STATISTICAL INVESTIGATION OF ERRORS IN PARTICLE IMAGE VELOCIMETRY

A Thesis

Presented in Partial Fulfillment of the Requirements for
the degree Master of Science in the
Graduate School of the Ohio State University

by

NIKOLAOS KIRITSIS, B.S.

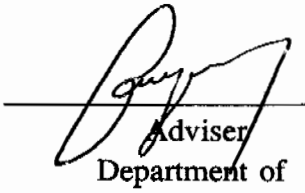
* * * * *

The Ohio State University

1989

Master's Examination Committee:
R. Brodkey
Y. G. Guezennec
M. Waldron

Approved by



Adviser
Department of
Mechanical Engineering

To My Father & Mother, *Ioannis and Anna Kiritsis*

To My Teacher, Mentor and Friend, "*Doc*"

ACKNOWLEDGEMENTS

Many people were of much help to me during the execution of this project. First of all I would like to thank my adviser, Dr. Yann Guezennec, for his constant help, encouragement and guidance during the time I spent working on the project. I also appreciate his assistance and valuable comments in writting the manuscript.

Thanks go the other members of my advisory committee, Drs. Robert Brodkey and Manjula Waldron, for their suggestions and comments.

I would also like to acknowledge the direct assistance I received from Woong-Chul Choi in helping me use the computer software. Special recognition is due to Tom Gieseke and Nizar Trigui for their help throughout the duration of the project.

VITA

September 16, 1964	Born, Florina, Greece.
1987	Bachelor of Science, Engineering Physics, Wright State University, Dayton, Ohio.
1987	Bachelor of Science, Physics, Wright State University, Dayton, Ohio.
1987-Present	Graduate Research and Teaching Associate Mechanical Engineering Dept., The Ohio State University.

FIELDS OF STUDY

Major Field : Mechanical Engineering
Area of Interest : Fluid Mechanics

TABLE OF CONTENTS

LIST OF FIGURES

NOMENCLATURE

a_b	Amplitude of P_b
a_p	Amplitude of P_p
A	Standard deviation of image noise
A_r	Standard deviation of particle radius
$A(n,m)$	Subregion of $I(x,y)$ or $S(x,y)$
C	Contrast between the particles and the background
D_{max}	Maximum particle displacement in a flow field
H(i)	Histogram of image
i	Particle number
I_θ	Threshold value
$I(x,y)$	Generated image
ndd	A normally distributed deviate with zero mean and unit variance
M	The number of pixels in subregion $A(x,y)$
N_p	Number of particles in a generated image
N_{pf}	Number of particles matched
N_{pm}	% of particles not found
$N_{pm}^{'}$	% of spurious particles
N_{pp}	Total number of pixels found in a particle
P	Probability density function
P_b	Probability density function of the background pixels
P_p	Probability density function of the particle pixels
P_{I_θ}	Probability of I_θ
R	Particle radius
R_v	Vortex core size

$S(x,y)$	Smoothed image
ud	A uniform distributed deviate with values between 0. and 1.
V	True velocity
V'	Calculated velocity
x	X coordinate
X_p	True X coordinate of center of a particle
X_p'	Calculated X coordinate of center of a particle
y	Y coordinate
Y_p	True Y coordinate of center of a particle
Y_p'	Calculated Y coordinate of center of a particle

Greek

Δt	Time between successive images
μ_b	True mean of background levels
μ_b'	Calculated mean background level distribution
$\mu_{location}$	Mean error in particle location
μ_p	True mean particle level
μ_p'	Calculated mean particle level distribution
μ_r	Mean particle radius
$\mu_{velocity}$	Mean error in particle velocity
σ_b'	Calculated standard deviation of background level distribution
σ_p'	Calculated standard deviation of particle level distribution
$\sigma_{location}$	Standard deviation of error in particle location
$\sigma_{velocity}$	Standard deviation of error in particle velocity

CHAPTER I

INTRODUCTION

1.1 LITERATURE REVIEW

Researchers have been investigating turbulent flows for more than a century, but no general approach to the turbulence problem exists. Considerable progress has been made by a combination of theoretical, numerical and experimental developments as well as a great deal of clever empiricism. It is clear that no simple approach is likely to unravel the key to understanding turbulence, but further progress must come from an interactive approach based on improved theory and numerics guided by ever more sophisticated experiments. The next few paragraphs describe briefly the various approaches used to study turbulence.

1.1.1 THEORETICAL METHOD OF ANALYSIS

In developing a theory for turbulence, there are three major approaches which have received considerable attention, namely the statistical, phenomenological and numerical approaches.

In the statistical theory of turbulence all quantities are decomposed into the sum of a time averaged part and a fluctuating part and then substituted into the Navier-Stokes equations and the mass and energy conservation laws. The equations are then averaged in order to obtain a set of equations that describes the mean behavior of turbulent flows (Tennekes, 1972). Unfortunately, the non linearity of the equations results in more unknowns than governing equations. Further relationships can be developed using various moments of the

conservation laws, but these additional manipulations inevitably introduce still more unknowns. This is the "Closure Problem" of the Statistical Theory of Turbulence. To attack this problem, several approaches have been suggested: a mathematical approach, where arbitrary simplifications of the existing equations are made by dropping terms or by expressing them in convenient forms suggested by analogy with laminar flow; a physical approach where specific models of turbulent activity were used to establish new relationships; and a combined approach where both mathematical and physical arguments are used (Frost and Mouldin, 1977).

Phenomenological or semi-empirical theories combine heuristic arguments and empirical data to predict the gross phenomena arising from turbulent activity. The goal of the phenomenological theories is to obtain a functional dependency for the Reynolds stress directly. Such theories are: Boussinesq's Theory, Prandtl's Mixing Length Theory, Taylor's Vorticity-Transport Theory, Von Karman's Similarity Hypothesis, etc. The phenomenological theories assume some simple mechanism for turbulence and then derive from this the desired relation for the Reynolds stress. Unfortunately, none of them gives a complete physical picture, nor a comprehensive mathematical model (Brodkey, 1967).

Conceptually, numerical techniques can be used to analyze almost any flow field. In a turbulent flow field, where both small and large scales of motion coexist, a mesh size small enough to resolve the smaller scales is needed. At the same time, the number of grid points has to be large enough in order to include the large scales of motion as well. Considering that the difference in magnitude between the small and large scales for high Reynolds numbers can be of the order of 10^4 to 10^5, the resulting number of grid points needed to solve the problem is extremely large and makes this approach unfeasible except for very low Reynolds numbers. An example of such flow field is the turbulent channel flow studied by Moin and Kim (1986). Their direct numerical simulation required 3,962,880 grid points (192 X 129 X 160 in x,y,z) for a Reynolds number of 3200. Even the best and fastest computers today are not capable to handle similar Navier-Stokes direct simulations with the number of grid points required for engineering Reynolds numbers.

Although the recent theoretical and computational developments of turbulence have provided many insights into its mechanisms, the majority of the researchers today is still using experiments to validate existing theories and computations and develop a conceptual understanding of the various turbulence mechanisms.

1.1.2 EXPERIMENTAL RESEARCH

The shortcoming of all the methods and techniques described above has put a great deal of emphasis on experimental research. Researchers have been using both qualitative methods such as flow visualization and quantitative methods such as hot wire anemometry, laser doppler velocimetry, streak photography, pulsed laser velocimetry, and various other imaging techniques, to try to understand the mechanisms of turbulence.

Flow visualization techniques provide valuable information about the behavior of the entire flow field but are not usually intended to provide quantitative velocity data. Using flow visualization, the fluid in a large region can be simultaneously observed. This allows the researcher to get a vivid, global and detailed picture of the physical process taking place in the flow, but the interpretation of such data is very subjective. Nevertheless, flow visualization has been a useful tool in the hands of turbulent flow researchers for many years and has brought many breakthrough discoveries. Examples of such discoveries include: the ejections and bursts that occur locally and randomly in space and time in a turbulent wall flows, (Corino and Brodkey, 1969 and others), the hairpin vortices in the boundary layer (Head and Bandyopadhyay, 1981) and the coherent structures which dominate the mixing layer (Brown and Roshko 1974). Other excellent studies were done by Offen and Kline (1974) where they used two basic flow visualization techniques, dye injection and hydrogen bubble generation to study the turbulent boundary layer over a flat plate, and by Nychas, Hershey and Brodkey (1973) where they used high speed photography to record the motion of very small solid particles in the outer region of a turbulent boundary layer along a flat plate. In 1978, Praturi and Brodkey used stereoscopic photography and they identified the axial vortex motions in the turbulent wall region because they were able to catch the three-dimensional aspects of the

flow. Up until recently, very little quantitative information could be extracted from the flow visualization images because of the time required to manually extract the data.

For more quantitative measurements, hot wires have been used in many different configurations by researchers for a long time. The hot wire anemometer is basically a thermal transducer. It consists of a very thin, short, metal wire heated by electrical current. Its operation relies on the fact that the flowing fluid cools the wire, causing a temperature drop and that temperature drop is related to the instantaneous velocity of the flow field at the point of the measurement (Hinze, 1975). The hot wire system has two modes of operation. The first is the constant current mode, where the current in the wire is kept constant and variations in the wire resistance caused by the flow are measured by monitoring the voltage drop variations across the hot wire, the second is the constant temperature mode, where the metal wire is placed in a feedback circuit which tends to maintain the wire at constant resistance and hence at constant temperature (Perry, 1982). While being widely used, hot wire anemometry suffers from many drawbacks. The small size of the wire (of the order of 5 micrometers in diameter) makes it unsuitable for use in harsh environments such as combustion flows (Goldstein, 1983) and it can not be used to study practical problems such as separated flows with high turbulence intensities and flow reversals (Goldstein, 1983). Hot wire anemometers are also sometimes inaccurate to obtain measurements very close to a solid boundary. Multiple hot wires can be used simultaneously to determine multiple velocity components but such probes produce more disturbances in the flow field and the measurement volume becomes large with respect to the smallest turbulence scales. Furthermore, measurements are made at a single point in space and spatial information would be useful to further understanding of turbulence.

The Pulsed Wire Anemometry technique was developed independently by Bradbury and by Tombach in 1969 (Bradshaw, 1971). In essence, the probe consists of three thin wires mounted short distances apart. The two "receiver" wires are mounted one on either side of the "transmitting" wire and at right angles to it. The "transmitting" wire is heated periodically with a pulsed current and the resulting streaks of heated fluid are convected downstream by the flow. The "receiving" wires operate in the same manner as a normal hot wire anemometer. The time elapsed between the release of a hot streak by the "transmitting" wire

and its detection by either the "receiver" wires is recorded electronically and then converted into velocity. The sign of the velocity is determined by noting which of the two "receiver" wires receives the pulse (Bradshaw, 1971). This technique is independent of the properties of the fluid and gives direct measurement of the velocity. It can also be used to measure velocities in very high turbulence regions. However, it suffers from large probe interference, poor spatial resolution and a very poor frequency response because the time interval between pulses can not be less than the smallest passage time between wires.

The first Laser Doppler Velocimeter (LDV) was described by Yeh and Cummins in 1964 and since then, there have been numerous developments of both the optics and the processing electronics of the system (Drain, 1976). The LDV technique utilizes the doppler effect to measure instantaneous velocity of small particles embedded in the flow. When a particle passes through the measurement volume formed by focusing two laser beams, it scatters the light. The scattering light is at a different frequency from that of the incident beam and the particle velocity can then be determined by measuring the Doppler shift of the scattered light. LDV presents some advantages over the Hot Wire Anemometry techniques, such as, no flow interference (since the probing is purely optical), direct velocity measurement without calibration regardless of the velocity field.

Furthermore, the technique is quite accurate, but its spatial resolution is not very good and its costs are high. The technique relies on seeding the flow with small scattering particles.

To overcome the limitations of single point measurements, new techniques have been devised which draw from the advantages of flow visualization while providing quantitative information about the spatial structure of the flow. The development of relatively inexpensive digitizing cameras and digital computers as well as the development of elaborate software has now allowed the researchers to develop techniques to extract full field, quantitative information from flow visualization of images.

Streak photography is the oldest and simplest method of experimental multi-point velocity measurements. The flow is seeded by particles and the fluid velocity as function of time can be measured by tracking the particles down through a succession of multiple exposure recordings or multiple frame techniques. Dimotakis et al. (1981) used sawdust as tracer

particles and using streak photography they investigated a turbulent shear layer by measuring two components of the velocity flow field. Sheet lighting is necessary to define the plane of fluid motion. This can present a problem due to the fact that when the particles move out of the light sheet in a three-dimensional field, their streaks are wrongly interpreted as low velocities. A solution to this problem was given by Kobaybashi and Yoshitake (1985), where they used fluorescent particles stimulated by a light pulse. Because of the fluorescence, the particles did not have to remain within the light sheet and their fluorescent tails could be observed for the complete duration of the exposure, enabling the determination of their velocities and directions (Adrian, 1986). Velocity measurements using streak photography have a large associated error when the length of the streak is small. Also, when the mean distance between tracers is of the same order of magnitude as the distance a tracer particle travels during the exposure time, streak photography fails to provide accurate results. Thus, the tracer concentration is usually kept small resulting in velocity measurements with poor spatial density. Involved interpolating methods have been designed by Kobaybashi et al. (1983) in order to increase spatial resolution of the technique, but their validity is questionable if the distance between the existing data points is larger than the turbulent flow scales (Lourenco, 1986). Multiple frame techniques such as conventional motion pictures, or multiple exposures on the same frame can be used. Nishino, Kasagi and Hirata (1985) filmed the motion of small patches of hydrogen bubbles in a turbulent boundary layer.

Pulsed Laser Velocimetry (PLV) is a technique which uses pulsed laser photography of many particles undergoing short displacements in order to extract velocity information from the flow field (Adrian, 1984). The advantage of the technique is that the velocity field can be measured over an entire plane of the flow field simultaneously with good accuracy and spatial resolution. Hence, this technique has the best potential for turbulent flow research. PLV has two distinct modes: Laser Speckle Velocimetry (LSV) and Particle Image Velocimetry (PIV). The first application of the speckle phenomena were in the field of solid mechanics (Erf, 1980). To apply this technique to fluids, the flow is first seeded with a large number of small tracer particles and then illuminated with a sheet of coherent light. The light is scattered by the seeding particles in the illuminating plane provides a moving pattern. When the seeding concentration is large, the images overlap, interfere constructively or destructively

and produce a random speckle pattern which moves with the fluid. The motion of the pattern is then determined by taking double exposure photographs. This photograph or specklegram contains two correlated grids which can be analyzed as a non-uniform diffraction grating. The technique can only be used when the displacement between exposures is greater than the speckle size but not so great as to destroy the correlation. The fundamental concept underlying (LSV) is that the speckle patterns translate with the scattering sites that created them, so that the measurement of the speckle motion can be correlated to the measurement of the scattering site motion. Because the speckle photograph is too random for the speckle displacements to be discovered by inspection, in 1968 Burch and Tokarski developed a Young's fringe technique where by scanning the double exposed photograph and determining the particle displacements, one can resolve the two components of the velocity vector at every point of the fluid (Adrian, 1986). However, the direction of the displacements though, is ambiguous unless the first and second exposures can be identified. Adrian (1986) has developed an image shifting technique to resolve this directional ambiguity for both modes of PLV. Particle Image Velocimetry (PIV) is applied when the scattering sites concentrations are low enough and produce individual particle images rather than laser speckles. The motion of the particles is again recorded photographically. Adrian (1986) has argued that in most fluid mechanical applications of PIV the scattering concentrations would be low enough to produce individual particle images rather than laser speckles. Double pulsed PIV was used by Lourenco and Whiften (1984) in studies of jet flows. Also, some work at the University of Illinois used double pulsed ruby lasers for PIV studies of turbulent thermal convective motions (Adrian, 1984). The quality of the recorded images governs the performance of PIV. Specific parameters like the tracer particles (type, dimension and concentration); exposure parameters (duration of exposure, time between exposures and number of exposures); film parameters (sensitivity, grain size and resolution); recording optics (magnification and aperture), play significant roles in the quality of the recorded images (Lourenco and Krothapalli, 1987). The overall accuracy of the technique depends on the accuracies which can be achieved in the photographic procedure and processing techniques. Work to study and improve the photographic procedure was done by Russ and Brodkey (1988). The lighting was varied to obtain the best images of the particles and the grid, an effort was put in minimizing

the obstruction of the grid used and the visibility of the particles was investigated. Also, the color of the particles was analyzed by digitizing the images and obtaining the average R, G and B values for them. In 1985, Chang et al. applied PIV to study turbulent flow in agitated tanks and observed an error on particle positions in X and Y directions of about 0.4 pixels on the average. Their velocity calculation produced an error of about ± 2.0 cm/sec in both directions. Landreth et al. (1987) report an overall RMS error of 2.4 % and 2.2 % in the two orthogonal directions of the velocity in a solid body rotation flow field. Majumdar et al. (1987) applied PIV on a seeded three dimensional shear flow and reported a maximum radial distance error between the actual and predicted flow field control points used of about 2.5 %/cm.

Imaging techniques are used after an image has been obtained using one of the methods described above. They are basically divided into two categories: Image Processing, which is concerned with the generation of new images from already existing ones (filtering, noise reduction, edge enhancement, etc.) and Image Analysis, which is the task of obtaining useful information from an image. Image analysis is subject to the individual interests of each researcher. During the initial stages of the application of Imaging techniques, Dimotakis et al. (1981) used a single threshold to identify the particles in black and white images obtained with streak photography. Sheu et al. (1982) developed an experimental technique based on stereoscopic motion pictures of flow phenomena occurring in an agitated tank containing tracer particles. Using frame to frame analysis of the scenes, the X and Y coordinates of the same tracer particles in each scene were obtained. The Z coordinate (depth) was determined using geometry. This technique is impractical and very subjective because of the manual labor involved. In 1985, Chang et al. presented an extension of the study done by Sheu et al.. In this work, the manual operation described by Sheu et al. was done automatically on digitized black and white images. They developed particle identification and tracking algorithms and used them to calculate velocities. The selection of the threshold was done manually after various comparisons were made at different threshold levels by verifying particle images identified by the software with those appearing on the computer screen. Filtering methods were not used. A priori knowledge of the flow field was used in order to devise constraints for the particle tracking algorithm. In 1987, Majumdar et al. used a combination

of image processing and numerical least squares determination of the three coordinates on a two-dimensional stereoscopic picture to automatically identify and track single or multiple particles in a three-dimensional shearing suspension space. The particle identification algorithm was applied in all sequential frames and using the time elapsed between frames, they determined the instantaneous velocity distribution. A priori knowledge of the particle locations was used for the threshold calculation, then a "mask" was calculated for every particle and by convolving masks of individual particles from frame to frame the particles were matched. Lakshmanan and Brodkey (1986) seeded turbulent flows with three color particles (red, white and blue) and by using a modified particle identification algorithm (a modification of the algorithm developed by Chang et al.) along with data reduction techniques they were able to track particles from frame to frame. Economikos and Brodkey (1988) studied the variables involved in the digitization process and developed techniques and algorithms to optimize them. Then using RGB and HSV models to define the color of pixels, they were able to identify tracer particles in arbitrary flow fields. Filtering and color separation techniques were applied. The two-dimensional stereoscopic coordinates of the particles were transformed into three-dimensional and the particles were tracked in the first three frames. By using the information produced by the three sequential frames, they formulated a predictor corrector method to automatically track the particles in the rest of the frames.

1.2 OBJECTIVES

Many researchers have started to use imaging techniques to extract quantitative information from digitized pictures of particles embedded in turbulent flow fields. Their efforts were concentrated into two main directions. First, parameters such as lighting conditions, contrast between the color of the background and the color of the particles, material, size and color of the particles, quality of the camera, filming speed, etc., were all investigated to get the best possible picture. It was found that high intensity, highly collimated light sources such as lasers, state of the art camera equipment (with the least amount of aberrations induced), improved the picture to a great extend. High resolution digitization equipment can now

produce an almost identical digital image to the analog one with little noise and good dynamic range. Second, most efforts were devoted to develop the most efficient imaging techniques in order to secure highly accurate results. Techniques that enhance the image such as noise reduction, smoothing, edge detection, spatial and frequency domain processing etc., have been able to improve the picture and consequently improve the result.

During the entire development of these techniques for turbulent flow research, there has been little work done on quantifying the effects of the various flow and image parameters on the accuracy of the results. For instance, it would be useful to systematically quantify the influence of the background noise and how noise reduction techniques improve the accuracy of the results as well as determine the effect of particle size and particle density. Most analyses of the errors are posteriori estimates obtained by indirect comparison with other techniques. This is linked to the fact that the exact velocity and position of particles in real flows is not known and errors can only be inferred. Furthermore, researchers usually have little control over the image parameters, although they always strive to obtain the "best" possible images.

The objective of this research is to address the questions described above. By using synthetic images rather than real ones, all the parameters involved can be controlled and varied as needed in order to systematically study their influence. Furthermore, the use of synthetic images allows an exact quantification of the errors involved since the velocity fields and particle locations, sizes, etc., are known exactly. However, in order to validate the results obtained by this procedure, the synthetic images must be representatives of a broad class of real images and a large number of images must be processed to obtain reliable statistics of the results. Hence the objectives of the research were fourfold:

--- First, generate "realistic" images with controlled parameters such as contrast, noise, number of particles, particle size distributions, etc.;

--- Second, develop algorithms for the automatic processing of such images to detect particles and determine their position in single images;

--- Third, develop algorithms to match particles across successive frames to infer velocity information.

--- Fourth, quantify statistically the influence of the various parameters on the accuracy of the results.

In such a study, it is imperative that the process be entirely automatic to avoid influencing the results with operator (subjective) decisions.

CHAPTER II

FORMULATION OF THE PROCEDURE

This chapter is a step by step description of the procedure used in order to meet the objectives of this study. The programs which implement these algorithms are listed in the appendix A and B for reference.

2.1 GENERATION OF IMAGES

Images were synthesized to represent the digitized output of a hypothetical real picture with particles embedded in some kind of a flow field. For the purpose of this research, the two main variables of the digitization process were chosen to be 256 X 256 for the resolution and 8 bits per pixel for the pixel depth. Resolution is a direct result of the digitization camera used and is associated with the number of scan lines and the number of pixels per line. The higher the resolution for an image of a given size, the clearer the digitized picture is, which results in a more accurate identification of the location of the particles and a more accurate calculation of their velocities. Each generated image is represented as a 256 X 256 array of digital values I(x,y). Each value in this array represents the intensity of the corresponding pixel on the real picture. Its relative position in the array corresponds to its pixel position in the actual image. While this resolution is low by today's standards and was chosen for ease of implementation of the computer, it is representative of higher resolutions without any loss of generality. In other words, the 256 X 256 image can be thought as a subset of an image digitized with a higher resolution. Further more, the software could easily be modified to accommodate larger arrays. Pixel depth is the number of bits per pixel. Assuming 8 bits per pixel results in $2^8 = 256$ significant grey levels. Most digitizers are

capable of at least 8 bits per pixel and often more, although the higher pixel depth is often plagued by noise in the lower bits while reduces the effective dynamic range to 8 bits per pixel.

