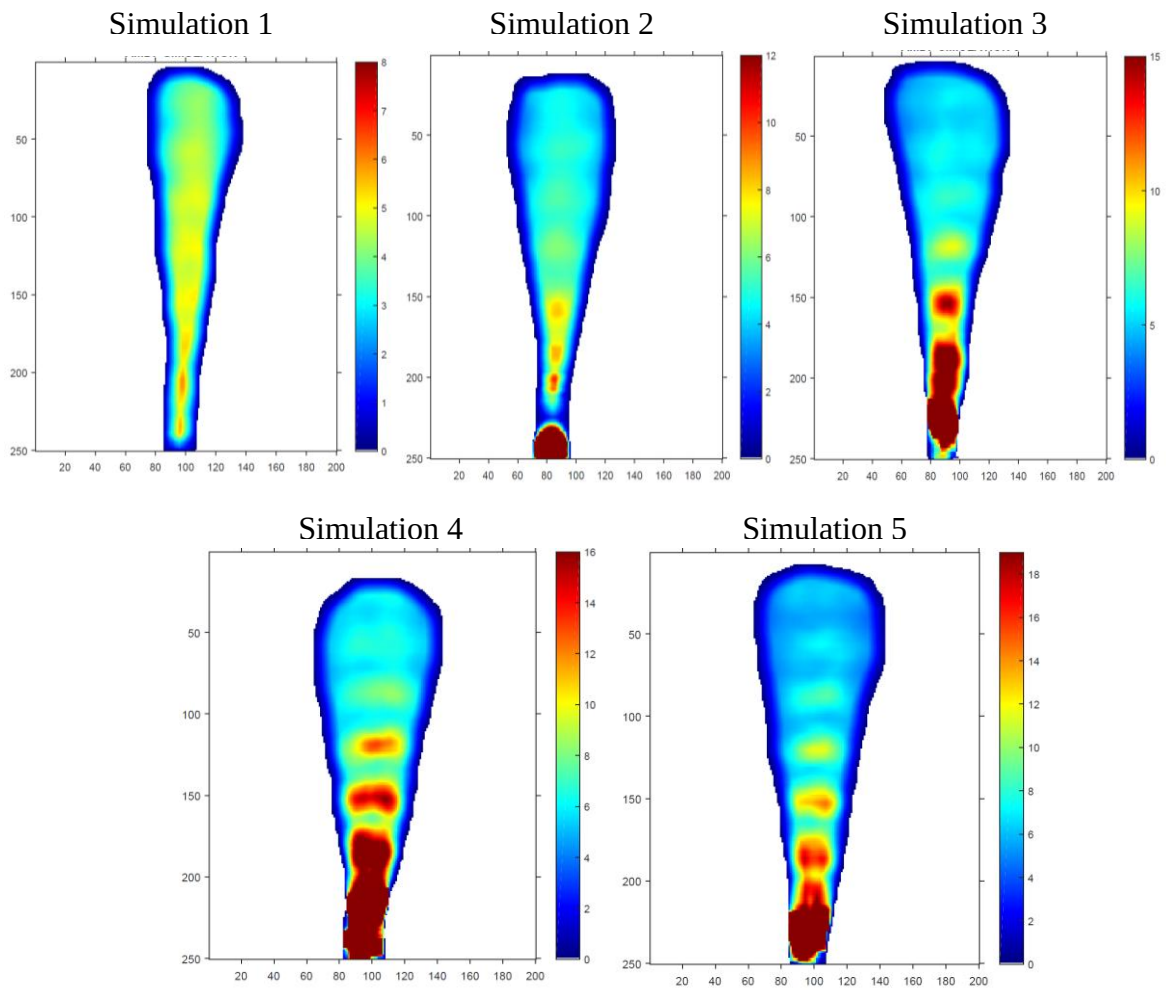


Figure 4.7: Typical 20-s time averaged image velocity field computed using the average magnitude difference function technique, AMDF.

4.5.2 Flow Rate Metric

Table 4.5 shows the estimated flow rates metric of the simulated fluid for the five simulations, 1 to 5, computed using the average magnitude difference function technique. Among the four techniques, this technique produces the highest percentage error which was about 54%.

Table 4.5: Estimated flow rates metric of simulated fluid computed using the average magnitude difference function technique, AMDF.

Simulation	**Mean image velocity (pixel/ frame)**	**Estimated velocity (m/s)**	**Estimated Flow rate Metric (m^3/s)**	**Percentage error (%)**
1	3.54	0.08	6.95x10-6	32
2	4.80	0.12	9.42x10-6	49
3	5.67	0.14	1.11x10-5	54
4	7.31	0.18	1.43x10-5	44
5	8.00	0.20	1.57x10-5	51

AMDF algorithm is suitable for real time operation. But for images with noise, AMDF will detect intensity incorrectly, thus this method will produce large error. In AMDF algorithm there is insufficient number of averaging for all lag time value and this is a limitation which also creates an error [14].

Figure 4.8 shows the values of the estimated velocity plotted against the actual velocity for the average square difference function technique. The non-zero-crossing regression intersects the horizontal axis at the actual velocity of - 0.06 m/s which is among the largest negative intercept value compared to when using other techniques. It shows that the result computed using average magnitude difference function technique is the most biased.