The synthetic images were generated to mimic a broad class of real images obtained from the actual implementations of the PIV. Images can be thought of a certain number of particles randomly sprinkled over a background. The noise can be modeled as being gaussian additive noise. In a realistic case, the experimental set up, photographic process, digitizer and even the environment where the picture is taken can be possible sources of noise. Illumination and distinct gray level difference between the particles and the background are the major parameters of setting up a "good" contrast. Even though the particles may be of the same size and shape, they can appear in the picture as having different sizes and shapes depending on their orientation on the plane where the picture is taken and the possible existence of shadows due to other particles. For the purpose of this research all the particles generated are circular with a normal distribution of radii. Based on this broad set of assumptions, every image is generated according to a specified set of parameters. These include the number of particles in the image (N_p), the mean grey level intensity of the particle pixels (μ_p), the mean grey level intensity of the background pixels (μ_b), the mean radius of the particles (μ_r), the standard deviation of the particle radius (A_r) and the standard deviation of the image noise (both particle and background) (A). The difference between (μ_p) and (μ_b) represents the contrast (C) for that image.

Figure 1a shows a typical example of a generated image with the following parameters: $N_p = 100$, $\mu_p = 228$, $\mu_b = 28$, A = 10.0, $\mu_r = 3.5$, $A_r = 1.0$. Although 100 particles were generated, only 84 distinct particles can be found, because some particles overlap with each other. This overlap is not an antifact of the synthetic images and is representative of the fact that the illumination of the flow is usually done with a finite width (although small) light sheet, hence illuminating particles belonging to different planes of motion. This effect is usually minimized by the use of collimated laser light sheets, but is still present in real images. The allowable range of coordinates for the particle centers in the X and Y direction is [1,256]. Since physically all values of the allowable X coordinates (X_p) and Y coordinates (Y_p) are "equally likely" to occur, the particle center coordinates were generated by a random

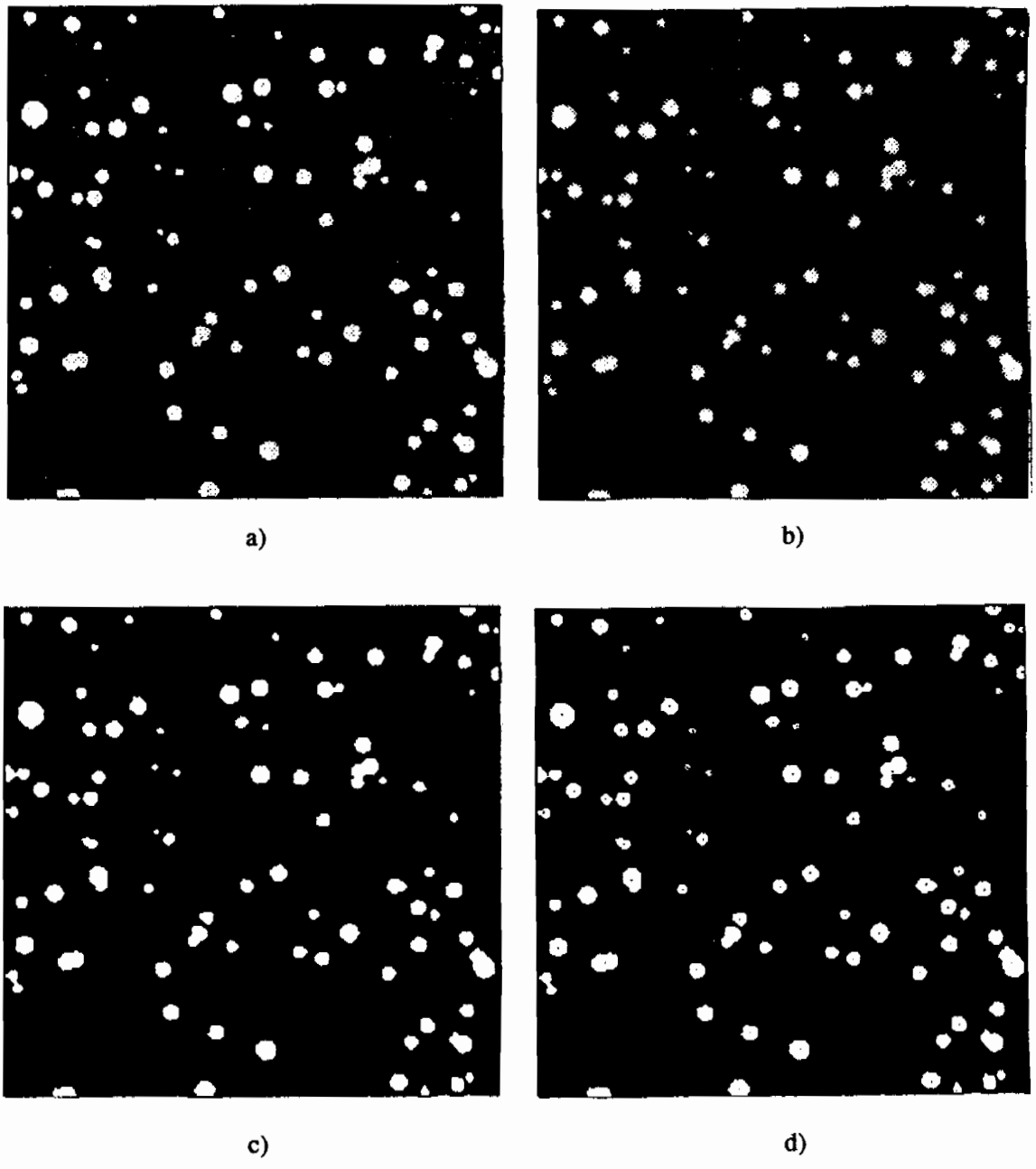


Figure 1. Synthetic and processed images for C = 200, A = 10, N_p = 100 and R = 3.5 ± 1.0; a) raw image, b) after filtering, c) after thresholding and d) after locating the particles

number generator (Press W. H. et al., 1986). Specifically the coordinates of the centers of each particle are calculated as follows:

$$X_p(\mathrm{i}) \text{ or } Y_p(\mathrm{i}) = ud \times 256$$

where (ud) is a uniform randomly distributed deviate which takes values between 0.0 and 1.0 and X_p with Y_p are the floating point coordinates of the particle (i). The generated gray intensity levels of the particle pixels $I_p(x,y)$, the gray intensity levels of the background pixels $I_b(x,y)$ and the radii of the generated particles (R), are set as being normally distributed around their mean values. For that reason, a Gaussian probability function is used with zero mean and unit variance (Press W. H. et al., 1986). Specifically, the background pixel levels are generated according to:

$$I_b(x,y) = \mu_b + A \times ndd$$

Similarly, the level of the particles is set as:

$$I_p(x,y) = \mu_p + A \times ndd$$

and the radius of the particles is set as:

$$R = \mu_r + A_r \times ndd$$

where (ndd) is a normally distributed deviate with zero mean and and unit variance. In case that any of the levels mentioned above becomes negative or greater than 255, it is set to 0 or 255 respectively. If any particle coordinate becomes negative or greater than 256, it is set to 0.8 and 255.8 respectively. A constraint is also set in the radius of the particles to never become less than 1.5 pixels. It was found that particles with radii smaller than 1.5 pixels

tend to disappear from the image because of the information loss after processing. This limitation would be met in practice by choosing the level of magnification required for the digitization of the images. Figure 2a shows an image with the same number and size of particles as the image in Figure 1a, but for $\mu_b = 78$, $\mu_p = 178$ and A = 30.0. A similar image is displayed in Figure 3a for $\mu_b = 103$, $\mu_p = 153$ and A = 40.0. A comparison between these three Figures shows that decreasing the contrast and increasing the noise level makes the identification of the particles progressively worse.

2.2 IMAGE PROCESSING

The goal of the image processing part of this research is to generate an image simple enough so that the computer can identify and locate all the particles in it without any ambiguities. To meet this goal, the processing used includes histograming, smoothing and thresholding.

2.2.1 Histograming

A histogram of an image is a plot of the number of pixels at a given intensity level versus the intensity level, or in other words, a histogram is a plot of the probability density function (P) of the pixel levels, since the levels of the image I(x,y) are selected randomly (Faugeras, 1983). The histogram of the image in Figure 1a is shown in Figure 4a. This particular histogram shows two distinct distributions. The first distribution corresponds to the pixels that belong to the background and the second distribution corresponds to the pixels that belong to the particles. The purpose of calculating the histogram of an image is to identify which intensity levels belong to the background and which belong to the particles. Figure 4a illustrates the difficulty in dealing with such images. Since the particle density is low, the particles occupy a very small percentage of the total picture area and therefore the

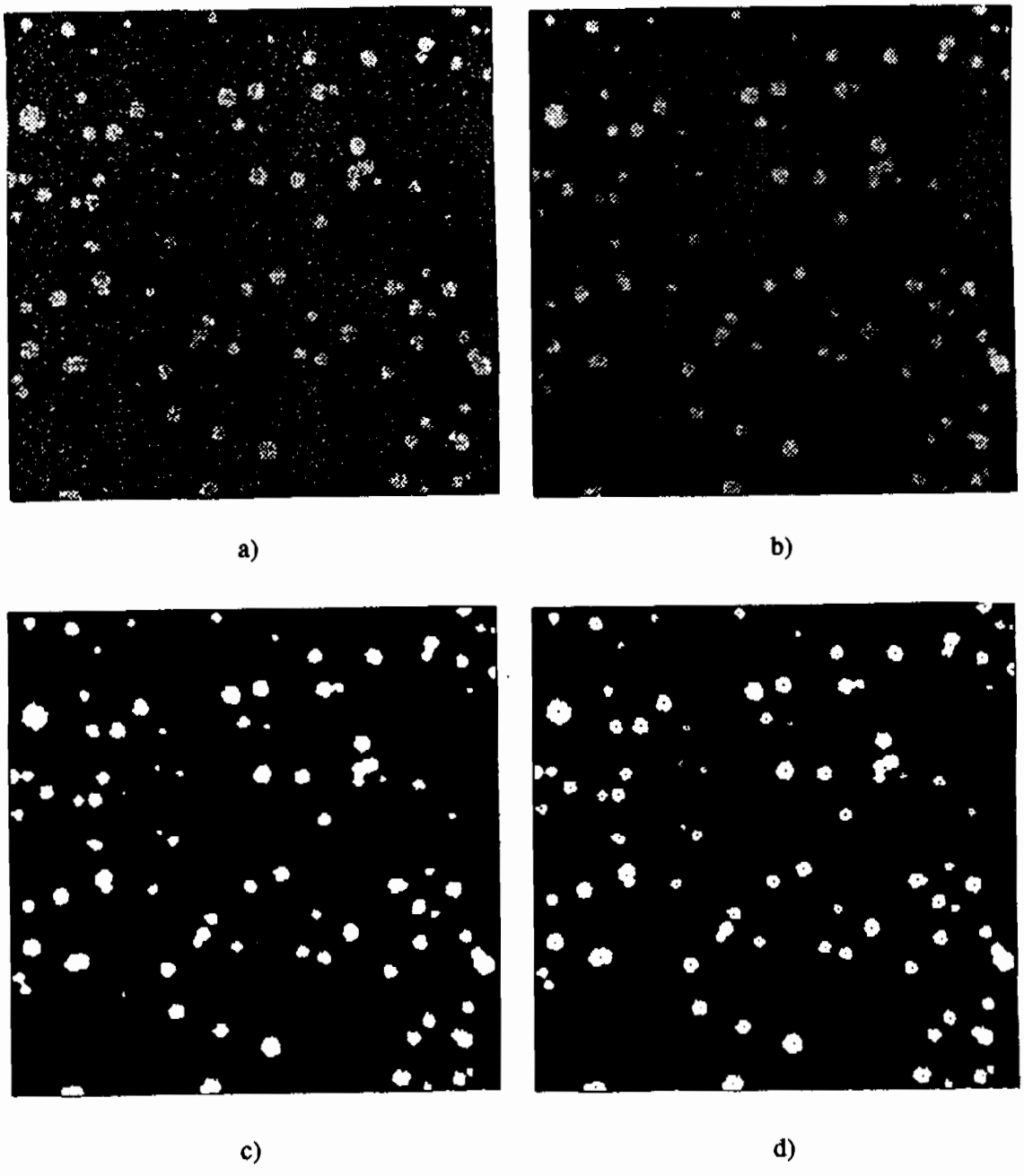


Figure 2. Synthetic and processed images for C = 100, A = 30, N_p = 100 and R = 3.5 ± 1.0; a) raw image, b) after filtering, c) after thresholding and d) after locating the particles

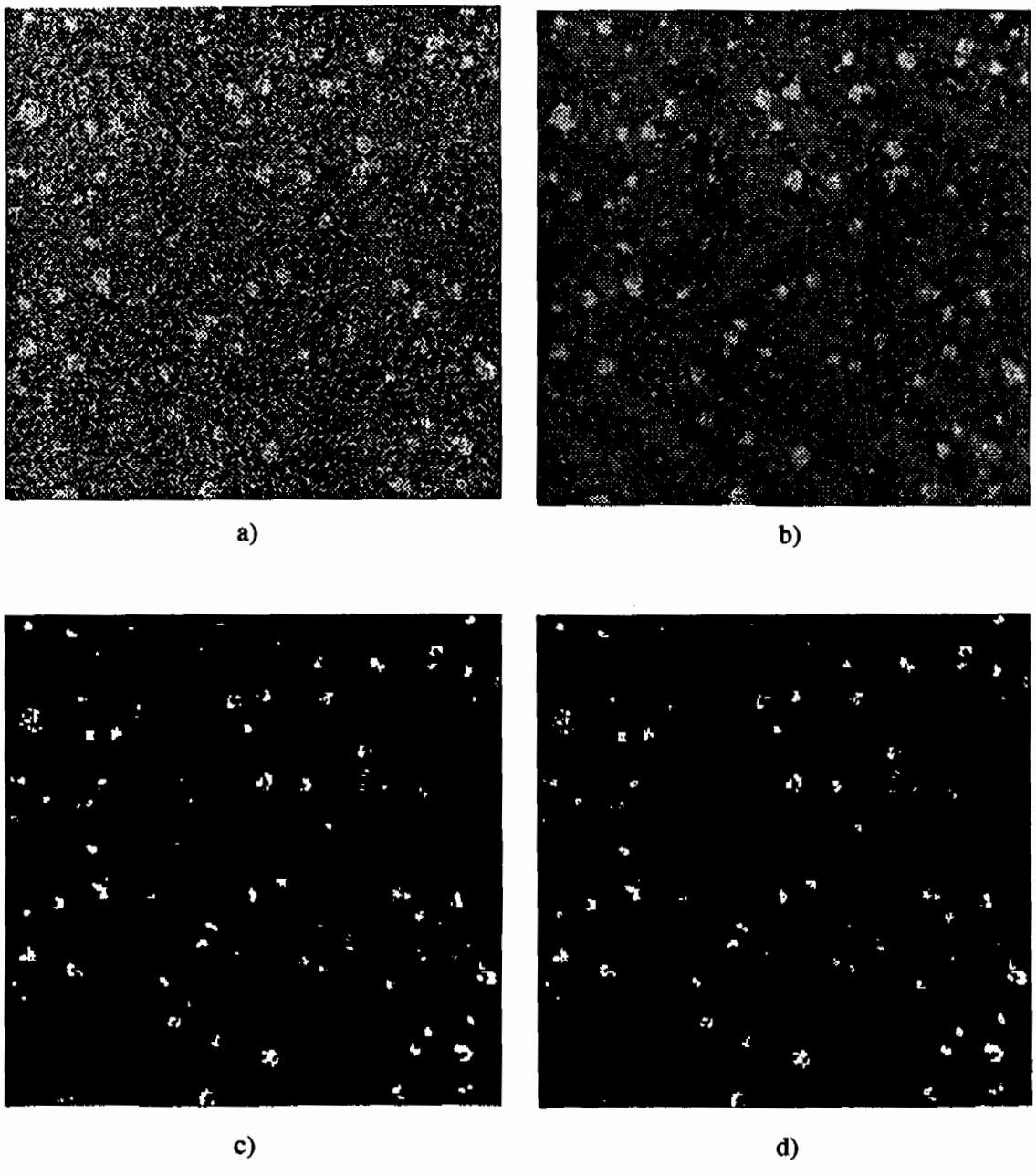


Figure 3. Synthetic and processed images for C = 50, A = 40, N_p = 100 and R = 3.5 ± 1.0; a) raw image, b) after filtering, c) after thresholding and d) after locating the particles

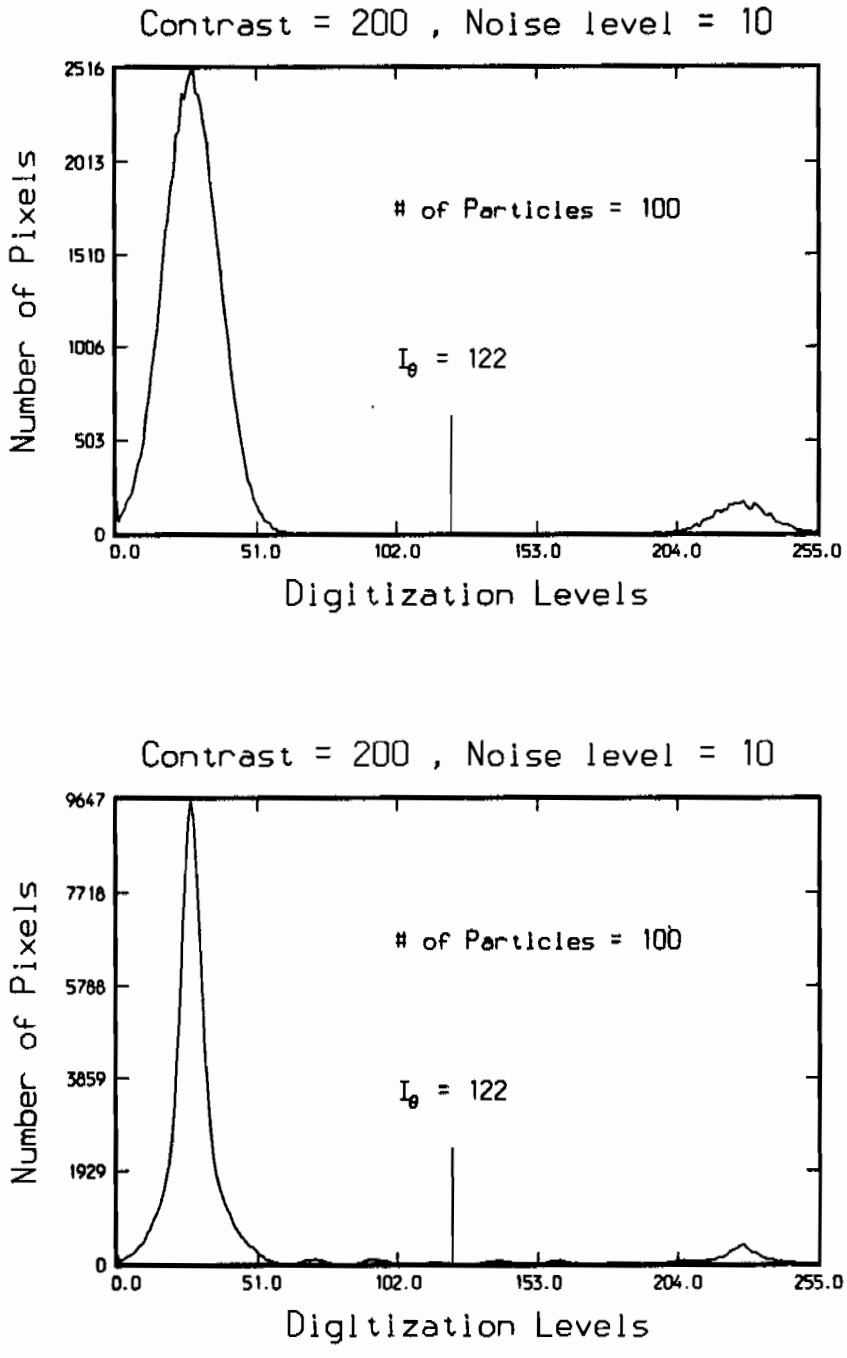


Figure 4. Histograms of image for C = 200, A = 10, N_p = 100 and R = 3.5 ± 1.0 highlighted with threshold value; a) before filtering and b) after filtering

amplitude of the distribution of the particle levels is very small compared to that of the background. The relative heights of the two distributions depend on the density and size of the particles and the noise level of the image. As the contrast (C) decreases and the image noise (A) increases, the two distributions overlap and the particles are lost in the noise of the background.

In an attempt to separate the two distributions, the histogram is assumed to be modeled by the sum of two gaussian distributions. The mean, which is the "center" of a probability distribution in the sense of an average and the standard deviation, which is a measure of the "spread" of a probability distribution, are found for both distributions by curve fitting (Miller, 1977). A mean of the background distribution (μ_b') can be first approximated as the intensity level which corresponds to the highest point in the probability distribution curve. Then, the standard deviation (σ_b') for that distribution is found by least square fitting the distribution to a gaussian distribution with μ_b' as its mean. After σ_b' is found, the fitted distribution representing the background pixels can be calculated as:

$$P_b = a_b \times \exp\left[-\frac{(I-\mu_b')^2}{\sigma_b'^2}\right]$$

Next, all the intensity levels in the region $\mu_b' - 5 \le I \le \mu_b' + 5$ are investigated as possible means and a search starts to find the mean value which minimizes the error of the curve fit. For each possible mean, the sum of the error differences $(P(I) - P_b(I))^2$ for 21 intensity levels (10 levels to the right and 10 levels to the left of each I) is formed. Whichever intensity level produces the minimum sum of the error differences is selected as the calculated mean (μ_b') and its standard deviation is selected as the calculated standard deviation for the background distribution. After the mean and the standard deviation of the background level distribution are found, the distribution P_b can be approximated using the above formula.