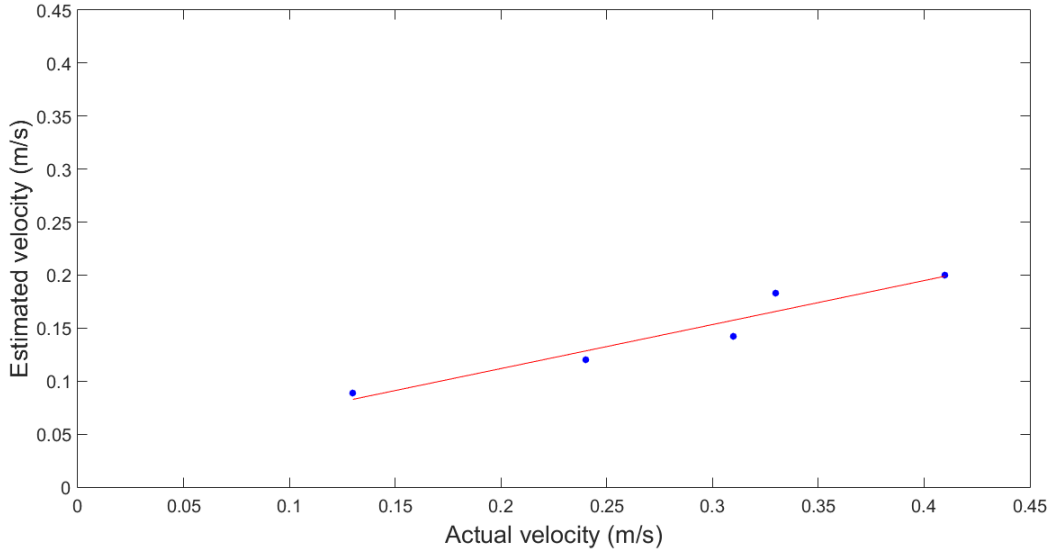


Figure 4.8: Estimated velocity from the five simulations compared to actual velocity using the average magnitude difference function technique, AMDF.

4.6 Discussion

Table 4.6 shows the percentage error when the flow rate was computed using the four techniques. Among the four techniques to compute the flow rate, the average magnitude difference function technique, AMDF has the least accuracy with a percentage error of 54% and the most accurate technique is the temporal cross correlation technique, with a maximum percentage error of 25%.

Table 4.6: Maximum percentage error when the flow rate was computed using the four difference techniques

No	Algorithm technique	Maximum percentage error (%)
1	Temporal cross correlation	25
2	Frequency domain cross correlation	48
3	Average square difference function, ASDF	49
4	Average magnitude difference function, AMDF	54

According to Crone et al, errors are contributed by inaccuracy within the images of the flow, the real flow or algorithm techniques used. Reddy et al. stated that, image registration methods using the frequency domain cross correlation such as fast Fourier transform, FFT methods are being applied. The FFT method is insensitive to translation, rotation, scale and noise thus making it more accurate compared to ASDF and AMDF [15]. Even though, the percentage error is lower compared to ASDF and AMDF, overall the percentage error of the frequency domain cross correlation technique is quite high due to its inability to search for optimal match of fluid flow images produced because of the dynamics of the fluid itself. For the Fourier approach to be successfully implemented it needs to match images that are translated, rotated, and scaled to one another [15].

AMDF method has the highest percentage error because this method is influenced by intensity variation of the images and background noise as being proved by Chen et al. [16].

From the result, it can be concluded that temporal cross correlation outperformed ASDF and AMDF method. This is further supported by Aiordachioaie and Nicolau

whereby both ASDF and AMDF have limitations in terms of producing the accurate delay range in comparison to direct cross correlation technique [17].

The image that was being captured was too bright and the feature of the flow was invisible and cannot be detected. The intensity of the entire image was almost the same, therefore the algorithm cannot differentiate the intensity within the entire image of the fluid flow. The adaptive histogram equalization can actually improve the contrast in the images and avoid amplifying any noise that might be present in the image. Even though the adaptive histogram equalization was performed to reduce the effect of poor illumination, the poor image quality is still the source of error. Figure 4.9 shows the effect of applying adaptive histogram equalization on the sample image where the features of the fluid flow are clearer.

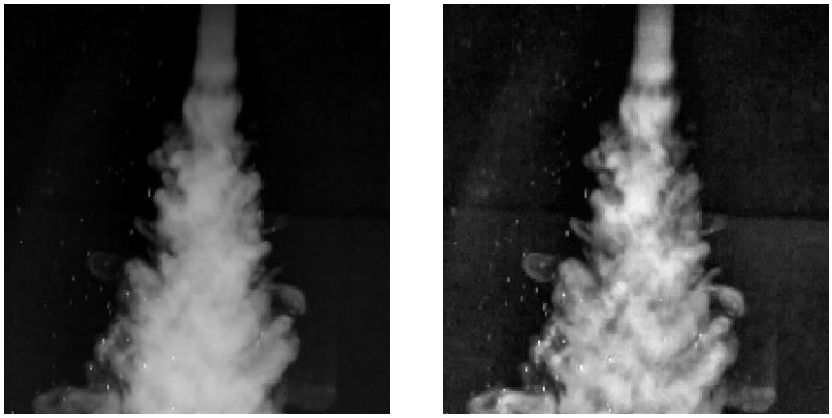


Figure 4.9: Effect on the features of the fluid flow before and after applying adaptive histogram equalization on the sample image

Besides, to calculate the image velocity, there are two important parameters that should be considered which are the distance of pixel separation and time delay. According to Crone et al, in order to obtain high accuracy measurement the best pixel separation should be five pixels. There is also the possibility that the pixel separation distance of five pixels might be suitable for certain condition of the fluid flow only. In this research, the distance of pixel separation is fixed to five pixels only. Therefore in the future work the implementation of iterative scheme is required, whereby the image-velocity was first estimated and then refined as the pixel separation distance was changed.