To lessen the overwhelming effect of the background distribution in the total image histogram, P_b is then subtracted from P. The remainder is the probability distribution of the

particles (P_p), plus some "noise" because P_b does not match P exactly. Sometimes, the "noise" left may be greater than the relative height of P_p, so the search for the mean and standard deviation of P_p is limited to the region $(\mu_b' + 1.5 \times \sigma_b') \leq I \leq 256$. Due to the low amplitude of the particle distribution and the noise of the histogram, an iterative procedure was designed to search for the mean of the particle levels. The curve is smoothed by neighborhood averaging until only one maximum remains which is selected as an estimate of the the particle mean μ_p'. The standard deviation σ_p' is then determined by least square fit of the distribution around the mean. The procedure is repeated iteratively for various values of the mean as before until the error is minimized. Figure 5a shows the histogram of the image in Figure 2a and Figure 6a is the histogram of the image in Figure 3a. As the contrast decreases, the location of the means of the two distributions approaches each other. As the noise level increases, the spread of a distribution over the intensity levels increases and distinguishing the two distributions becomes harder and finally, the distributions are indistinguishable as in Figure 6a in a high noise, low contrast case.

2.2.2 Smoothing

Smoothing is a way of filtering an image. Filtering refers to techniques which try to enhance or suppress certain features of the image using different transformations on the image intensities. Smoothing operations are used for diminishing spurious effects that may be present in the digital image and can be done in the spatial or the frequency domain (Levine, 1985). For this research, a spatial smoothing called neighborhood averaging is used. For a 256 X 256 image I(x,y), a smoothed image S(x,y) is generated by replacing each individual pixel by the average of all the pixel values in a predefined neighborhood A(x,y). The neighborhood A(x,y) is a subset of I(x,y). In other words the entire image is convolved with a matrix of the form:

$$A(n,m) = (\frac{1}{9}) \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

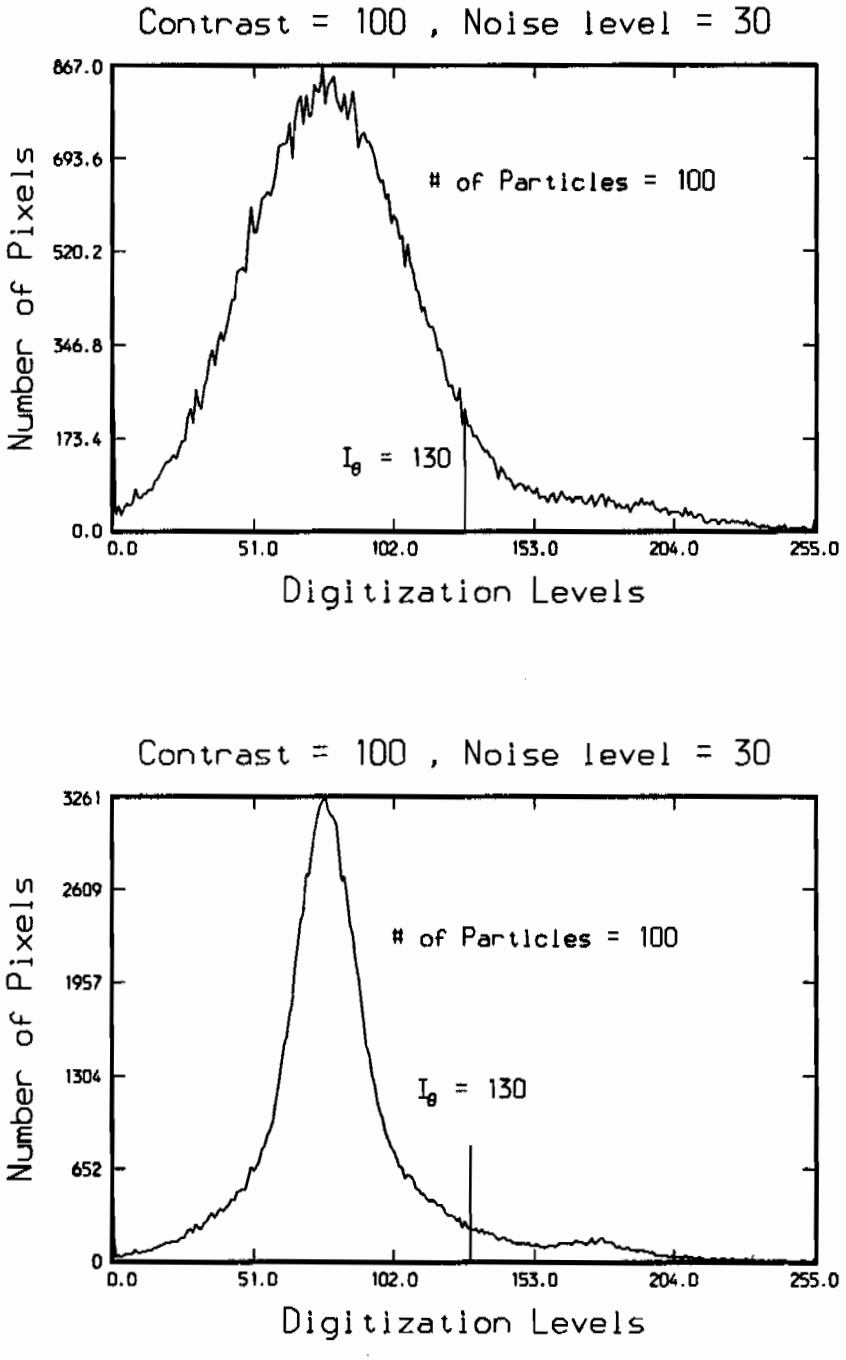


Figure 5. Histograms of image for C = 100, A = 30, N_p = 100 and R = 3.5 ± 1.0 highlighted with threshold value; a) before filtering and b) after filtering

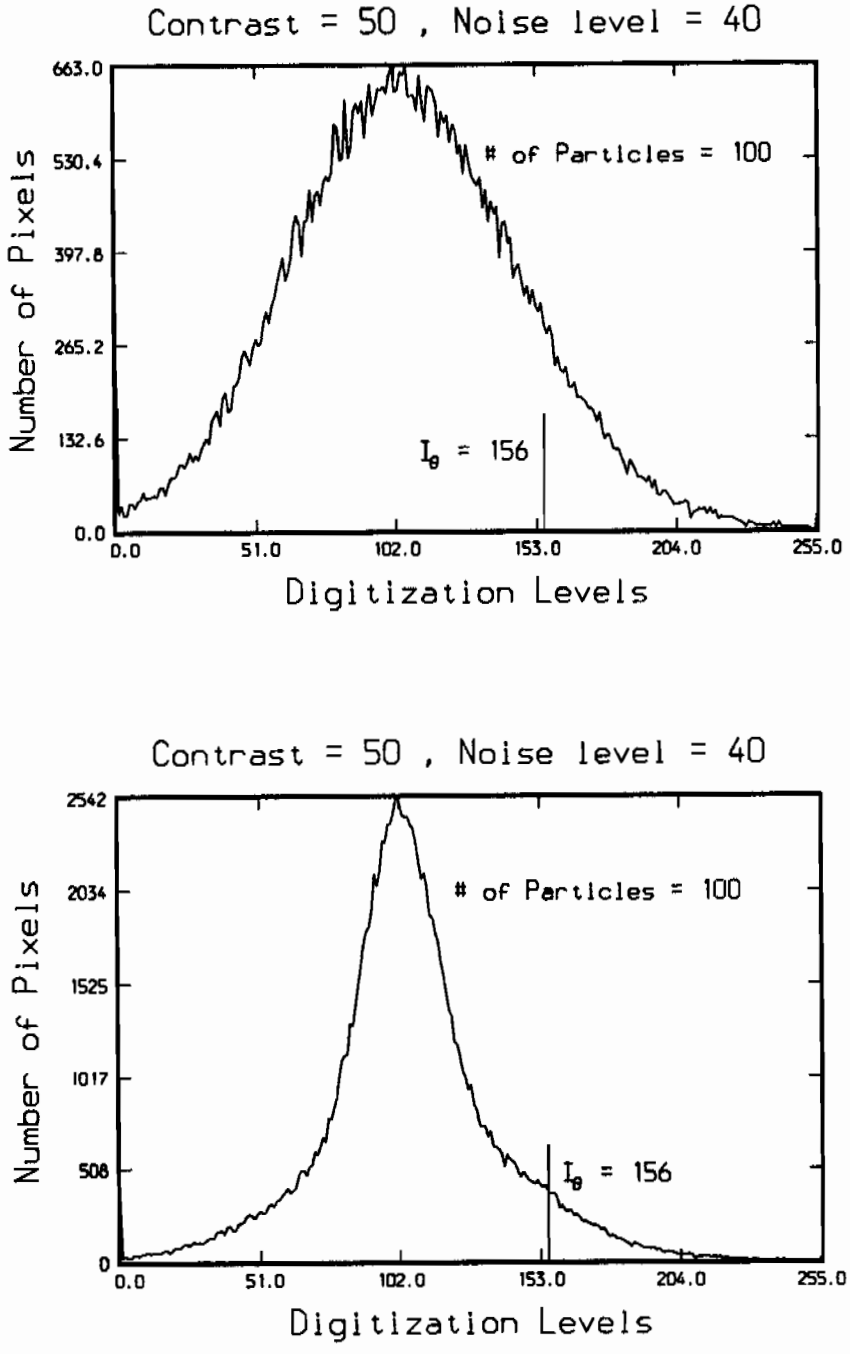


Figure 6. Histograms of image for C = 50, A = 40, N_p = 100 and R = 3.5 ± 1.0 highlighted with threshold value; a) before filtering and b) after filtering

In digital form the operation can be written:

$$S(x,y) = \left\{ \sum_{n=-1}^{n=1} \left[\sum_{m=-1}^{m=1} I(x+n,y+m) \times A(n,m) \right] \right\}$$

for every point (x,y) in the original image I(x,y). Since the noise in the picture tends to be pixel noise rather than low frequency noise, a 3 X 3 neighborhood is used. This choice is a compromise between the noise reduction and inherent blurring resulting from the use of such filters. Because of geometry limitations, the neighborhood averaging is applied only to pixels with coordinates $2 \leq x \leq 255$ and $2 \leq y \leq 255$. The intensity levels of the pixels located along the sides of S(x,y) are set equal to the corresponding intensity levels of the pixels in I(x,y). Figures 1b, 2b and 3b show the smoothed images that resulted from applying the neighborhood averaging technique on the images of Figures 1a, 2a and 3a, respectively. Comparing these three images, it is obvious that the noise in the image is substantially decreased, but at the same time the edges of the particles have lost their initial contrast and become a little bit blurred. Blurring is proportional to the size of the neighborhood used. After smoothing, the information left on the image is always less than before smoothing. Figures 4b, 5b and 6b show the histograms of the smoothed images of Figures 1b, 2b and 3b, respectively. The removal of noise by smoothing can be clearly identified by comparing the histograms of the smoothed images to the histograms of the unsmoothed ones. The spread of the probability density curves of the background and particles has decreased, rendering the separation of the background and particles more easy.

In this research, only images with relatively high noise levels were smoothed. An image was considered to be noisy if its probability density distribution curve assumed non zero values for 10 percent or more of the 256 available intensity levels.

2.2.3 Choosing a Threshold and Thresholding

Thresholding an image is the process by which the image image pixels are categorized into two classes. For the purpose of this research, the image is categorized into the particles and the background. This image splitting is done by choosing a certain intensity level on the histogram which separates in the best possible way the two probability density distribution curves. The optimal level for separation corresponds to the level which has equal probability of belonging to the background or the particles. This intensity level is called the threshold value (I_θ) of the image. An I_θ is calculated for every image after the means and the standard deviations of the two distributions are found. Recall that the two probabilities are approximated as being Gaussian using:

$$P_b = a_b \times \exp\left[-\frac{(I-\mu_b')^2}{\sigma_b'^2}\right]$$

$$P_p = a_p \times \exp\left[-\frac{(I-\mu_p')^2}{\sigma_p'^2}\right]$$

The threshold value is then chosen as the intensity level for which $P_b = P_p$ in the region $\mu_b' \le I \le \mu_p'$. When the image is determined as having low noise and it is not smoothed, the threshold value is more simply defined as:

$$I_\theta = \mu_b' + \frac{(\mu_p' - \mu_b')}{2}$$

since the two distributions are clearly distinct. Figures 4a, 4b, 5a, 5b, 6a, 6b all show the threshold value picked and its location relatively to the two distributions. The more the two distributions are overlapping, the harder it is to pick a threshold value which distinguishes the particles from the background completely. The procedure just described is the result of an extensive investigation aimed at defining "reasonable" thresholds under all circumstances

without any manual or subjective decisions. The procedure works for all cases where a decision could be made "by eye" and is probably more reliable in the case of mildly overlapping distributions. However, it fails (as it should) when the two distributions overlap completely.

After the threshold value is picked, the original image is converted into a binary image with 0 being the background and 1 being the particles. The thresholding is done as follows:

$$S(x,y)=0 \quad \text{for} \quad I(x,y) \leq I_\theta$$
$$S(x,y)=1 \quad \text{for} \quad I(x,y) > I_\theta$$

Figures 1c, 2c and 3c show the binary images that resulted after thresholding the images in Figures 1b, 2b and 3b, respectively. The choice of the threshold in Figure 1c is very accurate, thus only existing particles show in the binary image. In Figure 2c a few spurious particles that do not exist start to appear, but the choice of the threshold is still acceptable considering that the two probability density distributions in Figures 5a and 5b are undistinguishable even by eye. In Figure 3c, a large number of spurious particles appear which is an indication that the choice of threshold was very poor. Looking at the corresponding histogram, (Figures 6a and 6b), the distributions are overlapping each other to an extent so large that the particles cannot be separated from the background; that would be a typical case where the algorithm fails and where better quality images are required.

2.3 IMAGE ANALYSIS

The goal of the image analysis section is to identify and locate the particles in the already processed image, match them in pairs between two successive images, calculate their velocities and finally calculate the statistics of the errors in their positions and velocities.

2.3.1 Particle Identification

The image whose particles are to be identified is a thresholded binary image with 1 representing the particles and 0 representing the background. The 256 X 256 array of numbers is scanned line by line starting at pixel (2,2). If a pixel belongs to a particle, it is given a particle number. Pixel (1,1) and all the pixels in the lines x = 1 and y = 1 are treated separately. A pixel located at S(x,y) is recognized as a new particle when the pixels located at S(x,y-1), S(x-1,y-1), S(x-1,y) and S(x-1,y+1) are all 0. The particle number is incremented according to the total number of particles found up to that point. If a pixel does not meet the condition for a new particle, it is compared with every one of S(x,y-1), S(x-1,y-1), S(x-1,y) and S(x-1,y+1) which are not 0. If there is a match, it means that both pixels belong to the same particle and S(x,y) assumes the particle number of its match.

Depending on the shape of the particle boundary and the line sweep, it is possible to identify two subsets of pixels belonging to one particle as being first disjoined. This results into having two sets of pixels in the same particle with two different particle numbers. If this happens, the particle number of the pixels with the larger particle number is changed into the smaller one. The next available particle number is also updated. Then the search continues to the S(x,y+1) pixel and the procedure is repeated. This search for particles is achieved in one simple sweep through the picture for efficiency.

After identifying all the particles, their location is calculated by calculating their centers according to:

$$X_p' = \frac{\sum_{k=1}^{k=N_{pp}} x(k) \times I(x(k), y(k))}{\sum_{k=1}^{k=N_{pp}} I(x(k), y(k))}$$

$$Y_p' = \frac{\sum_{k=1}^{k=N_{pp}} y(k) \times I(x(k), y(k))}{\sum_{k=1}^{k=N_{pp}} I(x(k), y(k))}$$

where N_{pp} is the total number of pixels belonging to a given particle. Figures 1d, 2d and 3d show the position of the centroid of the particles in the images in Figures 1b, 2b and 3b, respectively. In case of the noisy images of Figures 2b and 3b, a number of spurious particles have also been identified. If the total number of particles found ever exceeds a given limit (350 in our case), this is an indication that the thresholding was erroneous and the case is not examined.

choi

2.3.2 Particle Matching

At this point, the true coordinates of the particles X_p and Y_p can be compared to the coordinates of the particles X_p' and Y_p' as they were found using the particle identification algorithm. To assess the accuracy of the algorithm, those two pairs of arrays must be compared by matching corresponding pairs of coordinates and computing the errors between the real and found ones. Morever, due to particle overlap, two original particles may be detected as one, small particles may be lost in the background noise and/or spurious particles may be found due to image noise. Hence, it is important to assess not only the accuracy of the particle matches but also the percentage of particles not found and the percentage of spurious particles.

The matching process is carried out by the following search. The particles whose centers overlap "exactly" (within one pixel) are matched first and removed from the list. Then a search area equal to the radius of every individual found particle is formed and the search for its match is carried out in that area in the generated image. This is done for the remainder not matched found particles. If there still are particles not matched after the second search, a third search starts trying to match the remaining generated particles with found ones. The search is carried out in the found image in an area with radius twice the radius of every individual remaining generated particle. Usually, the majority of the particles are matched in the first two processes and about 99 % after three passes. The number of particles from the original image for which a match has never been found is recorded along with the number of spurious particles which appear in the processed image.

The statistics of the errors in the particle locations are computed using the matching procedure described above. The mean $\mu_{location}$ and standard deviation $\sigma_{location}$ of the error is computed as follows:

$$\mu_{location} = \frac{\sum_{i=1}^{i=N_{pf}} \sqrt{(X_p(i) - X_p'(i))^2 + (Y_p(i) - Y_p'(i))^2}}{N_{pf}}$$

$$\sigma^2_{location} = \frac{\sum_{i=1}^{i=N_{pf}} \left\{ (X_p(i) - X_p'(i))^2 + (Y_p(i) - Y_p'(i))^2 \right\}}{N_{pf}} - \mu^2_{location}$$

where N_{pf} is the number of particles matched.

2.3.3 Calculation of the velocity field

To develop and test algorithms for the determination of the velocity fields, successive pairs of images were generated as follows. A first frame (frame A) was generated according to the procedure defined in section 2.1. Then a velocity field was specified analytically and a framing rate was chosen. Each particle is then displayed with its local velocity and a second frame (frame B) was generated according to the calculated new particle positions. Additive noise is then added on frame B as in section 2.1. A number of velocity fields with increasing complexity were specified to test the robustness of the algorithms.

The determination of the velocity field involves matching pairs of corresponding particles between the two successive images separated by a small instant of time Δt. To avoid ambiguities in the matching process, it is desirable to know the expected direction and magnitude of the displacement of each particle. However, it is imperative to respect the goals of this study and avoid any priori knowledge of the velocity field or rely on subjective guidance

from the operator. Since particles are essentially featureless and cannot be reliable recognized individually, the matching process is very difficult. However, under the assumption of a smoothly varying velocity field, small clusters of randomly spaced particles are recognizable for short periods of time since all particles have similar (but not equal) velocities. In other words, the motion of small clusters of particles which are identifyable due to their spatial pattern can be thought as a translation with the average velocity of the cluster plus a distortion of the pattern due to the non uniformity of the velocity field. The determination of this average velocity can be achieved by cross-correlating two corresponding subregions of the image. The size of the subregions is contained on both sides. If the region is too small, there may be no particles or very few particles to form a recognizable spatial pattern leading to an unsuccessful cross-correlation. If the region is too large, the found velocity is an average over a region of possible large velocity variation, the cross-correlation is poor and the velocity unrepresentative of each individual particle velocity. For a given size of the subset, the found velocity is assigned as being the velocity of the midpoint of every subset.

Depending on the complexity of the flow field, the average velocity which is calculated using the cross-correlation might not be an accurate estimate for each of the particles in the subset of the image. For a flow field where the velocity is uniform everywhere, the cross correlation technique works very good in estimating the velocity because all the particles in every subset move with the same velocity. If the flow field is more complex with large gradients in velocity, the cross correlation technique returns only an average velocity for every subset which does not necessarily represent the velocity of each and every particle in the subset. To solve this problem, a better estimation of the velocity of every individual particle is calculated by interpolating between the average velocities of the adjacent the image subsets. The interpolation is based on the particle coordinates and the coordinates of adjacent image subset midpoints. Hence, the choice of size of the image subsets is a compromise between the output accuracy, the limitations imposed by the cross correlation technique and the computer time involved.

For this research both A and B images are divided into 16 subsets each with dimensions 64 X 64. Each one of these subsets is reduced into a 32 X 32 array by averaging every four pixels. Reducing the subset size, decreases the computer time involved in

processing every subset and does not significantly change the velocity estimate since particles are more than 1 pixel in diameter. Next, the cross correlation between each 32 X 32 arrays of image A and B is performed and the average velocity for all of the 16 subsets is calculated. The calculated velocity is assigned as the velocity of point (16,16) in each subset. The cross-correlation is calculated in the usual manner by calculating the two dimensional Fourier transform of subset A and B by FFT technique, cross multiplying the results and taking the inverse Fourier transform (Horn, 1986). Then, depending on the location of every particle with respect to the 16 known velocity points, a local velocity estimate is found by bilinear interpolation. When a particle leaves close to the boundary of the image moves out of the image in the second time frame, this particle is not considered for any of the subsequent calculations. Figure 7 shows two images divided into 16 subsets, a sample subset is chosen from both images with a representative cluster of particles. The particle cluster in the subset from the second image is displayed in time with respect to the cluster from the first subset. Also shown are the 16 average velocity vectors as they resulted by cross-correlating each pair of the 16 subsets. The velocity vectors are located in the midpoint of every subset. The direction of the arrows and their length represent the direction and magnitude of the velocity at every point. Then, the average estimated velocity field is shown as it resulted from interpolation. The interpolation is done between the four adjacent to every particle location average velocities of the subsets.

The estimated velocity field is then used to create an intermediate image representing the found particles at estimated locations. Frame A of Figure 8 shows an example of some found particles at their original locations (in solid line) and the corresponding particles which were created using the estimated velocity field at the estimated locations (in dotted line). Frame B of Figure 8 shows the same created particles next to the originally displaced particles. A process to match the particles in Frame B is then initiated. If the estimated velocities were exact, the two images should overlap perfectly. Usually they do not exactly correspond, so the following particle matching scheme was devised. For every particle, a circular search area is defined starting with a radius equal to one tenth of the particle radius. If a match for the particle is not found, the radius of the search area is incremented by one tenth of the particle radius and the match procedure is repeated. The matched particles are

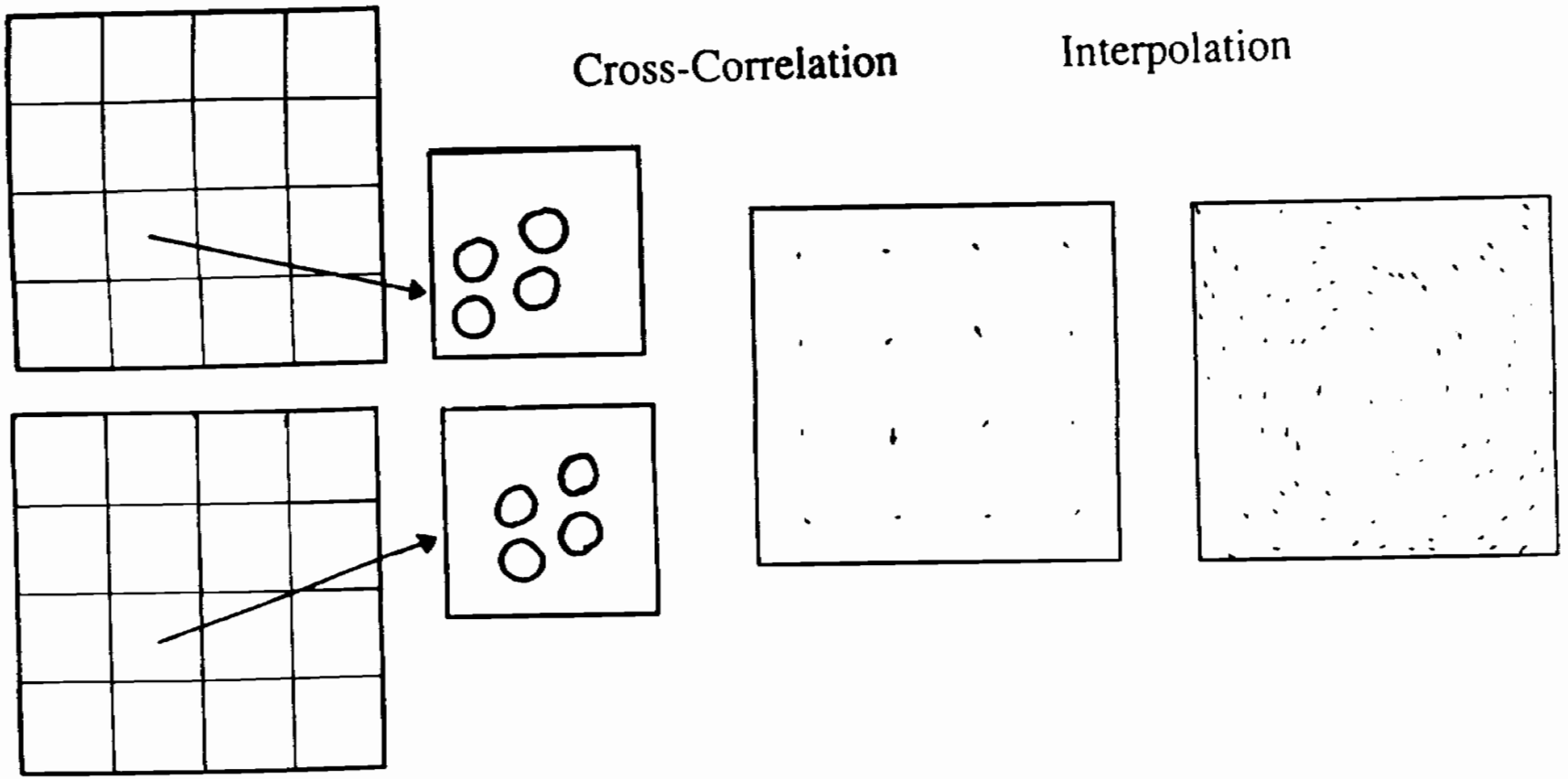


Figure 7. The Velocity Determination Process (Step 1): Velocity Approximation by Zonal Cross-Correlation and Interpolation

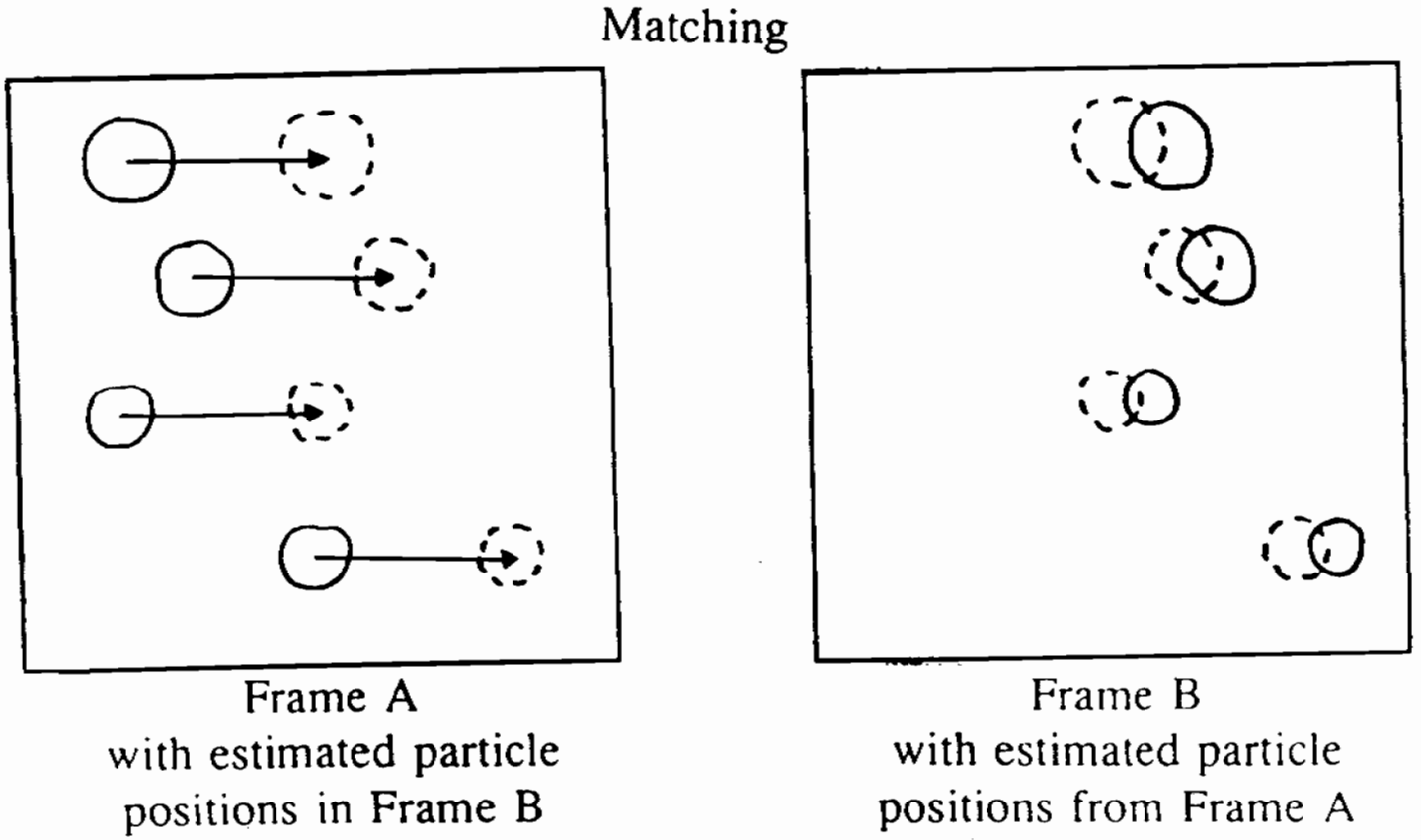


Figure 8. The Velocity Determination Process (Step 2): Matching Particles between Frames based on Velocity Approximation

removed from the subsequent passes of the search to limit erroneous matches. It was found from preliminary tests that the search should be terminated when the maximum search area has a radius equal to 15 times the radius of the particle and/or when 98 % of the found particles have been matched. The statistics of the error in determining the velocity are then computed. The magnitude of the true velocity (V) is compared to the magnitude of the measured velocity (V'). The mean ($\mu_{velocity}$) and the standard deviation ($\sigma_{velocity}$) of the error in the velocity of the particles are then computed according to:

$$\mu_{velocity} = \frac{\sum_{i=1}^{i=N_{pf}} \sqrt{(V(i) - V(i)')^2}}{N_{pf}}$$

$$\sigma^2_{velocity} = \left\{ \sum_{i=1}^{i=N_{pf}} \frac{(V(i) - V(i)')^2}{N_{pf}} \right\} - \mu^2_{velocity}$$

Examples to illustrate the overall procedure are Figures 9 and 10. Figures 9a and 10a show the "true" velocity fields of a vortex and of a stagnation point flow both located at (128,128). The particles which overlap are represented with one arrow in the found velocity field. Figures 9b and 10b show the average velocities calculated using the cross correlation technique, assigned at the midpoint of every image subset. Using these velocities and interpolating for every found particle, the estimated velocity fields are shown in Figures 7c and 10c. Figures 9d and 10d show the reconstructed velocity field which is found by applying the entire process described above. A comparison between the found and the true velocity field shows the effectiveness of the process.

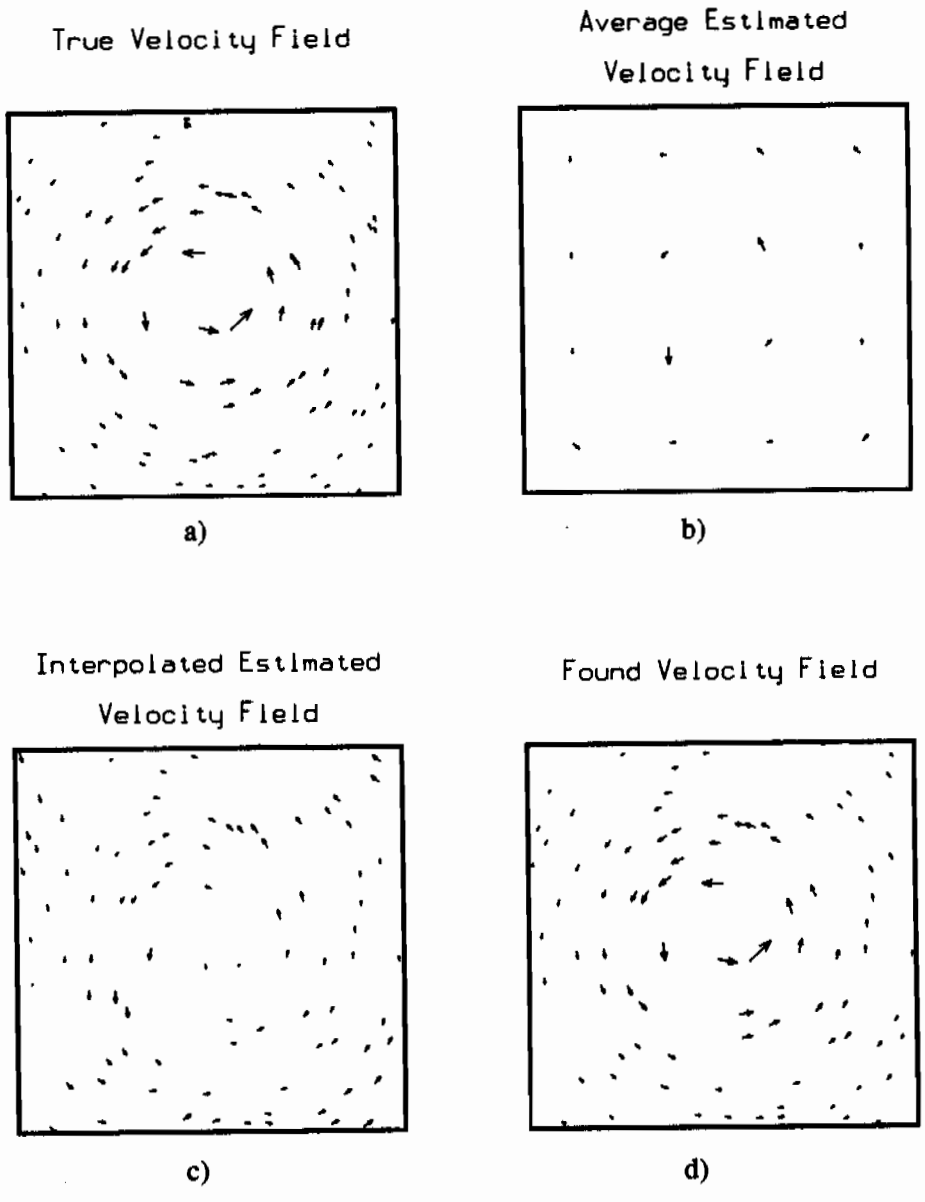


Figure 9. Velocity field reconstruction for vortex flow with N_p = 100; a) True velocity, b) Estimated velocity, c) Interpolated estimated velocity and d) Reconstructed velocity

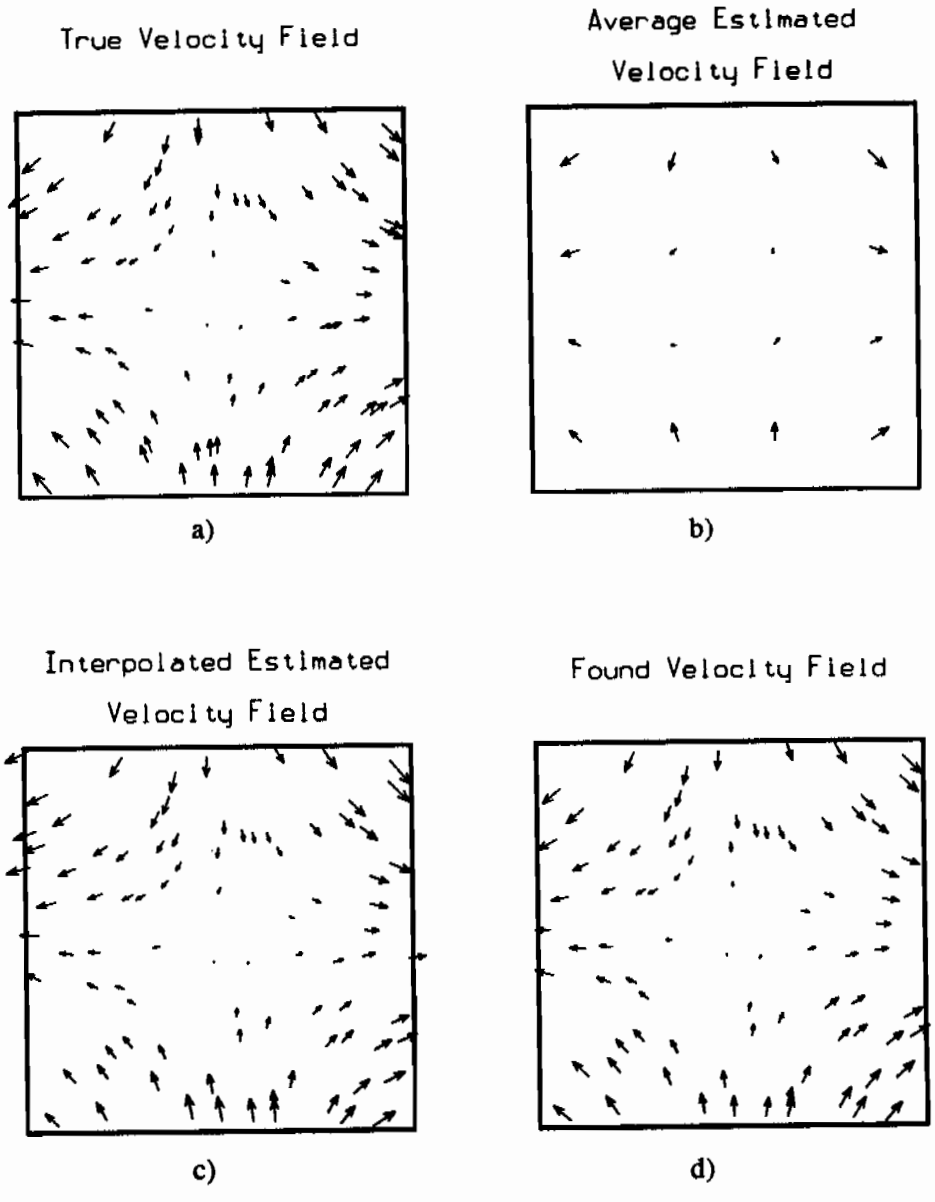


Figure 10. Velocity field reconstruction for stagnation point flow with N_p = 100; a) True velocity, b) Estimated velocity, c) Interpolated estimated velocity and d) Reconstructed velocity

CHAPTER III

RESULTS AND DISCUSSION

In order to investigate the accuracy of the Particle Image Velocimetry technique two computer programs were developed implementing the algorithms described in Chapter II. Both programs are included in the appendices.

The first program (Program Location, Appendix A) performs the generation of synthetic images according to a specified set of parameters, analyses the images to find the particle locations and calculates the average error (mean) and standard deviation of the error between the true locations and the found location of the particles. For each set of parameters, 50 images were generated and processed to obtain reliable statistical averages of all computed quantities. This number of 50 images was chosen based on preliminary tests of statistical convergence. A total of 122 different parameter sets were investigated resulting in the processing of a total of 6100 images. The parameters varied were the number of particles, their average size, the contrast between the particles and the background and the noise level in the image. The programs were implemented on a Masscomp 55S05 workstation. Without any special processing hardware, the average processing time was 45 seconds per image from image generation to the calculation of output statistics.

The second program (Program Velocity, Appendix B) generates synthetic images and the corresponding images of the same particles after a small instant of time in a prescribed velocity field, locates the particles in both images, estimates the particle velocities and evaluates the mean and standard deviation of the errors between the real velocities and the computed ones. A number of different velocity fields were tested (shear flow, vortex, stagnation point flow, etc...) but the systematic evaluation of the error was performed on a Rankine vortex flow (potential vortex with viscous, solid body rotation core). The parameters investigated were the size of the vortex core (spatial scale), maximum particle displacement between images in combination with the other parameters listed above governing the images themselves. Again, 50 pairs of images were generated for every set of parameters, representing a

total of 1800 pairs of images. The average processing time was 230 seconds for each pair of images.

Both programs are completely automatic and do not require the input of an operator. The results were written into an output file for post processing, graphing and analysis.

3.1 POSITION ACCURACY RESULTS

The number of particles, contrast, noise level and radii of the particles are parameters which affect the accuracy with which particles can be located. The next sections describe the combined effect of these various parameters.

3.1.1 Effect of image contrast and noise

The contrast (C) is the difference between the mean value of the particle levels and the mean value of the background level. Generally, for a given noise level (A), the higher the contrast the more distinct the particles are from the background and the lower the contrast the more difficult it is to distinguish them. The values of contrast examined were 200, 150, 100, 50, 25 and for every one of these contrasts the noise level is varied through 0, 10, 20, 30 and 40. This test was repeated for 50 and 100 particles in the image, while keeping the mean radius of the particles constant at 3 ± 1. Figures 11a and 11b show the effect of the contrast for different noise levels on the mean and standard deviation of the error of the particle locations respectively for $N_p = 50$. Each curve represents a different contrast. Figures 12a and 12b summarize the same information for $N_p = 100$. Those errors are calculated for the particles for which there is a match between a actual particle and a detected particle as described in section 2.3.2. In Figure 11a, $\mu_{location}$ remains approximately the same, and very close to zero for C = 200, C = 150 and C =100 for all the noise levels but it increases significantly

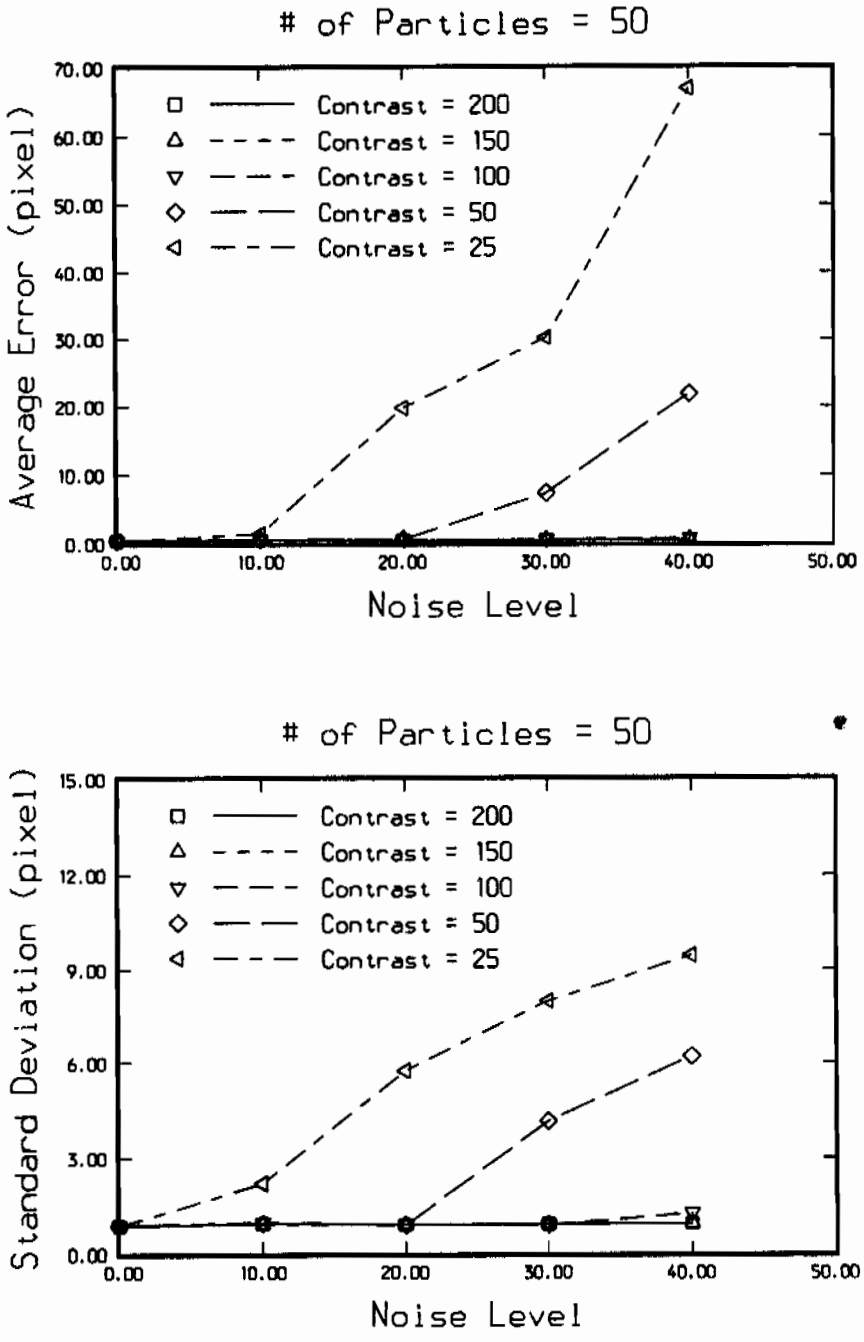


Figure 11. Dependence of error in locating particles on noise level and contrast for N_p = 50; a) Average error and b) Standard deviation of error