Furthermore, the number of images used for the analysis will also affect the accuracy of the estimated mean image velocity. According to Crone et al, the higher the number of images used, the lower the percentage error in estimating the mean image velocity. In this research, the number of image pairs used was 1000 pairs but yet the percentage error is still high. Thus, in the future work, larger number of image pairs should be used to estimate the mean image velocity.

CHAPTER 5: CONCLUSION AND RECOMMENDATION

In this work, a comparative analysis of four algorithms for optical plume velocimetry was performed for five different flow rates to investigate the accuracy of each algorithm. The four algorithms investigated were temporal cross correlation, frequency domain cross correlation, average square difference function, ASDF and average magnitude difference function, AMDF. Of the four algorithms considered, the temporal cross correlation algorithm was the most accurate, with a maximum percentage error of 25%. However, all techniques still produced a non-zero crossing relationship between the estimated velocity and actual velocity across the jet nozzle of -0.01 m/s, -0.02 m/s, -0.03 m/s and -0.06 m/s for temporal cross correlation, frequency domain cross correlation, average square difference function, ASDF and average magnitude difference function, AMDF respectively.

The other three algorithms still need further improvement in order to obtain more accurate result. In the future work, more test-run should be done and in order to get lower percentage error, larger number of image pairs should be used to estimate the mean image velocity. Besides, different pixel separation distance should also be used to investigate which value of pixel separation distance will result in higher accuracy in estimating the image velocity. Furthermore, in the future work, 3D camera can also be used to capture 3D image of fluid flow which will allow the volume of the fluid flow to be computed and the volumetric flow rate can be obtained by dividing the volume of the fluid flow with time.

REFERENCES

[1] M. K. McNutt, R. Camilli, T. J. Crone, G. D. Guthrie, P. A. Hsieh, T. B. Ryerson, O. Savas and F. Shaffer, “Review of flow rate estimate of the Deepwater Horizon Oil spill,” In *Proceedings of Natural Academy of Science,* vol. 109, no. 50, pp. 20260- 20267, 2012.

[2] B. Galvin, B. McCane, K. Novins, D. Mason, and S. Mills, "Recovering Motion Fields: An Evaluation of Eight Optical Flow Algorithms," in BMVC, 1998, pp. 195-204.

[3] T. J. Crone, R. E. Mcduff, and W. S. D. Wilcock, “Optical plume velocimetry: a new flow measurement technique for use in seafloor hydrothermal systems,” *Exp Fluids*, vol. 45, no. 5, pp. 899-915, 2008.

[4] O. A. B. Mohmmed, M. Ovinis, and F. M. Hashim, "Comparative analysis of multi-resolution optical flow techniques for flow rate estimation," in *Control Conference (ASCC), 2015 10th Asian*, 2015, pp. 1-5.

[5] G. Jacovitti and G. Scarano, "Discrete time techniques for time delay estimation,"*IEEE Transactions on Signal Processing,* vol. 41, pp. 525-533, 2002.

[6] D. Dabiri, "Cross-correlation digital particle image velocimetry—a review," *Turbulencia. ABCM, Curitiba*, pp. 155-199, 2006.

[7] T. J. Crone and M. Tolstoy, "Magnitude of the 2010 Gulf of Mexico oil leak," *Science*, vol. 330, pp. 634-634, 2010.

[8] S. Ferrari, M. Badas, L. Besalduch, and G. Querzoli, "A Feature Tracking Velocimetry algorithm to determine the velocities in Negatively Buoyant Jets," in *PIV13; 10th International Symposium on Particle Image Velocimetry, Delft, The Netherlands, July 1-3, 2013*, 2013.

[9] H. Zafar, F. Sharif, and M. J. Leahy, "Measurement of the blood flow rate and velocity in coronary artery stenosis using intracoronary frequency domain optical coherence tomography: Validation against fractional flow reserve," *IJC Heart & Vasculature,* vol. 5, pp. 68-71, 12// 2014.

[10] A. A. Parthasarathi, S. A. Japee, and R. N. Pittman, "Determination of Red Blood Cell Velocity by Video Shuttering and Image Analysis," *Annals of Biomedical Engineering*, vol. 27, pp. 313-325, 1999.

[11] Y. Zhang and W. H. Abdulla, "A comparative study of time-delay estimation techniques using microphone arrays," *Department of Electrical and Computer Engineering, The University of Auckland, School of Engineering Report,* vol. 619, 2005.

[12] L. H. P and R. M. Nisarg, "Comparative Study of Frequency Vs Spacial Domain for Multi Sensor Image Fusion," *International Journal of Innovative Research in Science, Engineering and Technology,* vol. 3, 2014.

[13] S. Kruger and A. Calway, "Image Registration using Multiresolution Frequency Domain Correlation," in *BMVC*, 2008, pp. 1-10.

[14] G. Muhammad, "Extended average magnitude difference function based pitch detection," *The International Arab Journal of Information Technology,* vol. 8, pp. 197-203, 2011.

[15] B. S. Reddy and B. N. Chatterji, "An FFT-based technique for translation, rotation, and scale-invariant image registration," *IEEE transactions on image processing,* vol. 5, pp. 1266-1271, 2014.

[16] J. Chen, J. Benesty, and Y. A. Huang, "Performance of GCC-and AMDF-based time-delay estimation in practical reverberant environments," *EURASIP Journal on Advances in Signal Processing*, vol. 2005, p. 498964, 2005.

[17] D. Aiordachioaie and V. Nicolau, "On time delay estimation by evaluation of three time domain functions," in *2010 3rd International Symposium on Electrical and Electronics Engineering (ISEEE)*, 2010, pp. 281-286.

APPENDICES

A) Image velocity field

$$u = \frac{d}{t}$$ Equation 1

Where :

u =image velocity (pixel/frame)

d =distance of pixel separation (pixel)

t = time delay (frame)

B) Velocity of fluid

$$v = u \times FPS \times k$$ Equation 2

Where

v = Velocity of fluid (m/s)

$u =$ image velocity (pixel/frame)

FPS = speed of the video changing from one frame to another (frame/s)

k = calibration constant of the image (m/pixel)

Calibration constant is the ratio of set known correspondences between point features in the real world (m) and their projections on the image (pixel)

C) Volumetric flow rate

$$Q = v \times A$$ Equation 3

Where

Q = Flow rate (m^3/s)

v = Velocity of fluid (m/s)

A= Area of nozzle

= $\pi d^2/4$ (diameter of nozzle is 10mm)

D) Theories of four different technique:

a) Temporal Cross correlation

Temporal cross correlation function defined by:

$$C_{i,j,l}(d) = \sum_{k=1}^{N-1} a_{i+d,j,k} \cdot a_{i,j,k+l} \qquad \text{Equation 4}$$

Where $a_{i,j,k}$ represent a three dimensional pixel intensity matrix corresponding to an image sequence with N frames which has been detrended at each pixel location in the time direction , where i,j and k index a in the vertical, horizontal and time direction.

The temporal cross correlation estimate the delay based on measuring the similarity between two signal (intensity) in time domain. The time delay correspond to the point of maximum cross correlation coefficient as shown in Figure 1 below.

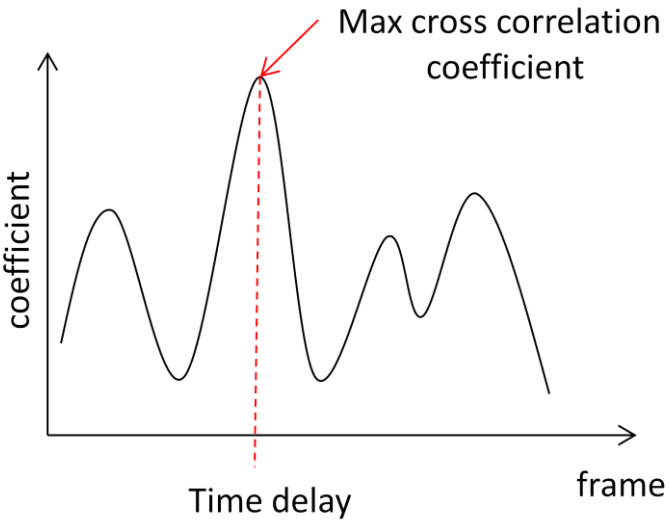


Figure 1: The cross correlation coefficient versus frame after cross correlation was done

b) Frequency domain cross correlation

The different between Frequency domain cross correlation and Temporal cross correlation where Temporal cross correlation is in time domain. In frequency domain cross correlation, the time domain will be converted into frequency domain using Fourier transform.

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt$$ Equation 5

The frequency domain cross correlation estimate the delay based on measuring the similarity between two signal (intensity) in frequency domain. It apply the same concept with temporal cross correlation where the time delay correspond to the maximum cross correlation coefficient.

c) Averaged squared difference function, ASDF

The ASDF method is based on finding the position of the minimum error square between two received signal and considering this position value as the estimated time delay. It is defined by :

$$R_{ASDF}[\tau] = \frac{1}{N} \sum_{n=0}^{N-1} [r_1(n) - r_2(n+\tau)]^2$$ Equation 6

$$D_{ASDF} = \arg\min_{\tau} [R_{ASDF}(\tau)]$$

d) Averaged magnitude difference function, AMDF

The AMDF does not involve multiplication and it is useful in application where low computational complexity is required.

$$R_{AMDF}[\tau] = \frac{1}{N}\sum_{n=0}^{N-1}[r_1(n) - r_2(n+\tau)] \qquad \text{Equation 7}$$

$$D_{AMDF} = \arg\min_{\tau}[R_{AMDF}(\tau)]$$

Both algorithm, ASDF and AMDF estimate the delay based on minimum error between the two signal (intensity) by finding the position of minimum error between two received signal and considering this position value as the estimated time delay as shown in Figure 2 below.