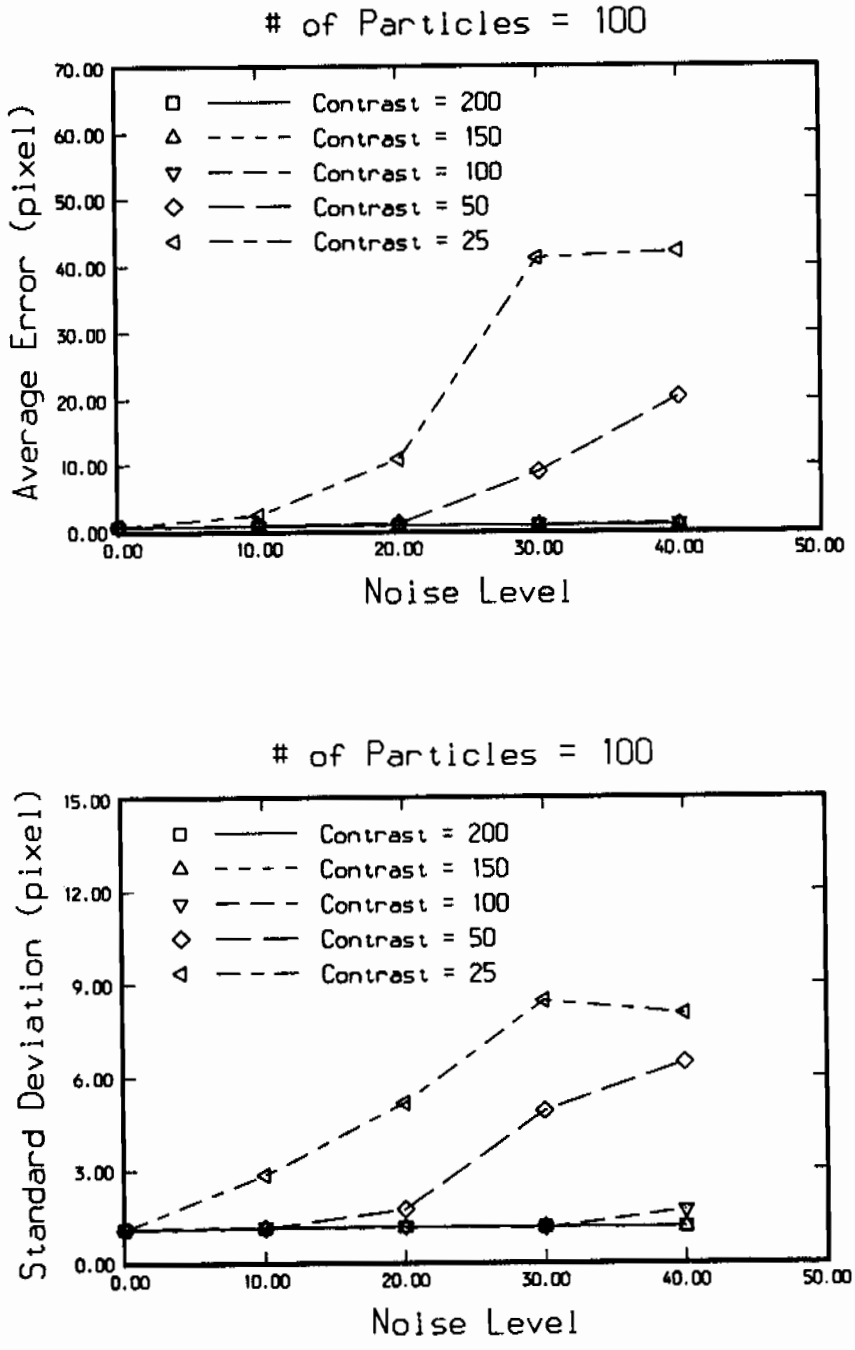


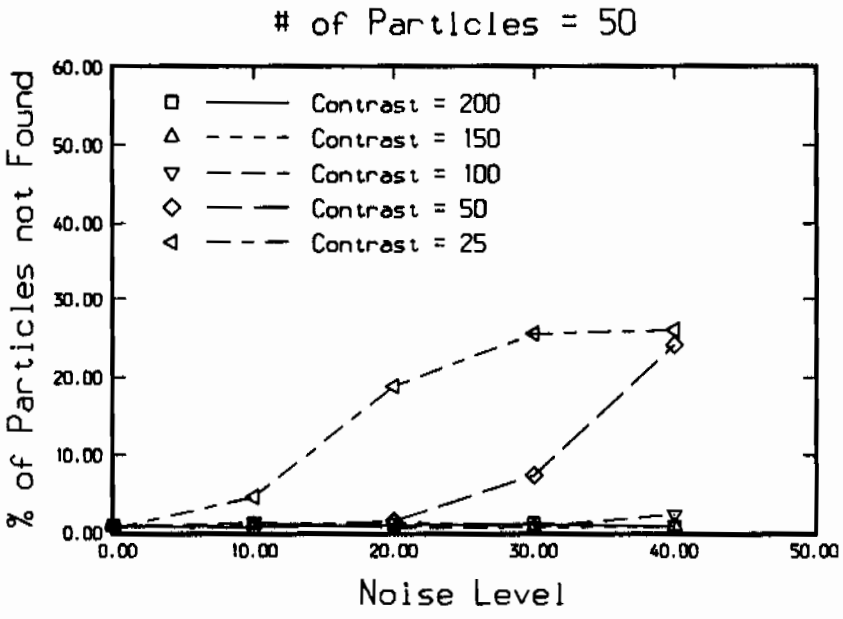
Figure 12. Dependence of error in locating particles on noise level and contrast for N_P = 100; a) Average error and b) Standard deviation of error

after A = 20 for C = 50 and after A = 10 for C = 25. A graph in a smaller scale (not included here) shows that $\mu_{location}$ for the three higher contrasts and for all levels stays constant at about 0.5, i.e, providing sub-pixel accuracy. The graph of $\sigma_{location}$ in Figure 11b behaves the same way as the graph at Figure 11a. $\sigma_{location}$ for C =50 and C =25 starts to increase drastically after A = 20 and A = 10, respectively. A graph on a smaller scale (also not included here) shows that on the average $\sigma_{location}$ = 0.9 for C = 200, C = 150 and C = 100. Increasing the number of particles to N_p = 100 (Figures 12a and 12b) increases $\mu_{location}$ to an average of about 0.7 and $\sigma_{location}$ to an average of about 1.2 for the three higher contrasts.

In addition, the percentage of particles not found (i.e., real particles not detected) and the percentage of "spurious" particles (i.e., particles detected not corresponding to real particles) were calculated. Figures 13a and 13b summarize those percentages as a function of contrast and noise for N_p = 50. In Figure 13a, the number of undetected particles in the original image (N_{pm}) is about 1 % of the percentage of spurious particles found ($N_{pm}^{'}$) is very close to 0 % for the three higher contrasts and for all noise levels. Decreasing the contrast further increases the percent of undetected and spurious particles in both images. The maximum N_{pm} reaches 30 % but $N_{pm}^{'}$ goes well beyond that to about 425 %. Similar results for N_p = 100 are shown in Figures 14a and 14b. The behavior of the curves is the same as in Figures 13a and 13b.

3.1.2 Effect of particle size and number of particles

To investigate the effect of the average particle size (R) and number of particles (N_p) on the accuracy of the results, a combination of two contrast levels (C = 50 and C = 150) and two image noise levels (A = 10 and A = 30) were further investigated. Based on the results of section 3.2.1, these contrasts and noise levels were deemed representative of the various image qualities likely to be encountered (combination of low or high contrast with low or high noise). The mean particle radii were chosen to be 2, 3.5 and 5 pixels while



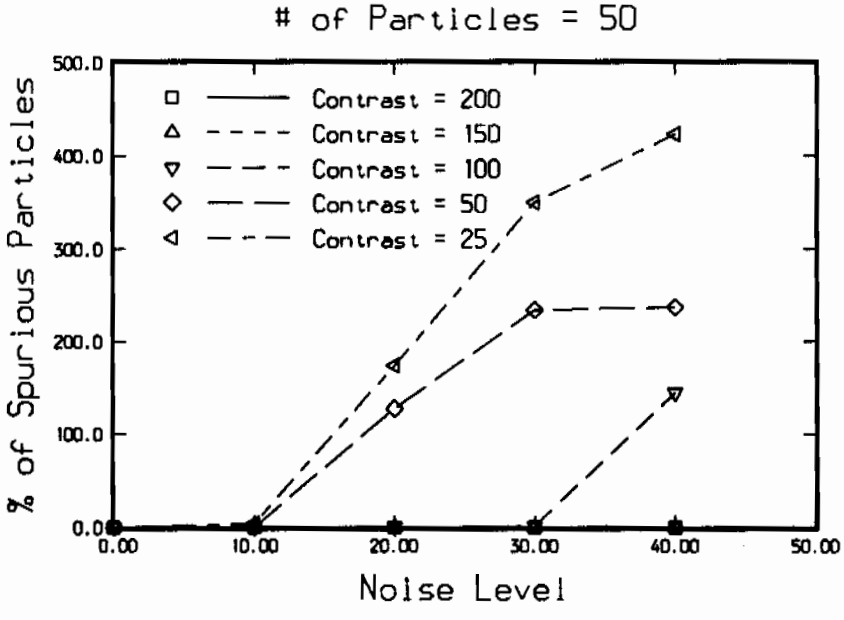


Figure 13. Dependence of percentage of a) particles not found and b) spurious particles on noise level and contrast for $N_p = 50$

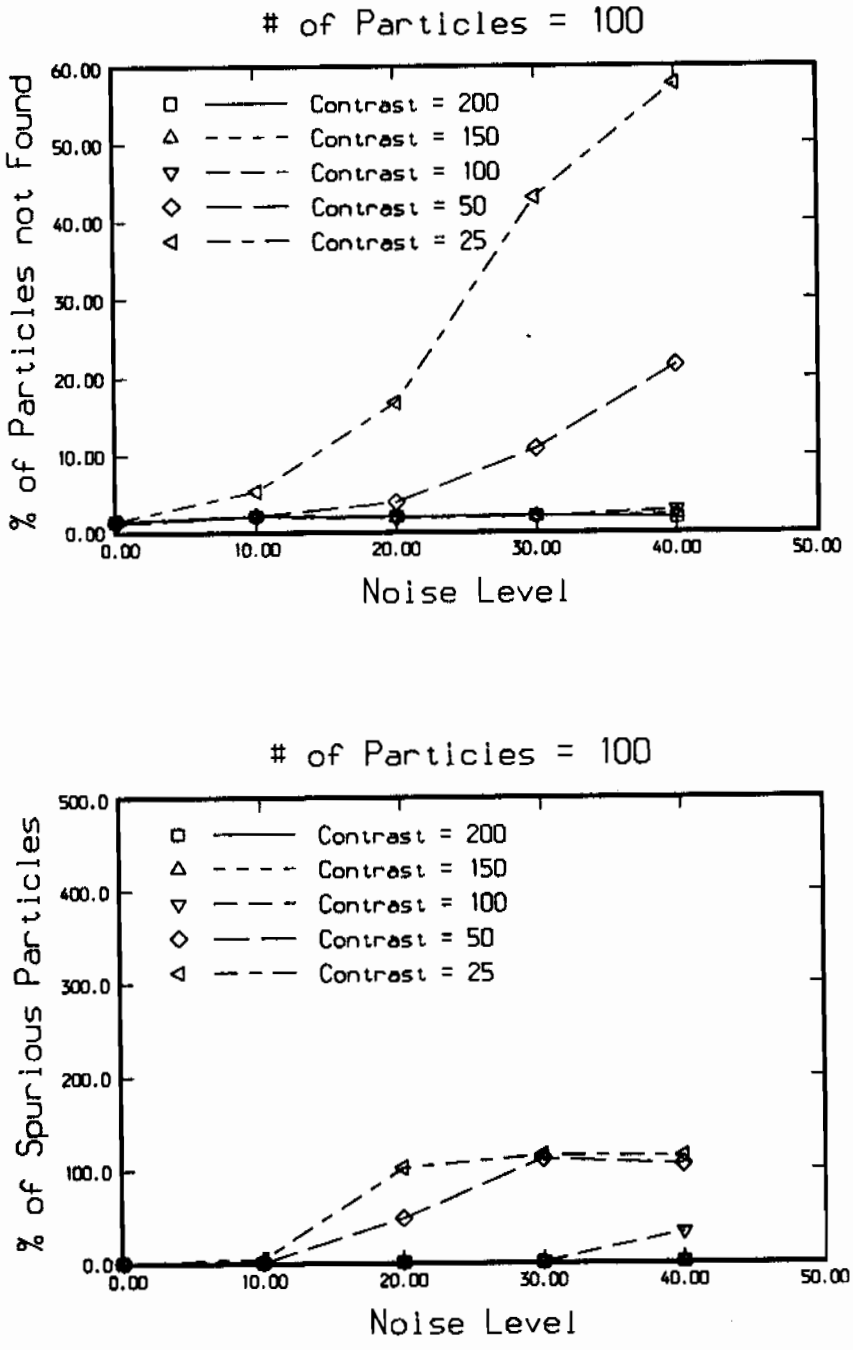


Figure 14. Dependence of percentage of a) particles not found and b) spurious particles on noise level and contrast for N_p = 100

keeping the distribution of the radii the same (A_r = 1.0). Figures 15a and 15b depict the mean and standard deviations of the error in particle locations as a function of the number of particles for various particle sizes. These Figures correspond to a contrast of 150 and an image noise of 10 (i.e., a high quality image). Similar information is summarized for high contrast (C = 150), high noise (A = 30) images in Figures 16a and 16b, low contrast (C = 50), low noise (A = 10) images in Figures 17a and 17b and low contrast (C = 50), high noise (A = 30) images in Figures 18a and 18b. The percentages of undetected particles and spurious particles for these same cases are depicted in Figures 19 through 22, respectively.

Figures 15a and 15b show that as the number of particles increases, the average error and the standard deviation increases. Also, as R increases, $\mu_{location}$ and $\sigma_{location}$ also increase for all the N_p 's examined. Figure 19a shows that the number of particles not found N_{pm} is kept less than 2.7 % for all the N_p 's and the two smaller particle sizes but it increases significantly for the larger particle size (5 ± 1) and for $N_p > 75$. In Figure 19b, the number of spurious particles $N_{pm}^{'}$ stays less than 0.2% for all particle sizes and all N_p 's greater than 50.

For the case where C =150 but the noise level increases to 30 (Figures 16a and 16b), the basic behavior of the magnitude of the curves for $\mu_{location}$ and $\sigma_{location}$ remain the same as in Figures 15a and 15b. A difference appears for 2 ± 1 and $25 < N_p < 50$, where both $\mu_{location}$ and $\sigma_{location}$ assume relatively high values for N_p = 25 ($\sigma_{location}$ assumes the maximum value over all the particle sizes) and then start to decrease until N_p = 50. This behavior is anomalous and probably limited to processing error. After that, both curves follow a monotonically increasing behavior. In Figure 20a N_{pm} is kept below 5 % for all the particles sizes but is smaller for R = 3.5 ± 1 and R = 2 ± 1. On the contrary in Figure 20b, $N_p^{'}$ is more than 450 % for R = 2 ± 1 and N_p = 25. After that it starts to decrease and it assumes low percentages only after N_p = 75. Although $N_p^{'}$ for 3.5 ± 1 is about 20 % for N_p = 25, it decreases after that and becomes very close to 0 % for $N_p > 50$.

The behavior of the curves in Figures 17a and 17b is the same as in Figures 15 and 16. In Figure 21a, the number of particles not found increases as the number of particles in the image increase following the trend of Figure 20a. Here, N_{pm} rises up to about 6 %. In

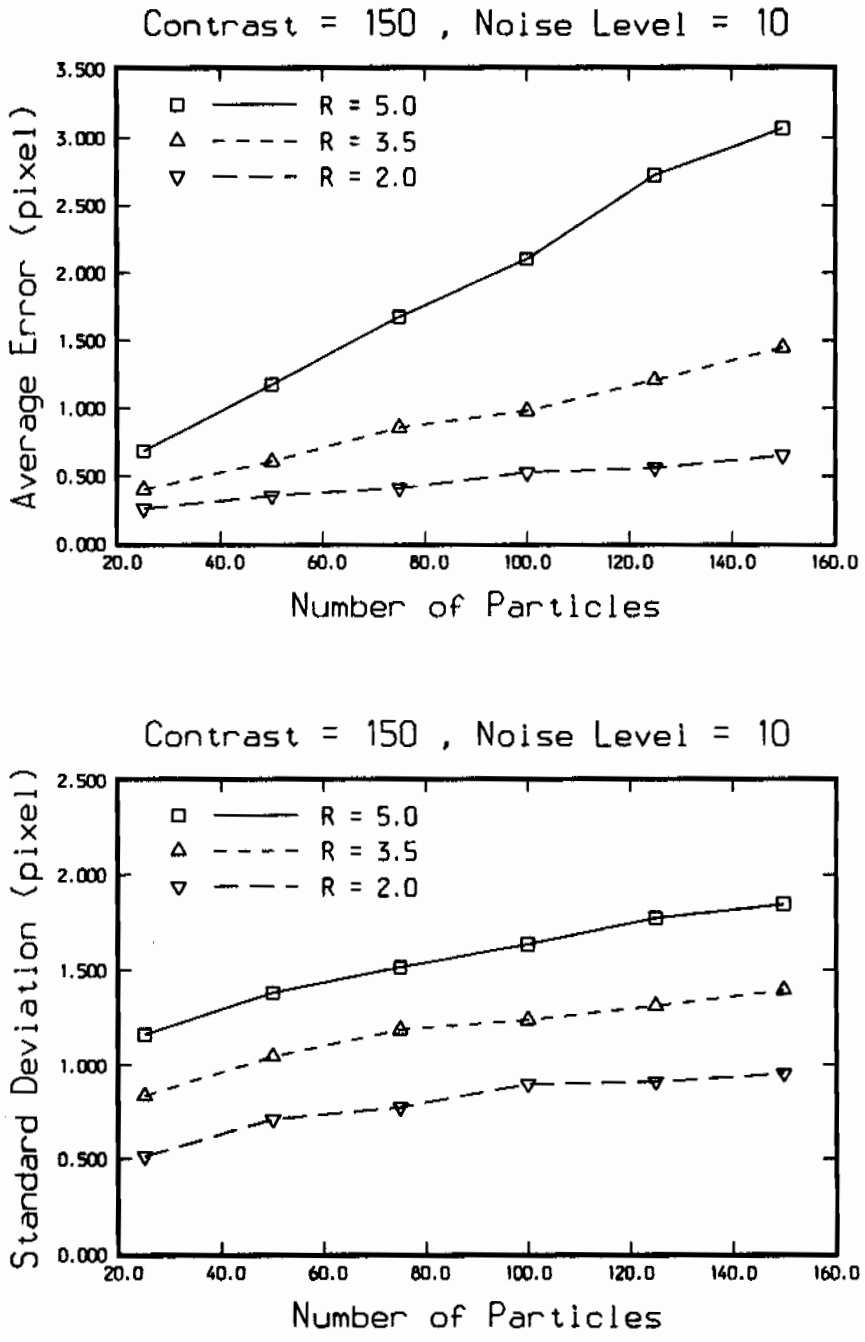
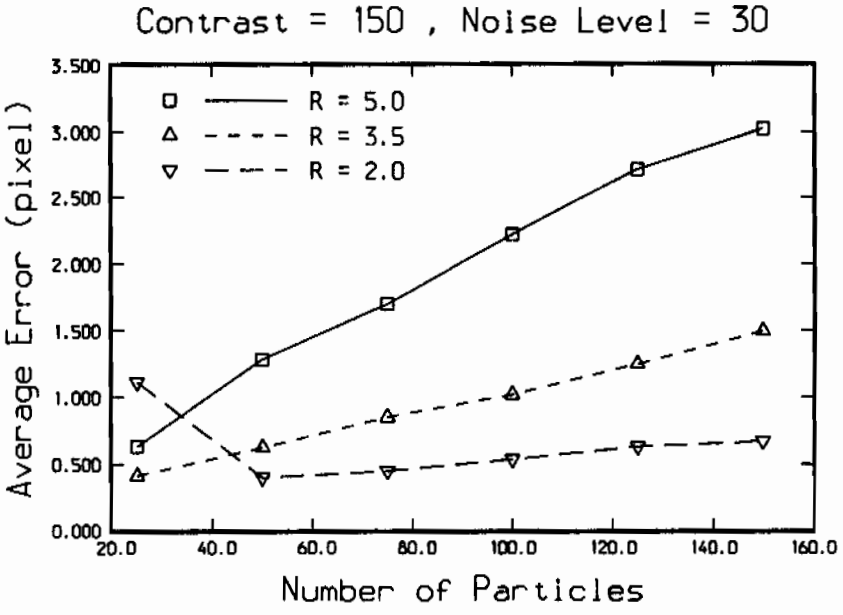


Figure 15. Dependence of error in locating particles on number of particles and particle size for C = 150 and A = 10; a) Average error and b) Standard deviation of error