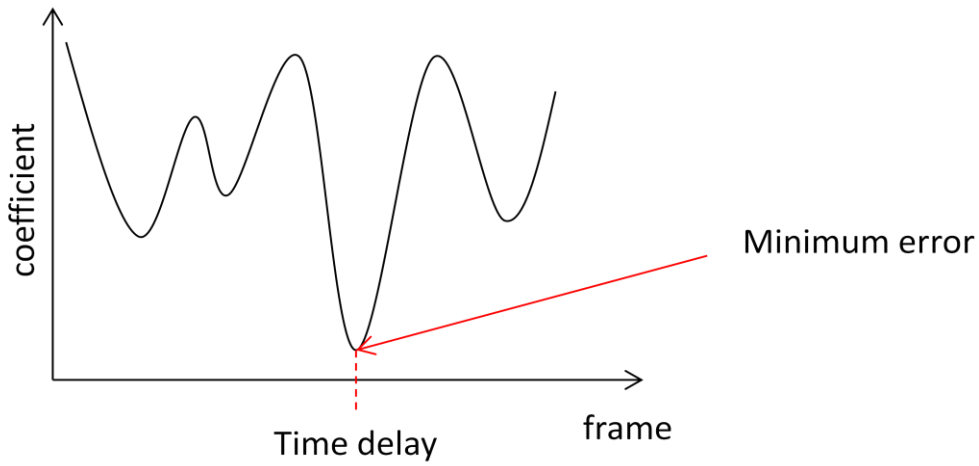


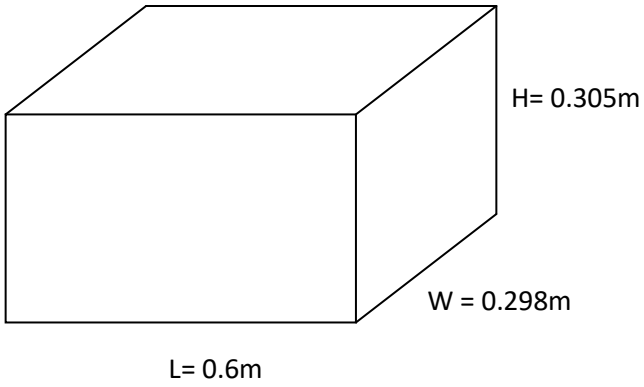
Figure 2: The coefficient of ASDF or AMDF versus frame

E)Example for the conversion of flow rate to image velocity, velocimetry for simulation

1. T_i is the time taken for the mixed fluid in the lower source tank to displaced by 2cm.

Simulation	**Time, T_i (s)**	**Average Time, (s)**	**Volumetric Flow rate (m^3/s)**	**Flow velocity (m/s)**	**Image-velocity, Velocimetry (pixel/frame)**
1	344	345	1.0365×10^{-5}	0.13	5.2
	346				
	343				

Dimension of lower source tank:



$V_D = A \times h$ Equation 8

Where V_D= Displaced volume (m^3)

A= Area of container (m^2)

h= Displaced height (m)

$V_D = (0.6m \times 0.298m) \times 0.02m = 3.576 \times 10^{-3}\ m^3$

$$\dot{V} = \frac{V_D}{t}$$ Equation 9

Where $\dot{V}$ = Volumetric flow rate (m^3/s)

V_D= Displaced volume (m^3)

t = Time taken (s)

$$\dot{V} = \frac{V_D}{t} = \frac{3.576\times10^{-3}\ m^3}{345\ s} = 1.0365 \times 10^{-5}\ \frac{m^3}{s}$$

The radius of the nozzle is r = 0.01m, and the calibration constant of the image, k= (1/2000) m/pixel which has been determined from the ruler with alternating black and white squares, which was placed beside the nozzle as indicator to determine the relationship between the pixel and the length. The camera used in the experiment had the speed of 50 fps , $v_{f/s}$ = 50fps. Thus, the image velocity, velocimetry (pixel/frame) can be calculated by using the given information.

$$v = \frac{\dot{V}}{\mathrm{A_{nozzle}}}$$ Equation 10

Where v = Flow velocity (m/s)

$\mathrm{A_{nozzle}}$ = Cross sectional area of nozzle (m^2)

$$v = \frac{1.0365\times10^{-5}(m^3/s)}{\pi(0.005\mathrm{m})^2} = 0.13\ \mathrm{m/s}$$

$$v_{p/f} = \frac{v}{\mathrm{kv_{f/s}}}$$ Equation 11

Where $v_{p/f}$ = Image velocity, velocimetry (pixel/frame)

$$v_{p/f} = \frac{0.13\ \mathrm{m/s}}{\left(\frac{1}{2000}\frac{\mathrm{m}}{\mathrm{pixel}}\right)\times\left(\frac{50}{1}\frac{\mathrm{frame}}{\mathrm{second}}\right)} = 5.2\frac{\mathrm{pixel}}{\mathrm{frame}}$$