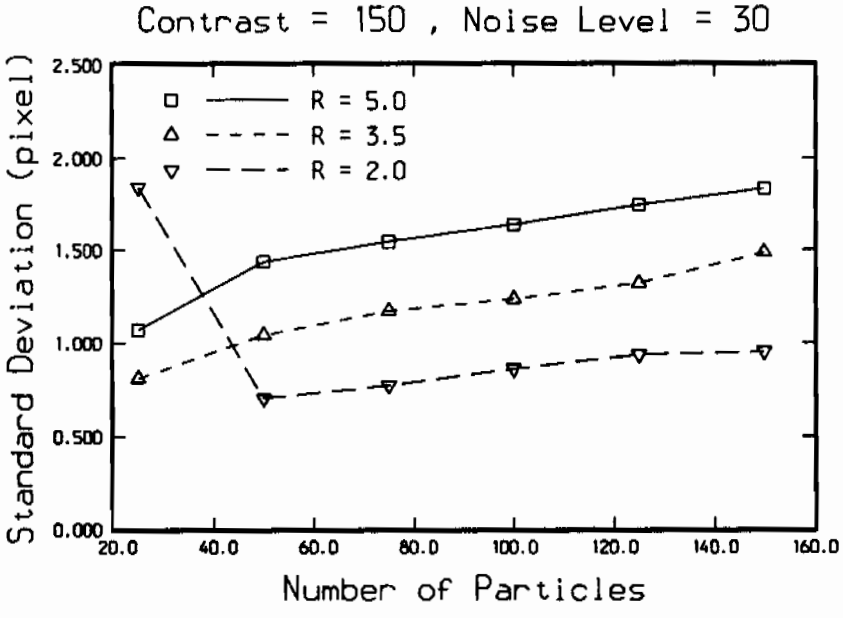
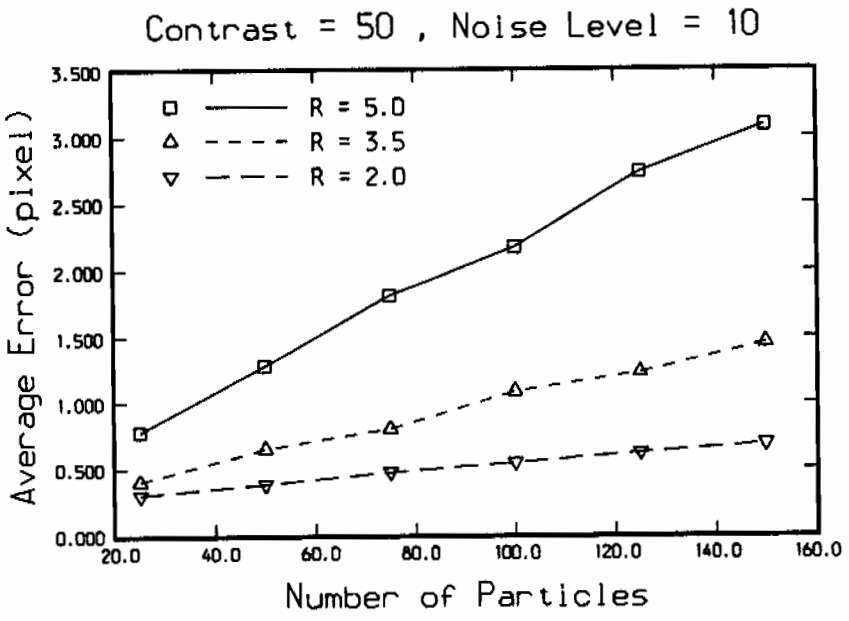


Figure 16. Dependence of error in locating particles on number of particles and particle size for C = 150 and A = 30; a) Average error and b) Standard deviation of error



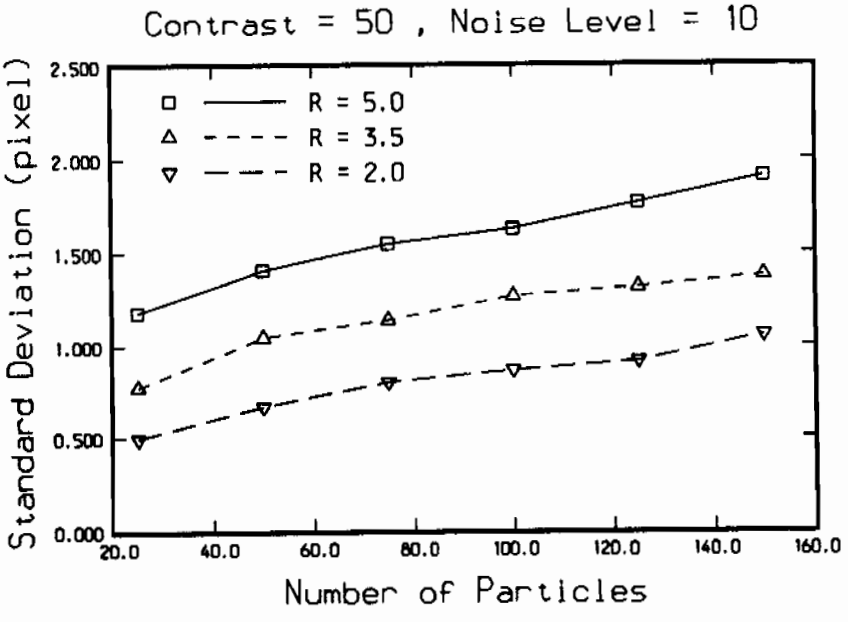


Figure 17. Dependence of error in locating particles on number of particles and particle size for C = 50 and A = 10; a) Average error and b) Standard deviation of error

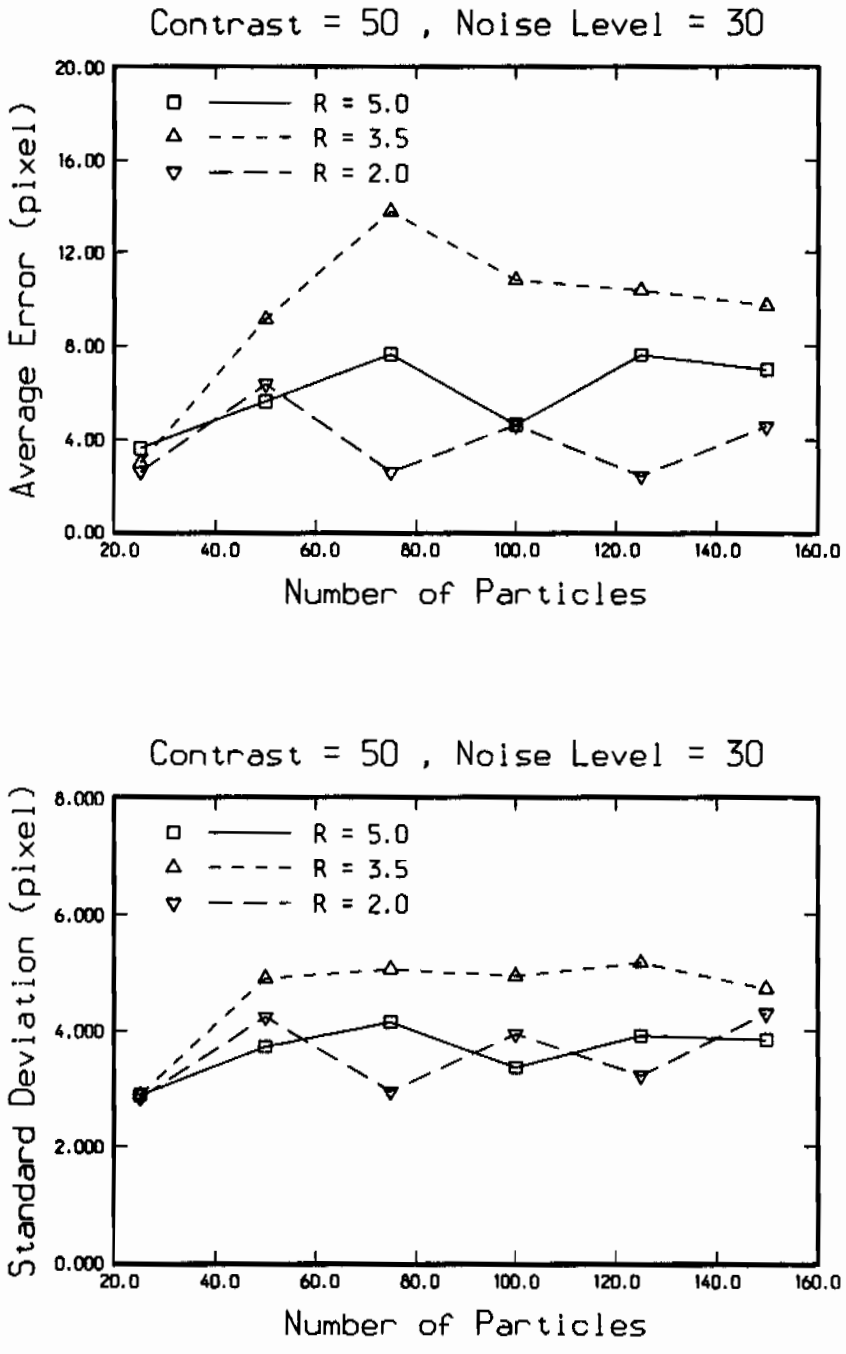


Figure 18. Dependence of error in locating particles on number of particles and particle size for C = 50 and A = 30; a) Average error and b) Standard deviation of error

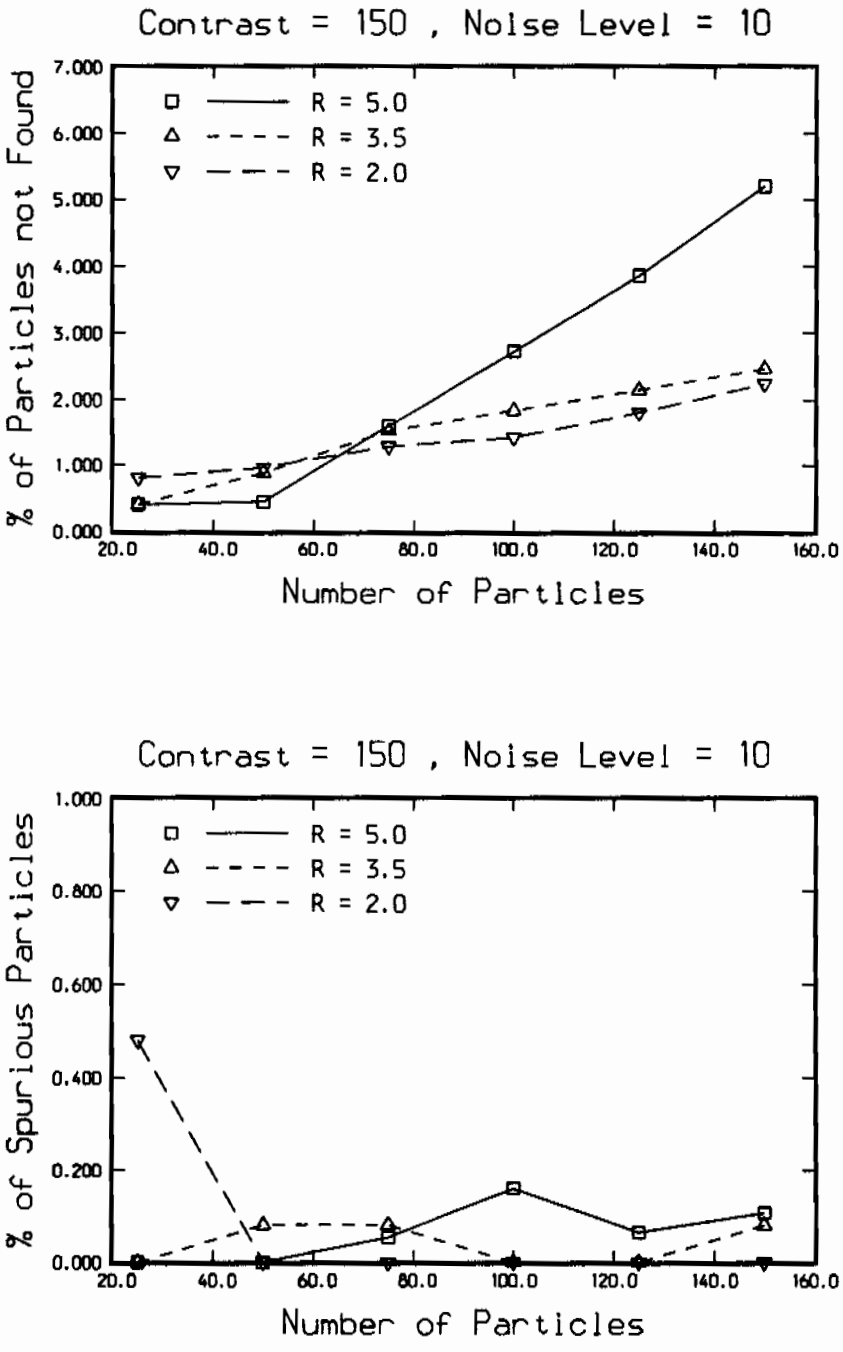


Figure 19. Dependence of percentage of a) particles not found and b) spurious particles on number of particles and particle size for C = 150 and A = 10

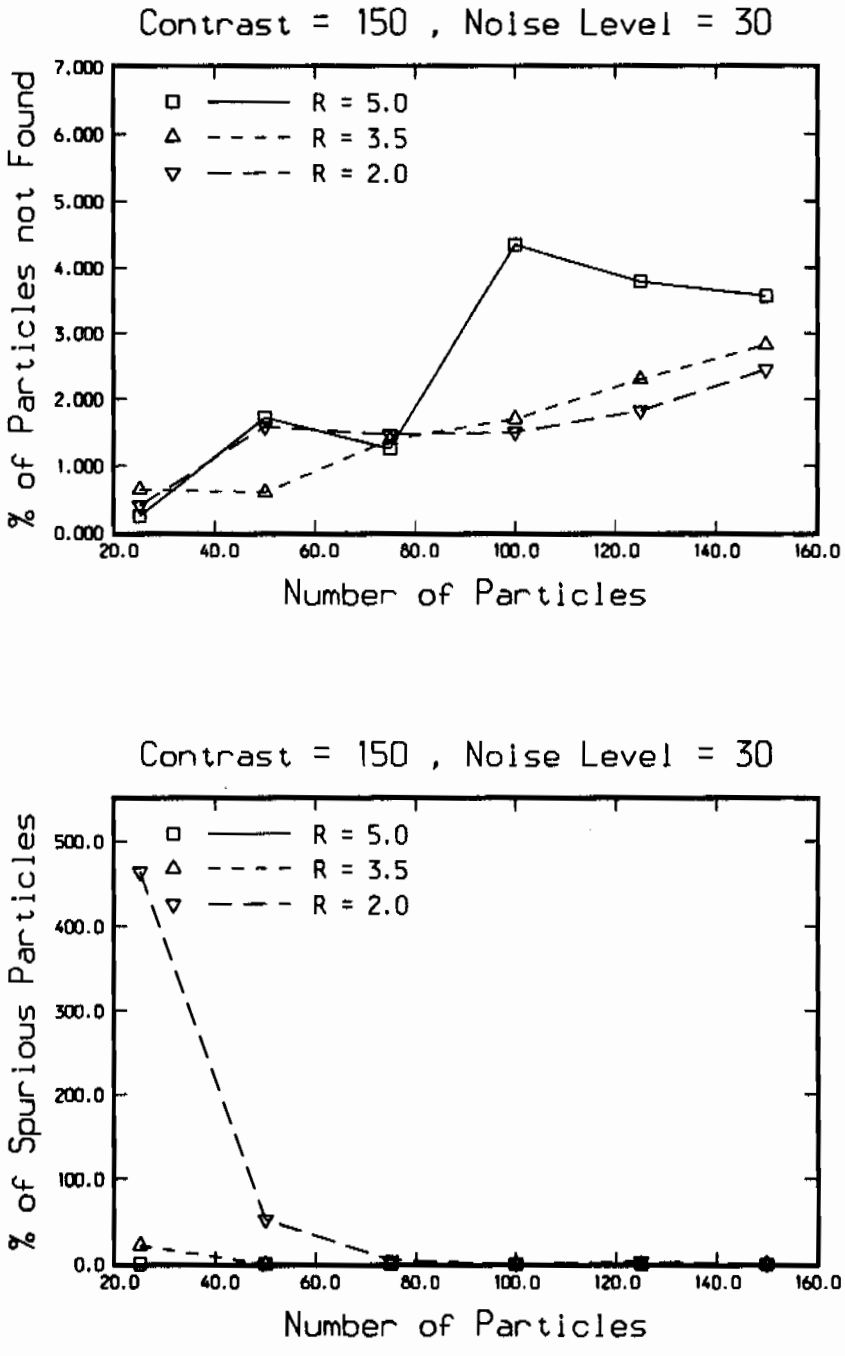


Figure 20. Dependence of percentage of a) particles not found and b) spurious particles on number of particles and particle size for C = 150 and A = 30

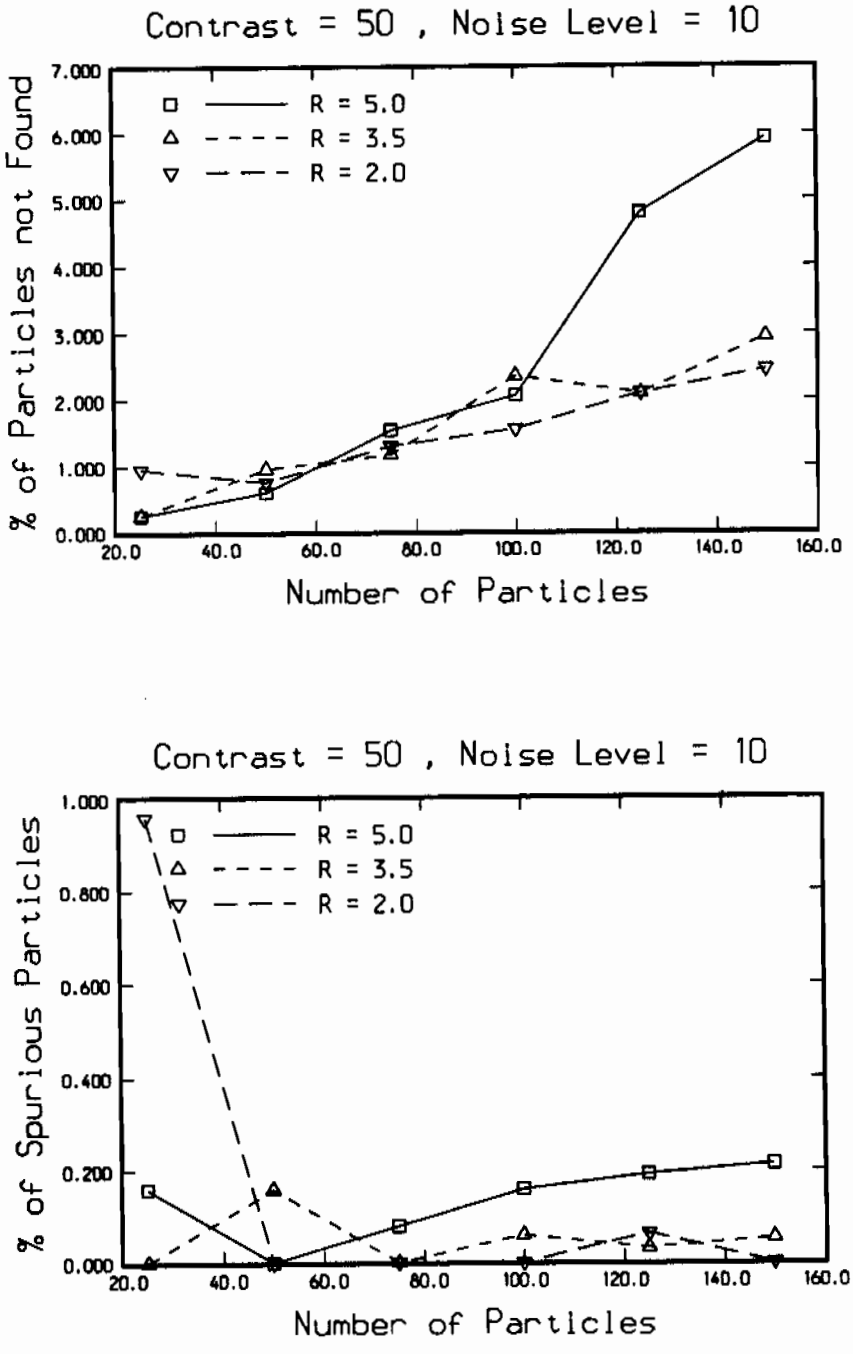
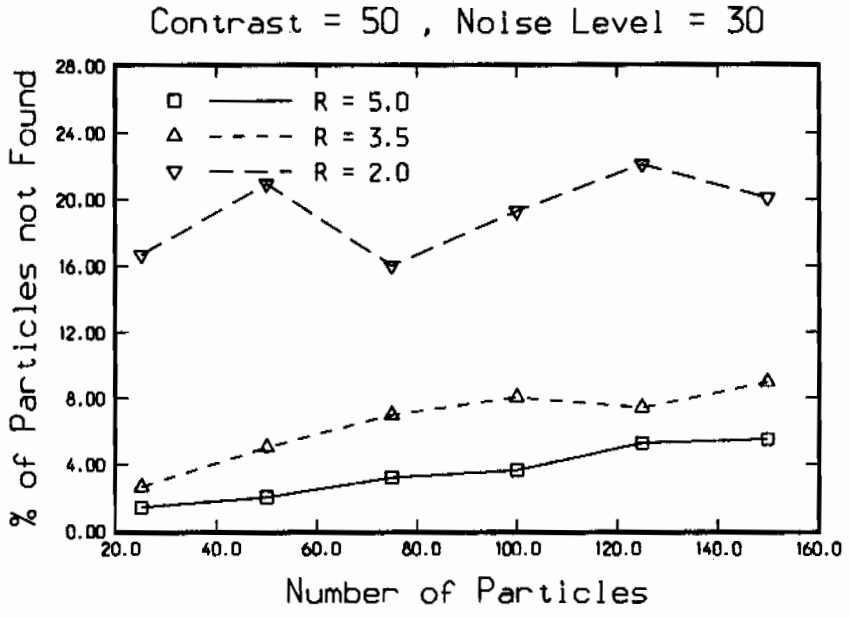


Figure 21. Dependence of percentage of a) particles not found and b) spurious particles on number of particles and particle size for C = 50 and A = 10



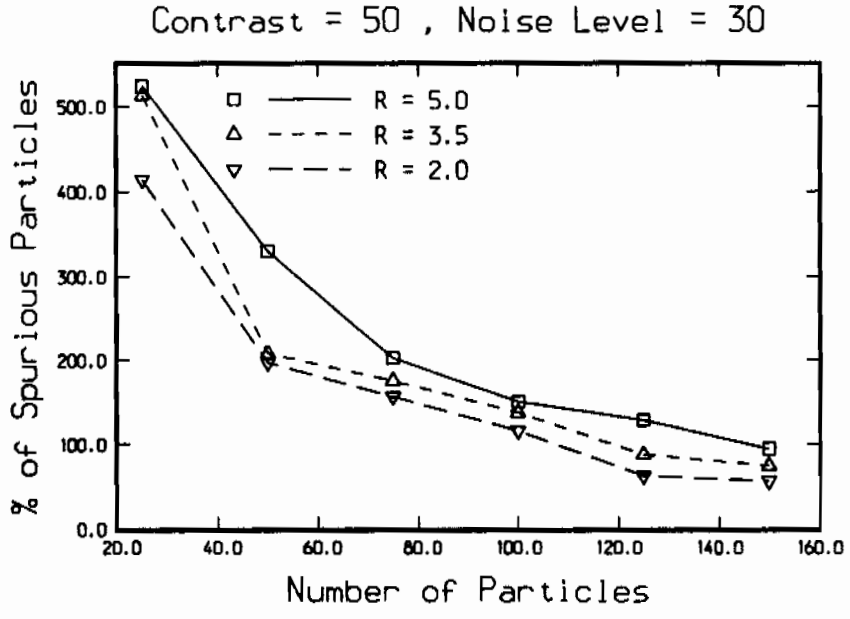


Figure 22. Dependence of percentage of a) particles not found and b) spurious particles on number of particles and particle size for C = 50 and A = 30

Figure 21b, N'_{pm} remains less than 0.2 % for all the particle sizes and all the particle numbers except for the case when the particle radius is 2 ± 1 and N_p = 25.

The graphs in Figure 18 show the worst case where both $\mu_{location}$ and $\sigma_{location}$ are very large for all particle sizes and particle numbers compared to all the previous cases. $\mu_{location}$ in Figure 18a never falls below 2.5 pixels and goes as high as 13 pixels for particle sizes of 3.5 ± 1 and N_p = 75. $\sigma_{location}$ in Figure 18b stays between 2.4 pixels and 5 pixels for all particle sizes and particle numbers. The percentage of particles not found (Figure 22a) is also very large compared to previous cases. In Figure 22b, N'_{pm} never falls below 80 % indicating a total failure of the processing regardless of image contrast due to poor image quality. On the contrary, N'_{pm} goes up to about 520% for the case where N_p = 25 and R = 5 ± 1.

3.2 VELOCITY ACCURACY RESULTS

The flow field of a Rankine vortex with variable core size and strength was chosen for this investigation because it is a realistic representation of a small portion of a turbulent flow field. The radius of the vortex core (R_v) is varied through 10, 25, and 40 pixels and its maximum strength velocity at the edge of the core is varied through 5, 10, 15, and 20. The number of particles investigated is 50, 100 and 150. A high contrast of 200 and low noise of 10 is set for every picture generated. All the particles are generated with a mean radius of 3.5 ± 1. Figures 23a, 24a, and 25a depict the variations of the average error of the particle velocity $\mu_{velocity}$ (as a percentage of the maximum velocity) as a function of the maximum particle displacement in pixels D_{max}. The maximum particle displacement represents the combined effect of the maximum velocity encountered in the field and the framing rate. Each graph shows three curves corresponding to the three vortex core sizes used. In all three Figures, $\mu_{velocity}$ increases as the vortex core size increases for every D_{max}. When the number of particles in the image is 50, $\mu_{velocity}$ is maximum for all vortex core sizes and all maximum particle displacements. As N_p increases, $\mu_{velocity}$ decreases. In Figure 23a $\mu_{velocity}$

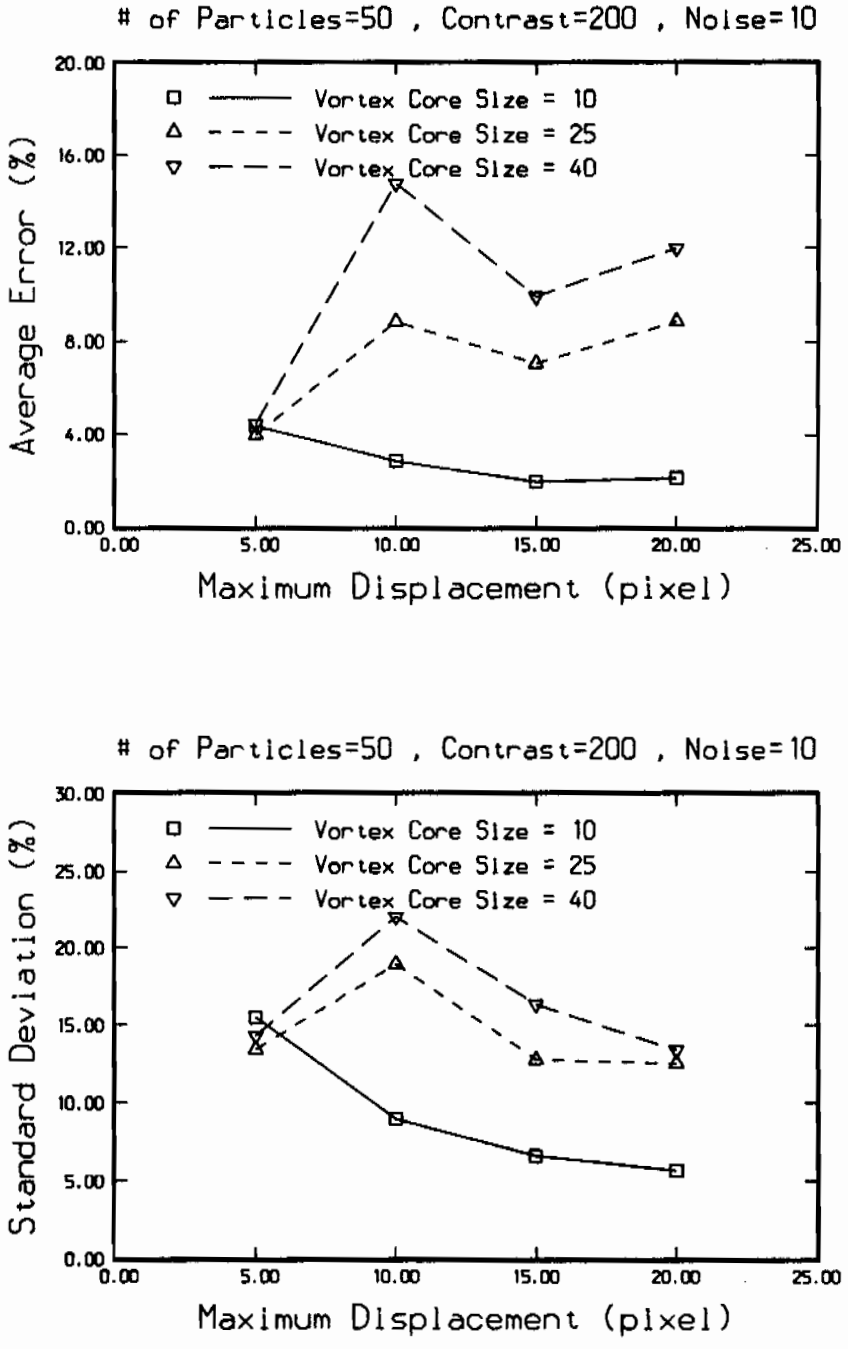
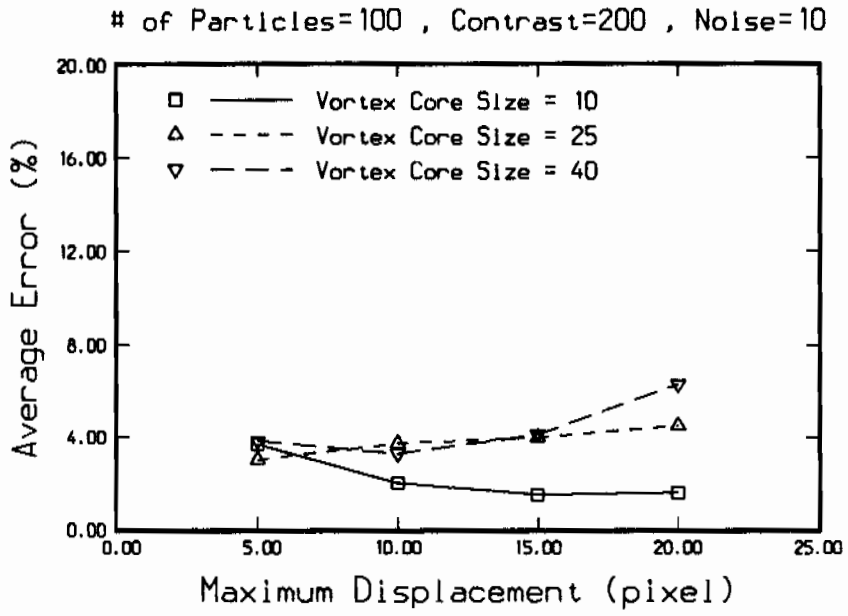


Figure 23. Dependence of error in particle velocity on maximum particle displacement and vortex core size for N_p = 50, C = 200 and A = 10; a) Average error and b) Standard deviation of error



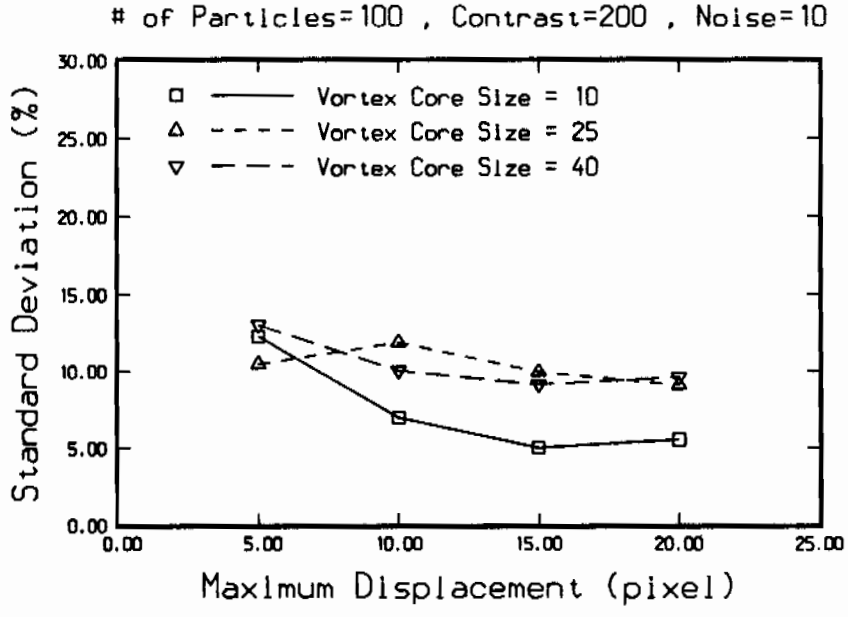


Figure 24. Dependence of error in particle velocity on maximum particle displacement and vortex core size for N_p = 100, C = 200 and A = 10; a) Average error and b) Standard deviation of error

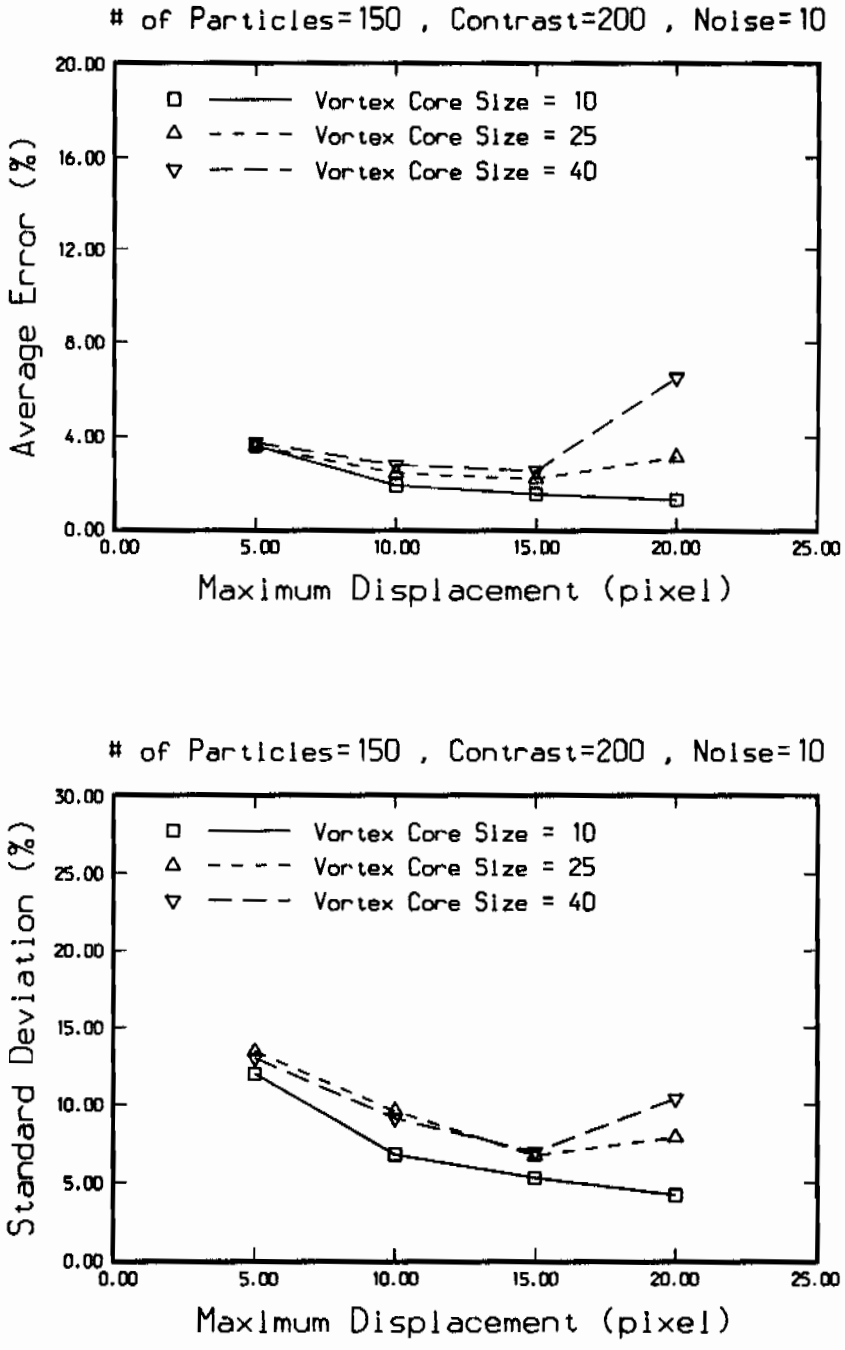


Figure 25. Dependence of error in particle velocity on maximum particle displacement and vortex core size for N_p = 150, C = 200 and A = 10; a) Average error and b) Standard deviation of error

increases and fluctuates between 8 % and 15.5 % for R_v of 25 and 40 but it decreases to about 3 % for R_v = 10 as D_{max} increases. In Figures 24a and 25a, $\mu_{velocity}$ stays approximately constant up to D_{max} = 15 as the maximum particle displacement. After that, it increases only for R_v = 25 and R_v = 40. The smallest $\mu_{velocity}$ of about 1.8 % is achieved with 150 particles in the picture, a vortex core size of 10 pixels and a maximum particle displacement of 20. Figures 23b, 24b and 25b show the standard deviation of the error in particle velocities $\sigma_{velocity}$ (as a percentage of the maximum velocity) as a function of the same maximum particle displacement D_{max}. In all three Figures, 23b, 24b and 25b, the behavior of the curves is similar to the behavior of the curves in Figures 23a, 24a and 25a, respectively, but the standard deviation is generaly higher than the average error for corresponding parameters. The smallest $\sigma_{velocity}$ of about 5 % is achieved with 150 particles in the picture, a vortex core size of 10 pixels and a maximum particle displacement of 20.

Figure 26 represents the percentage of particles found which could not be matched from frame to frame to yield velocity information (Section 2.3.3) The percentage of particles not matched is plotted against the maximum particle displacement D_{max} for the various vortex core sizes R_v. The number of particles present in the original image is equal to 50, 100 and 150 in Figures 26a, 26b and 26c, respectively. In all three graphs the percentage of particles not matched increases as N_p, D_{max} and R_v increase but remains very low (less than 2 %). Figure 27 shows the overall percentage of particles not matched through the entire process (particle identification in each frame and matching between frames). In other words, it is a measure of the information lost through the entire process. Figure 27a, 27b and 27c are obtained for a number of particles of 50, 100 and 150, respectively. The percentage of particles not matched is fairly insensitive to the structure of the flow field (D_{max}, R_v), but is directly proportional to the number of particles in the image. Once again, the limiting factor is particle overlap in both frames as well as particle in frame A convecting out of the image in frame b. In any case, 80 to 93 % of the actual particles in frame A result in a velocity determination through the whole process.

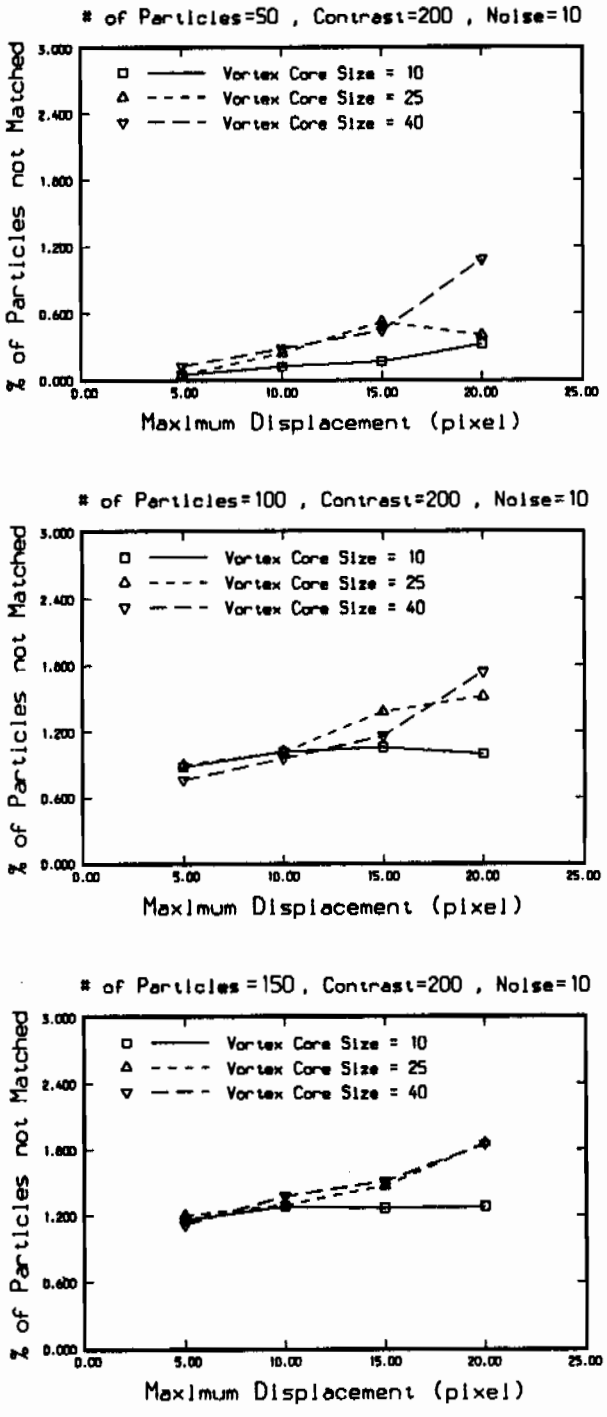


Figure 26. Dependence of percentage of particles not matched in velocity determination on maximum particle displacement and vortex core size for C = 200 and A = 10; a) N_p = 50, b) N_p = 100, and c) N_p = 150

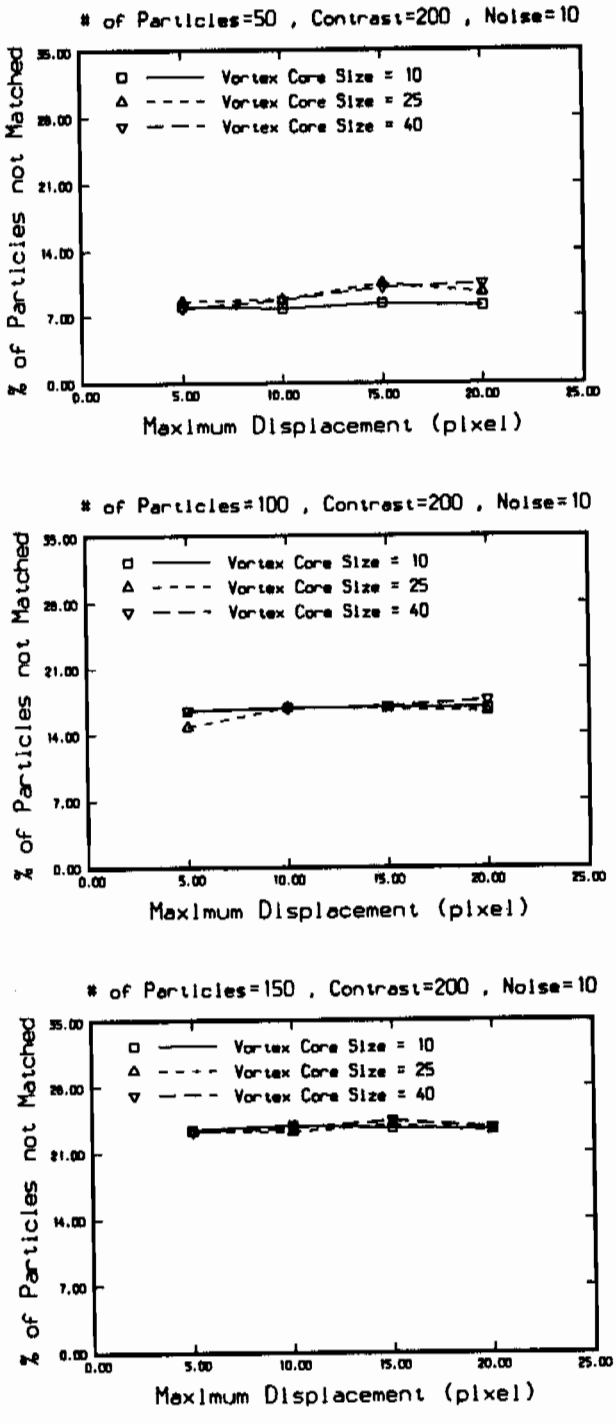


Figure 27. Dependence of percentage of particles not matched through overall process on maximum particle displacement and vortex core size for C = 200 and A = 10; a) N_p = 50, b) N_p = 100, and c) N_p = 150

3.3 DISCUSSION

Based on the evidence presented above, it is clear that the key operation in securing good valid results is choosing a correct threshold value. Providing that an appropriate threshold has been selected, the error in locating the particles is in general proportional to the number of particles, the size of the particles and the image noise and is inversely proportional to the image contrast. This also holds true for the percentages of particles not located and spurious particles. The details of the results are discussed in detail in the next few paragraphs.

3.3.1 Reliability of the choice of threshold

Choosing the "best" threshold value for every image examined is the most important task of the entire process. The particle identification is done on a thresholded image. All the high value pixels which appear in the thresholded image are identified as individual particles or parts of other particles. If the choice of threshold is "poor", pixels belonging to the background are identified as distinct particles. A "good" choice of threshold allows only the real particles to appear on the thresholded image and a correct identification to be done. Therefore, choosing the best threshold is very crucial. The threshold I_θ is chosen as the gray level intensity having an equal probability P_{I_θ} of belonging to the particles or the background. Hence, the determination of the correct I_θ relies upon an accurate characterization of the distribution of gray levels of the background and the particle. As described in section 2.2.1, the histogram of each picture is assumed to be a superposition of two normally distributed (gaussian) distributions. Based on this assumption, estimates of the means (μ_p' and μ_b') and standard deviations (σ_p' and σ_b') are made and the point of equal probability, (I_θ), is determined. Calculating the correct μ_b' and μ_p' guarantees the best possible threshold for that image. The calculation of the background mean μ_b' is always reliable regardless of the contrast and the

noise level, because it always corresponds to gray level intensity for which the probability distribution curve is maximum. However, the accuracy for calculating the correct μ_p' depends on the contrast and the noise level. In a high contrast, low noise image the two distributions are very distinct and the determination of μ_p' and μ_p' is very reliable and consequently the threshold value is reliable (very low P_{I_θ}). In a low contrast, high noise image the two distributions overlap which increases the probability of μ_p' to be wrong and increases the value of P_{I_θ}. Independent of the quality of the image, an I_θ is always calculated.