F) MATLAB coding for converting video into sequence of images

```
clc
clear all
close all

% Demo macro to extract frames and get frame means from an avi movie
% and save individual frames to separate image files.
% Then rebuilds a new movie by recalling the saved images from disk.
% Also computes the mean gray value of the color channels
% And detects the difference between a frame and the previous frame.
clc;    % Clear the command window.
close all;  % Close all figures (except those of imtool.)
imtool close all;  % Close all imtool figures.
clear;  % Erase all existing variables.
workspace;  % Make sure the workspace panel is showing.
fontSize = 14;

% Change the current folder to the folder of this m-file.
% (The line of code below is from Brett Shoelson of The Mathworks.)
if(~isdeployed)
    cd(fileparts(which(mfilename)));
end

% Open the rhino.avi demo movie that ships with MATLAB.
folder = fullfile(matlabroot, 'C:\Users\User\Documents\MATLAB');
% movieFullFileName = fullfile(folder, 'rhinos.avi');
% movieFullFileName = fullfile(folder, 'DSCN1120.MOV');
movieFullFileName = fullfile(folder, '3.MOV');
% Check to see that it exists.
if ~exist(movieFullFileName, 'file')
    strErrorMessage = sprintf('File not found:\n%s\nYou can choose a
new one, or cancel', movieFullFileName);
    response = questdlg(strErrorMessage, 'File not found', 'OK -
choose a new movie.', 'Cancel', 'OK - choose a new movie.');
    if strcmpi(response, 'OK - choose a new movie.')
        [baseFileName, folderName, FilterIndex] =
uigetfile('*.avi');
        if ~isequal(baseFileName, 0)
            movieFullFileName = fullfile(folderName, baseFileName);
        else
            return;
        end
    else
        return;
    end
end

try
    videoObject = VideoReader(movieFullFileName)
    % Determine how many frames there are.
    numberOfFrames = videoObject.NumberOfFrames;
    vidHeight = videoObject.Height;
    vidWidth = videoObject.Width;

    numberOfFramesWritten = 0;
```

```
   % Prepare a figure to show the images in the upper half of the
screen.
    figure;
    %   screenSize = get(0, 'ScreenSize');
    % Enlarge figure to full screen.
    set(gcf, 'units','normalized','outerposition',[0 0 1 1]);

    % Ask user if they want to write the individual frames out to
disk.
    promptMessage = sprintf('Do you want to save the individual
frames out to individual disk files?');
    button = questdlg(promptMessage, 'Save individual frames?',
'Yes', 'No', 'Yes');
    if strcmp(button, 'Yes')
        writeToDisk = true;

        % Extract out the various parts of the filename.
        [folder, baseFileName, extentions] =
fileparts(movieFullFileName);
        % Make up a special new output subfolder for all the
separate
        % movie frames that we're going to extract and save to disk.
        % (Don't worry - windows can handle forward slashes in the
folder name.)
        folder = pwd;   % Make it a subfolder of the folder where
this m-file lives.
        outputFolder = sprintf('%s/Movie Frames from %s', folder,
baseFileName);
        % Create the folder if it doesn't exist already.
        if ~exist(outputFolder, 'dir')
            mkdir(outputFolder);
        end
    else
        writeToDisk = false;
    end

    % Loop through the movie, writing all frames out.
    % Each frame will be in a separate file with unique name.
    meanGrayLevels = zeros(numberOfFrames, 1);
    meanRedLevels = zeros(numberOfFrames, 1);
    meanGreenLevels = zeros(numberOfFrames, 1);
    meanBlueLevels = zeros(numberOfFrames, 1);
    for frame = 1 : numberOfFrames
        % Extract the frame from the movie structure.
        thisFrame = read(videoObject, frame);

        % Display it
        hImage = subplot(2, 2, 1);
        image(thisFrame);
        caption = sprintf('%d.', frame, numberOfFrames);
        title(caption, 'FontSize', fontSize);
        drawnow; % Force it to refresh the window.

        % Write the image array to the output file, if requested.
        if writeToDisk
            % Construct an output image file name.
            outputBaseFileName = sprintf('Frame %4.4d.jpg', frame);
```

```
            outputFullFileName = fullfile(outputFolder,
outputBaseFileName);

            % Stamp the name and frame number onto the image.
            % At this point it's just going into the overlay,
            % not actually getting written into the pixel values.
            % text(5, 15, outputBaseFileName, 'FontSize', 20);

            % Extract the image with the text "burned into" it.
            frameWithText = getframe(gca);
            % frameWithText.cdata is the image with the text
            % actually written into the pixel values.
            % Write it out to disk.
            imwrite(frameWithText.cdata, outputFullFileName, 'jpg');
        end

        % Calculate the mean gray level.
        grayImage = rgb2gray(thisFrame);
        meanGrayLevels(frame) = mean(grayImage(:));

        % Calculate the mean R, G, and B levels.
        meanRedLevels(frame) = mean(mean(thisFrame(:, :, 1)));
        meanGreenLevels(frame) = mean(mean(thisFrame(:, :, 2)));
        meanBlueLevels(frame) = mean(mean(thisFrame(:, :, 3)));

        % Plot the mean gray levels.
        hPlot = subplot(2, 2, 2);
        hold off;
        plot(meanGrayLevels, 'k-', 'LineWidth', 2);
        hold on;
        plot(meanRedLevels, 'r-');
        plot(meanGreenLevels, 'g-');
        plot(meanBlueLevels, 'b-');
        grid on;

        % Put title back because plot() erases the existing title.
        title('Mean Gray Levels', 'FontSize', fontSize);
        if frame == 1
            xlabel('Frame Number');
            ylabel('Gray Level');
            % Get size data later for preallocation if we read
            % the movie back in from disk.
            [rows, columns, numberOfColorChannels] =
size(thisFrame);
        end

        % Update user with the progress.  Display in the command
window.
        if writeToDisk
            progressIndication = sprintf('Wrote frame %4d of %d.',
frame, numberOfFrames);
        else
            progressIndication = sprintf('Processed frame %4d of
%d.', frame, numberOfFrames);
        end
        disp(progressIndication);
```

```
        % Increment frame count (should eventually = numberOfFrames
        % unless an error happens).
        numberOfFramesWritten = numberOfFramesWritten + 1;

        % Now let's do the differencing
        alpha = 0.5;
        if frame == 1
            Background = thisFrame;
        else
            % Change background slightly at each frame
            %           Background(t+1)=(1-alpha)*I+alpha*Background
            Background = (1-alpha)* thisFrame + alpha * Background;
        end
        % Display the changing/adapting background.
        subplot(2, 2, 3);
        imshow(Background);
        title('Adaptive Background', 'FontSize', fontSize);
        % Calculate a difference between this frame and the
background.
        differenceImage = thisFrame - uint8(Background);
        % Threshold with Otsu method.
        grayImage = rgb2gray(differenceImage); % Convert to gray
level
        thresholdLevel = graythresh(grayImage); % Get threshold.
        binaryImage = im2bw( grayImage, thresholdLevel); % Do the
binarization
        % Plot the binary image.
        subplot(2, 2, 4);
        imshow(binaryImage);
        title('Binarized Difference Image', 'FontSize', fontSize);
    end

    % Alert user that we're done.
    if writeToDisk
        finishedMessage = sprintf('Done!  It wrote %d frames to
folder\n"%s"', numberOfFramesWritten, outputFolder);
    else
        finishedMessage = sprintf('Done!  It processed %d frames
of\n"%s"', numberOfFramesWritten, movieFullFileName);
    end
    disp(finishedMessage); % Write to command window.
    uiwait(msgbox(finishedMessage)); % Also pop up a message box.

    % Exit if they didn't write any individual frames out to disk.
    if ~writeToDisk
        return;
    end

    % Ask user if they want to read the individual frames from the
disk,
    % that they just wrote out, back into a movie and display it.
    promptMessage = sprintf('Do you want to recall the individual
frames\nback from disk into a movie?\n(This will take several
seconds.)');
    button = questdlg(promptMessage, 'Recall Movie?', 'Yes', 'No',
'Yes');
    if strcmp(button, 'No')
```

```
        return;
    end

    % Create a VideoWriter object to write the video out to a new,
different file.
    writerObj = VideoWriter('NewRhinos.mov');
    open(writerObj);

    % Read the frames back in from disk, and convert them to a
movie.
    % Preallocate recalledMovie, which will be an array of
structures.
    % First get a cell array with all the frames.
    allTheFrames = cell(numberOfFrames,1);
    allTheFrames(:) = {zeros(vidHeight, vidWidth, 3, 'uint8')};
    % Next get a cell array with all the colormaps.
    allTheColorMaps = cell(numberOfFrames,1);
    allTheColorMaps(:) = {zeros(256, 3)};
    % Now combine these to make the array of structures.
    recalledMovie = struct('cdata', allTheFrames, 'colormap',
allTheColorMaps)
    for frame = 1 : numberOfFrames
        % Construct an output image file name.
        outputBaseFileName = sprintf('Frame %4.4d.jpg', frame);
        outputFullFileName = fullfile(outputFolder,
outputBaseFileName);
        % Read the image in from disk.
        thisFrame = imread(outputFullFileName);
        % Convert the image into a "movie frame" structure.
        recalledMovie(frame) = im2frame(thisFrame);
        % Write this frame out to a new video file.
        writeVideo(writerObj, thisFrame);
    end
    close(writerObj);
    % Get rid of old image and plot.
    delete(hImage);
    delete(hPlot);
    % Create new axes for our movie.
    subplot(1, 3, 2);
    axis off;  % Turn off axes numbers.
    title('Movie recalled from disk', 'FontSize', fontSize);
    % Play the movie in the axes.
    movie(recalledMovie);
    % Note: if you want to display graphics or text in the overlay
    % as the movie plays back then you need to do it like I did at
first
    % (at the top of this file where you extract and imshow a frame
at a time.)
    msgbox('Done with this demo!');

catch ME
    % Some error happened if you get here.
    strErrorMessage = sprintf('Error extracting movie frames
from:\n\n%s\n\nError: %s\n\n)', movieFullFileName, ME.message);
    uiwait(msgbox(strErrorMessage));
end
%}
```

G) MATLAB coding to convert the images into grayscale, cropped the image and convert the sequences of images into three dimensional pixel intensity matrix

```
clc
clear all;

%% read images fromfolder
fileFolder =
fullfile('C:\Users\nurul\Desktop\FYPII\New\Video\Q1_0.13ms\used
frames'); % images1 is a folder in MATLAB directory
dirOutput = dir(fullfile(fileFolder,'*.jpg')); %new picture is the
name of the image
fileNames = {dirOutput.name}';
numFrames = numel(fileNames);
images = imread(fileNames{1});
%% Put all images in 3D matrix..
for i = 1:numFrames
 sequence_Q1(:,:,i) = rgb2gray(imread(fileNames{i}));
end
%%
save sequence_Q1 sequence_Q1

%
load sequence_Q1.mat

imtool(sequence_Q1(:,:,1),[])
%%
for i = 1:numFrames
 sequence_Q11(:,:,i) = imcrop(sequence_Q1(:,:,i),[205 15 199 249]);
end
save sequence_Q11 sequence_Q11

for i = 1:numFrames
 sequence_Q1_adap(:,:,i) = adapthisteq(sequence_Q11(:,:,i));
end
save sequence_Q1_adap sequence_Q1_adap

%}
```

H) MATLAB coding to determine the time delay using temporal cross correlation

```
function delay = OPV_delayest_3point(u2,u1);

u1 = reshape(u1,[1 length(u1)]);
u2 = reshape(u2,[1 length(u1)]);

N_p=numel(u2);%number of elements
% temporal cross correlation
[xc lags] = xcorr(u1,u2,'coeff');
[tmp idx] = max(xc(:));

R=zeros(1,3);%three points to use for interpolation
R(2)=xc(idx);%peak

%find neighbors, assuming periodic
if idx==1
    R(1)=xc(end);
    R(3)=xc(2);
elseif idx==numel(xc)
    R(1)=xc(end-1);
    R(3)=xc(1);
else
    R(1)=xc(idx-1);
    R(3)=xc(idx+1);
end

   c=(log(R(3))-log(R(1)))/(4*log(R(2))-2*log(R(1))-2*log(R(3)));

lg = lags(idx);
delay = lg + c;
```

I) MATLAB coding to determine the time delay using frequency domain cross correlation

```
function delay = OPV_delayest_3point(u2,u1);

u1 = reshape(u1,[1 length(u1)]);
u2 = reshape(u2,[1 length(u1)]);

N_p=numel(u2);%number of elements
% frequency domain cross correlation
        U1=fft(u1);
        U2=fft(u2);
        xc=ifft(U2.*conj(U1));
        [tmp idx]=max(xc);

R=zeros(1,3);%three points to use for interpolation
R(2)=xc(idx);%peak

%find neighbors, assuming periodic
if idx==1
    R(1)=xc(end);
    R(3)=xc(2);
elseif idx==numel(xc)
    R(1)=xc(end-1);
    R(3)=xc(1);
else
    R(1)=xc(idx-1);
    R(3)=xc(idx+1);
end

  c=(log(R(3))-log(R(1)))/(4*log(R(2))-2*log(R(1))-2*log(R(3)));

lag = mod(idx-1+floor(N_p/2),N_p)-floor(N_p/2); %integer part
delay = lag + c; % delay estimate
```

J) MATLAB coding to determine the time delay using average square difference Function, ASDF.

```
function delay = OPV_delayest_3point(u2,u1);
u1 = reshape(u1,[1 length(u1)]);
u2 = reshape(u2,[1 length(u1)]);

N_p=numel(u2);%number of elements

 % average squared difference function
   xc = (-2*ifft(fft(u2).*conj(fft(u1))) + sum(u1.^2) +
sum(u2.^2))/N_p;%ADSF
  [tmp idx]=min(xc);

R=zeros(1,3);%three points to use for interpolation
R(2)=xc(idx);%peak

%find neighbors, assuming periodic
if idx==1
    R(1)=xc(end);
    R(3)=xc(2);
elseif idx==numel(xc)
    R(1)=xc(end-1);
    R(3)=xc(1);
else
    R(1)=xc(idx-1);
    R(3)=xc(idx+1);
end

 c=(log(R(3))-log(R(1)))/(4*log(R(2))-2*log(R(1))-2*log(R(3)));


lag = mod(idx-1+floor(N_p/2),N_p)-floor(N_p/2); %integer part
delay = lag + c; % delay estimate
```

K) MATLAB coding to determine the time delay using average magnitude difference Function, AMDF

```
function delay = OPV_delayest_3point(u2,u1);

u1 = reshape(u1,[1 length(u1)]);
u2 = reshape(u2,[1 length(u1)]);

N_p=numel(u2);%number of elements

   %average magnitude difference function (this is a lot slower for
large N_p)
    xc=sum(abs(repmat(u1',1,N_p)-hankel(u2',[u2(end)'; u2(1:(end-
1))'])))/N_p;%AMDF
    [tmp idx]= min(xc);

R=zeros(1,3);%three points to use for interpolation
R(2)=xc(idx);%peak

%find neighbors, assuming periodic
if idx==1
    R(1)=xc(end);
    R(3)=xc(2);
elseif idx==numel(xc)
    R(1)=xc(end-1);
    R(3)=xc(1);
else
    R(1)=xc(idx-1);
    R(3)=xc(idx+1);
end

 c=(log(R(3))-log(R(1)))/(4*log(R(2))-2*log(R(1))-2*log(R(3)));

lag = mod(idx-1+floor(N_p/2),N_p)-floor(N_p/2); %integer part
delay = lag + c; % delay estimate
```

L) MATLAB coding to determine the image velocity field

```
clc
clear all
close all

tic

 load sequence_Q5_adap.mat
 sequence = sequence_Q5_adap;


zz = sequence;
[row1, column1, frame1] = size(zz);
maxcorr = frame1;

d = 5;
%%

for jj = 1:column1
 for ii = 1:row1-d

delay(ii,jj) = OPV_delayest_3point(zz(ii,jj,:),zz(ii+d,jj,:));
disp('I am calculating time delays...')

U(ii,jj) = d./delay(ii,jj);
%%
if U(ii,jj) == Inf
    U(ii,jj) = 0;
else
    U(ii,jj) = U(ii,jj);
end
 end
end


%% output velocity field...
U = imrotate(U,180);
U_TCC_Q5 = U;
save U_TCC_Q5 U_TCC_Q5
%%

% U = imrotate(U,180);
figure ('color','white')
imshow (abs(U),[0 16.4])
colormap jet
colorbar

toc
```

M) MATLAB coding to produce image velocity field result

```
clc
clear all
close all

%%
%% Binary image..
load binaryImage_Q1.mat
load U_AMDF_Q1.mat

%%
% scale = 50/2000;
% U1 =  scale*(abs(U_TCC_Q1));

U1 =  abs(U_AMDF_Q1);
%%
[m, n] = size (binaryImage_Q1);
for i = 1:m
     for j = 1:n
           if binaryImage_Q1(i,j) == 0
               U1(i,j)=0;

          else if binaryImage_Q1(i,j) == 1
                 U1(i,j) = U1(i,j);
                  end
          end
     end

end
%% Filter velocity field..
sm = 5;
U1 = smooth2a(U1,sm,sm);
%% display results
figure('color','white')
imshow(U1,[0 8])

currentMap = colormap(jet(250));
newMap = [1 1 1; currentMap];
colormap(newMap)
axis on
colorbar

title('AMDF SIMULATION 1')

%%
```