Figures 4a and 4b are examples of a successful calculation of I_θ for an image with high contrast and low noise. The two distributions are very distinct and the I_θ chosen definitely separates the image into the background and the particles. Figures 5a and 5b are examples of calculating I_θ for an image with an intermediate contrast and high noise level. While the particle distribution does not even show in the histogram of the original image (Figure 5a), is more apparent in the histogram of the smoothed image (Figure 5b), but there is still significant overlap between the two distributions. The threshold value chosen caused only two spurious particles to be detected processed image. The application of the threshold selection algorithm performed extremely well in this case, despite the relatively "poor" quality of the image. A case where the choice of I_θ is erroneous is illustrated in the low contrast, high noise image shown in Figures 6a and 6b. It is obvious that despite the image smoothing the particle distribution is not recognizable thus, not separable from the background distribution. The found threshold value of 156 caused 25 spurious particles to be detected in the processed image. Considering that particles can not be distinguished reliably from the noise in the original image (Figure 3a) even with the naked eye, a "bad" choice of the threshold is to be expected.

Based on the percentage of spurious particles identified in the processed image, the algorithm for choosing the threshold value was proven to be very reliable for all the images generated for the following combinations parameters: contrast (C) of 200 and 150 with noise of 0, 10, 20, 30 and 40 , contrast of 100 and noise of 0, 10, 20 and 30, and contrasts of 50 and 25 only when $A < 10$. In general, the failure of the algorithm seems insensitive to the number of particles N_p in the image and their size R.

As described above, the accuracy of the overall processing is intimately linked to the choice of a threshold. In other words, the image contrast and noise control the domain of validity of the overall procedure through the determination of the contrast. Hence, it is meaningless to examine the accuracy of the particle positions when the choice of the threshold is bad. The discussion of the results which follows will therefore concentrate on the effect of the various parameters within the domain of validity of the procedure.

3.3.2 Accuracy of the particle positions

The average error in locating the particles was found to be of the order of 0.5 pixels with 50 particles and 0.7 pixels with 100 particles in the image for an average particle size of 3 pixels in diameter. In other words, subpixel accuracy is achieved by this processing. Furthermore, the standard deviations of the error distributions around their mean are 0.9 and 1.2 pixels respectively. Due to the fact that the error cannot be negative, this indicates that the error distributions are very skewed with the majority of the particles located within a small fraction of a pixel and a small number of particles located with a large error (of the order of one pixel or more). It is believed that the particles with a large error correspond to cases of particle overlap, i.e, two or more real particles are detected as one and hence the centers may be off by a distance of the order of the particle radius. Further evidence of this comes from the fact that the error increases as the number of particles increases, due to the higher likelyhood of particle overlap (Figures 15 to 18).

Based on this evidence, it is believed that if particle overlap can be eliminated in a real application of PIV (by sharply focused light for example), the accuracy of the technique can probably be of the order of 0.2 to 0.3 pixels. Further investigation should be performed with this software. This particle overlap is also responsible for the increasing number of particles not found with increasing the number of particles in the image and the particle size (Figures 19 to 22). In any case, the number of overlapping particles is of the order of a low percent.

The number of spurious particles detected is always very low (less than 1 %) when the threshold is appropriate except in the case of very small particles on a noisy background (Figure 20). In other words, the particle size should be of the order of several pixels to be clearly distinguishable from pixel noise. This is particularly true for low number of particles since the particle level distribution becomes exceedingly small with respect to the background and renders the choice of the threshold somewhat sensitive.

3.3.3 Accuracy of the particle velocities

The accuracy of the velocity determination increases with the number of particles. This is probably linked to a better estimate of the velocity field by zonal cross-correlation and hence a better matching between successive frames. The maximum displacement of the particles between frames also increases the accuracy of the velocity. This is to be expected since the velocity is determined as the difference of two particle positions. The bigger the difference is, the smaller the error of such particle positions is in relative terms. The effect of the vortex core size is somewhat misleading since higher accuracy should be expected for a large vortex having small velocity gradients. However, the results indicate otherwise. This is possibly linked to the fact that a single isolated vortex is considered and a smaller number of particles is found in the case of a small vortex, hence contributing less to the overall error over the whole frame. Regardless of the various parameters, the distribution of the error on the velocity follows the distribution of the error of the position, i.e, the majority of the particles have a very small error (smaller than the mean) and a few particles have a large error. Further development of the algorithm could possibly lead to the rejection of the erroneous velocities by making use of the evolution of the velocity over a longer sequence of frames, or by spatial filtering of the reconstructed velocity field.

The percentage of particles not matched between frames (Figure 26) is usually very small (0.5 to 2 %) and increases with the number of particles and particle displacement. This

is to be expected but could be further reduced for larger sequences of images by relying on the zonal cross-correlation for the velocity estimation, but on an actual predictor corrector methods based on previous frames. The overall loss of information through the complete process (Figure 27) is never excessive and primarily governed by particle overlap (i.e, thickness of the light sheet) and particles moving out of the field of view between frames. This latter factor would become smaller with larger images (1024 X 1024 or more).

CHAPTER IV

CONCLUSIONS AND RECOMMENDATIONS

In this work, Particle Image Velocimetry algorithms were developed and their accuracy was systematically investigated for a large variety of image and flow parameters. Since the practical use of Particle Image Velocimetry requires the processing of large number of images, a substantial effort was devoted to develop robust algorithms which provide reliable results over a wide range of inputs, and furthermore, algorithms which are fully automatic and do not rely upon subjective operator inputs. The processing was organized into the following phases: image enhancement by removal of noise, automatic thresholding for optimal separation of particles from background, particle identification and determination of their positions, estimation of the unknown velocity field by zonal cross-correlation between successive time frames, and particle matching between pairs of successive images for velocity determination. By this sequence of operations, all parameters guiding the processing are extracted from the images themselves and no priori knowledge of the velocity field or the parameters of the image is necessary. The only assumptions implicit to the algorithms are that the images represent particles that are distinguishable from a background and are moving according to a spatially varying velocity field (to be determined) and that the image quality is degraded by additive, random (gaussian) noise. Under this broad set of assumptions, the algorithms performed remarkably well without operator input over a wide range of image parameters.

Unlike other studies where the accuracy of the results is inferred indirectly by some error analysis, the accuracy of the processing was evaluated directly in this study. Synthetic images were carefully generated with known parameters and information content and the results of the processing were compared directly to the true parameters. In this fashion, the errors were calculated instead of inferred. Statistics of the errors were then compiled by processing large data sets for good statistical convergence. The sensitivity of the results to the

various images parameters was then systematically evaluated. These extensive tests also validated the robustness of the processing. The parameters included the number of particles in the image, their average size and size distribution, the average contrast between the particles and the background, and the noise level in the image. Since the accuracy with which the velocity can be determined is intrinsically related to the accuracy with which particles can be located, most of the analysis was performed on the errors involved in locating the particles. Then, a representative subset of parameters was selected to carry out the error analysis on the velocity and the effect of the particle density and underlying structure of the velocity field was parametrically investigated.

The results allowed to determine the limit of validity of the processing in parameters space (i.e. the range of parameter values within which the results are meaningful) and also, within the acceptable domain, the influence of each parameter on the accuracy. The dominating factor controlling the validity of the processing is the ability to separate the particles from the background. To that effect, the most important part of the algorithm was to choose an appropriate threshold value. The proposed algorithm was proven to be very reliable and is capable of performing well even in cases where the particles are undistinguishable from the background with the naked eye. The failure of the algorithm results in the detection of a very large number of spurious particles.

By investigating the combined effect of the contrast and the noise level on the error in locating the particles, it is found that the results are valid for contrasts of 100, 150 and 200 and noise levels of 0, 10, 20, and 30. Within that range, the number of undetected particles is kept below 1 % and the number of spurious detections is very close to 0 %. Within this validity domain, the errors increase midly with the number of particles due to the higher likelyhood of particle overlap. A contrast of 50 can be used for noise level below 20 or better below 10. A contrast of 25 can be used with a noise level below 10 but is not recommended. Noise levels greater than 30 should only be used with very high contrasts. In practical applications of the Particle Image Velocimetry technique the goal should be to obtain pictures with the minimum noise profile and the maximum contrast between the particles and the background.

It was also found that the error increases as the mean radius of the particles increases (for a given number of particles). Once again, this relates to the higher likelyhood of particle overlap. In practical Particle Image Velocimetry, particle overlap is controlled by the width of the light sheet illuminating the particles and it can be practically eliminated with sharply collimated laser light sheets.

When the algorithm is applied on a vortex flow field which is prototypical of a small zone of turbulent flow, the results show that the velocity field prediction becomes more accurate as the number of particles in the image increases. This result, which contradicts the fact that the particle locations are less accurate with an increasing number of particles, can probably be attributed to the better estimation of the velocity field by cross-correlation which is an important step in finding pairs of matching particles in the determination of the velocity. Also, an increase in the maximum displacement of the particles between frames improves the error in velocity. The effect of the vortex core size (i.e., turbulence length scale) indicates that the average error in velocity over the entire image is decreasing with decreasing core size. This point is counterintuitive and needs to be investigated further. It is possibly due to the lesser percentage of particles within the vortex core and therefore the same effect would not be observed for a juxtaposition of several vortices (as in turbulent flows). For the best case, the average average error on the measured velocities is about 1.8 % with a standard deviation of 5 %. Such accuracy definitely validates the use of PIV for turbulence research. In summary, the proposed algorithms are capable of locating particles with subpixel accuracy and determine velocities within an accuracy comparable to probe techniques.

A future continuation of this project is needed and can be very rewarding. The next step should be to take actual pictures of real particles in seeded flow fields for which the velocity field is known, digitize them and use the algorithms developed to calculate the particle locations and their velocities in successive frames. This would provide in situ validation of the technique. In case that the particle density in the flow is large and particle overlap is a common phenomenon, color particles should be used and a color separation technique should be incorporated in the algorithms in order to identify particles of different color and achieve particle matching between overlapping particles. (It is very unlikely for overlapping particles to also have the same color). The algorithm should be generalized to handle larger (more

realistic) images with resolution of the order of 1024 X 1024 or more. Conceptually, this step does not pose any problems, provided that better hardware (high speed, larger memory computer) becomes available. The lengthiest part of the processing is the zonal estimation of the velocity by cross-correlation which could be implemented on specialized array processors to increase the speed. However, to process multiple frame sequences, this step is only required for the first pair of images since predictor-corrector schemes can be used for subsequent frames once velocity information becomes available (Economikos, 1988). Additional front end processing to improve image quality can also be applied to remove additional noise, background non-uniformities, etc. Applying even the best possible processing technique on a poor quality photograph produces a less accurate result than trying to get the best quality photograph possible and then apply any kind of processing technique. Thus, a great deal of importance should be given in understanding, improving and optimizing photographic techniques.

With these straight forward extensions, PIV has the potential for becoming a valuable tool for the study of turbulent processes.

APPENDIX A

PROGRAM "LOCATION"

APPENDIX A

```
                                        program location
c
      integer*2 data(4,256,256)
      dimension ix(20000),iy(20000),ipart(20000)
       dimension x(256),y(256),in(300)
       dimension ip(300),ra(300),amp1(300),par(300),bkg(300),amp2(300)
      common /imag/ data,ix,iy,ipart
c
       nim=50
       im=1
       open(unit=20,file='location.dat',form='formatted')
       call readfile(iset,kk,nn,mm,in,ip,ra,amp1,par,bkg,amp2)
       idum=-500
       do 1 m=1,kk
       index=in(m)
       ipartnum=ip(m)
       radlev=ra(m)
       amplrad=amp1(m)
       partlev=par(m)
       bkgndlev=bkg(m)
       ampl=amp2(m)
       do 2 i=1,nim
       call partgen(i,im,index,ipartnum,radlev,amplrad,partlev,
&      bkgndlev,ampl)
       call sub(im,ipartnum,nnum)
       call test(nnum,ipartnum,im)
2      continue
1      continue
       close(unit=20)
      stop
       end
c
c  ----------INPUT THE DATA----------
c
       subroutine readfile(iset,k,n,m,in,ip,ra,amp1,par,bkg,amp2)
c
       dimension ip(300),ra(300),amp1(300),par(300),bkg(300),amp2(300)
       dimension in(300)
       character*80 name
c
       k=0
       write(*,100)
```

```
100     format(' Supply the file name you want to read:  '$)
        read *,name
        write(*,103)
103     format(' From what set do you want to start ?.')
        read *,n
        write(*,104)
104     format(' Until what set do you want to read ?.')
        read *,m
        open(unit=11,file=name,status='old')
        read(11,105)iset
105     format(1x,i3)
        do 101 i=1,iset
        if (i .ge. n .and. i .le. m) then
        k=k+1
        read(11,102)in(k),ip(k),ra(k),amp1(k),par(k),bkg(k),amp2(k)
102     format(1x,i3,1x,i3,1x,f4.1,1x,f4.1,1x,f5.1,1x,f5.1,1x,f4.1)
        else
        read(11,106)ii,ir,ri,anp1,pir,blg,anp2
106     format(1x,i3,1x,i3,1x,f4.1,1x,f4.1,1x,f5.1,1x,f5.1,1x,f4.1)
        endif
101     continue
        close(unit=11)
        return
        end
c
c   ----------GENERATE AN IMAGE----------
c
        subroutine partgen(mim,im,index,ipartnum,radlev,amplrad,
&       partlev,bkgndlev,ampl)
c
        integer*2 data(4,256,256)
        dimension ix(20000),iy(20000),ipart(20000)
        dimension iiarea(999),xcoord(999),ycoord(999),rad(999)
      common /imag/ data,ix,iy,ipart
        common/wri/iiarea,xcoord,ycoord,rad
c
        do 5 n=1,ipartnum
        xcoord(n)=0.
        ycoord(n)=0.
5       continue
        do 6 i=1,256
        do 6 j=1,256
        var=bkgndlev+ampl*gasdev(idum)
        if (var .lt. 0.) then
        var=0.
        else if (var .gt. 255.) then
        var=255.
        end if
        data(im,i,j)=int(var)
6       continue
```

```
      do 10 n=1,ipartnum
      xcoord(n)=ran1(idum)/0.0039063
      if (xcoord(n) .gt. 256.) xcoord(n)=255.8
      if (xcoord(n) .lt. 1.) xcoord(n)=0.8
      ycoord(n)=ran1(idum)/0.0039063
      if (ycoord(n) .gt. 256.) ycoord(n)=255.8
      if (ycoord(n) .lt. 1.) ycoord(n)=0.8
      rad(n)=radlev+amplrad*gasdev(idum)
      if (rad(n) .lt. 1.5) then
      rad(n)=1.5
      end if
10    continue
      do 13 n=1,ipartnum
      iiarea(n)=0
      dummy1=xcoord(n)-rad(n)
      dummy2=xcoord(n)+rad(n)
      dummy3=ycoord(n)-rad(n)
      dummy4=ycoord(n)+rad(n)
      if (dummy1 .lt. 0.) dummy1=1.
      if (dummy3 .lt. 0.) dummy3=1.
      if (dummy2 .gt. 256.) dummy2=256.
      if (dummy4 .gt. 256.) dummy4=256.
      do 11 i=int(dummy1),int(dummy2)
      do 11 j=int(dummy3),int(dummy4)
      x=abs(float(i)-xcoord(n))
      y=abs(float(j)-ycoord(n))
      argu=x**2+y**2
      radi=sqrt(argu)
      if (radi .le. rad(n)) then
      data(im,i,j)=int(partlev+ampl*gasdev(idum))
      if (data(im,i,j) .gt. 255) data(im,i,j)=255
      if (data(im,i,j) .lt. 0) data(im,i,j)=0
      iiarea(n)=iiarea(n)+1
      end if
11    continue
13    continue
      write(20,103)index,mim,ipartnum,radlev,amplrad,partlev,
&     bkgndlev,ampl
103   format(1x,i3,1x,i3,1x,i3,1x,f4.1,1x,f4.1,1x,f5.1,1x,f5.1,
&     1x,f4.1)
&     bkgndlev,ampl
      return
      end
c
c  ----------PROCESS AN IMAGE----------
c
      subroutine sub(im,ipartnum,nnum)
c
      integer*2 data(4,256,256)
      dimension ix(20000),iy(20000),ipart(20000)
```

```
      dimension rxc(999),ryc(999),iarea(999),rrad(999)
      dimension rnumxc(999),rnumyc(999),rden(999),x(256),y(256)
      dimension m(11),er(11),sig(11),a(11)
      common /imag/ data,ix,iy,ipart
      common/subr/rxc,ryc,iarea,rrad
c
      do 31 i=1,999
      rnumxc(i)=0.
      rnumyc(i)=0.
      rden(i)=0.
      rxc(i)=0.
      ryc(i)=0.
      iarea(i)=0
31    continue
      do 32 i=1,20000
      ipart(i)=0
      ix(i)=0
      iy(i)=0
32    continue
      do 33 i=1,256
      do 33 j=1,256
      data(2,i,j)=0
      data(3,i,j)=0
      data(4,i,j)=0
33    continue
      do 29 i=1,256
      x(i)=0.
      y(i)=0.
29    continue
c
c  Determine if the image needs smoothing or not
c
      call histo(1,x,y,jk,rmax)
      perc=(float(jk)/256.)*100.
      if (perc .gt. 90.) then
      call mean0(y,rmax,mu1,mu2)
      sig1=0.
      sig2=0.
      ithresh=mu1+int(float(mu2-mu1)/2.)
      num=4
      do 760 i=1,256
      do 760 j=1,256
      data(num,i,j)=data(1,i,j)
760   continue
      goto 744
      endif
      num=4
c
c  Smooth the image
c
```

```
      do 10 i=2,255
      do 10 j=2,255
      rdata=(1./9.)*(data(im,i-1,j-1)+data(im,i-1,j)+data(im,i-1,j+1)
&      +data(im,i,j-1)+data(im,i,j+1)+data(im,i,j)
&      +data(im,i+1,j-1)+data(im,i+1,j)+data(im,i+1,j+1))
      data(num,i,j)=ifix(rdata)
10     continue
      do 34 i=1,1
      do 34 j=1,256
      data(num,i,j)=data(num,i+1,j)
34     continue
      do 35 i=1,256
      do 35 j=256,256
      data(num,i,j)=data(num,i,j-1)
35     continue
      do 36 i=256,256
      do 36 j=1,256
      data(num,i,j)=data(num,i-1,j)
36     continue
      do 37 i=1,256
      do 37 j=1,1
      data(num,i,j)=data(num,i,j+1)
37     continue
c
c  Find the threshold value
c
      call histo(num,x,y,jk,rmax)
      call mean1(y,mu1)
      mu=mu1
      do 557 i=1,11
      m(i)=0
      er(i)=0.
      sig(i)=0.
      a(i)=0.
557    continue
      do 550 i=1,11
      err=0.
      mu1=mu-6+i
      if (mu1 .lt. 1) mu1=1
      call stddev(y,mu1,sig1,a1,err,cc)
      if (cc .gt. 0.) err=1.0E20
      m(i)=mu1
      er(i)=err
      sig(i)=sig1
      a(i)=a1
550    continue
      err=amin1(er(1),er(2),er(3),er(4),er(5),er(6),er(7),er(8)
&      ,er(9),er(10),er(11))
      do 551 i=1,11
      if (er(i) .eq. err) then
```

```
      mu1=m(i)
      sig1=sig(i)
      a1=a(i)
      endif
551   continue
      do 500 i=1,256
      dummy=a1*exp(-(x(i)-mu1)**2/sig1**2)
      y(i)=y(i)-dummy
      if (y(i) .lt. 0) y(i)=0
500   continue
      ilow = mu1+int(1.5*sig1+.5)
      call mean(y,mu2,ilow,256)
      mu=mu2
      do 556 i=1,11
      m(i)=0
      er(i)=0.
      sig(i)=0.
      a(i)=0.
556   continue
      do 552 i=1,11
      err=0.
      mu2=mu-6+i
      if (mu2 .gt. 256) mu2=256
      call stddev(y,mu2,sig2,a2,err,cc)
      if (cc .gt. 0.) err=1.0E35
      m(i)=mu2
      er(i)=err
      sig(i)=sig2
      a(i)=a2
552   continue
      err=amin1(er(1),er(2),er(3),er(4),er(5),er(6),er(7),er(8),
&     er(9),er(10),er(11))
      do 553 i=1,11
      if (er(i) .eq. err) then
      mu2=m(i)
      sig2=sig(i)
      a2=a(i)
      endif
553   continue
      do 600 i=mu1,mu2
      p1 = a1*exp(-(i-mu1)**2/sig1**2)
      p2 = a2*exp(-(i-mu2)**2/sig2**2)
      if (p1 .le. p2) goto 650
600   continue
650   ithresh=i
      p=(p1+p2)/2.
c
c  Threshold the image.
c
744   iim=0
```

```
      do 21 i=1,256
      do 21 j=1,256
      if (data(num,i,j) .lt. ithresh) then
      data(2,i,j)=0
      else
      data(2,i,j)=250
        iim=iim+1
      endif
21     continue
        perc=(float(iim)/256.0**2)*100.
        write(20,559)mu1,sig1,mu2,sig2,ithresh,perc
559     format(4x,i3,1x,f7.4,1x,i3,1x,f7.4,1x,i3,1x,f7.4)
c
c   Identify and locate the particles.
c
      nnum=0
      l=1
      im=2
      if (data(im,1,1) .ne. 250) goto 30
      ix(1)=1
      iy(1)=1
      ipart(1)=1
      data(3,1,1)=nnum+1
      l=l+1
      nnum=nnum+1
30     continue
      do 42 i=1,1
      do 42 j=2,256
      if (data(im,i,j) .ne. 250) goto 42
      if (data(im,i,j) .eq. data(im,i,j-1)) then
      ix(l)=i
      iy(l)=j
      ipart(l)=data(3,i,j-1)
      data(3,i,j)=data(3,i,j-1)
      l=l+1
      else
      ix(l)=i
      iy(l)=j
      ipart(l)=nnum+1
      data(3,i,j)=nnum+1
      l=l+1
      nnum=nnum+1
      endif
42     continue
      do 41 j=1,1
      do 41 i=2,256
      if (data(im,i,j) .ne. 250) goto 41
      if (data(im,i,j) .eq. data(im,i-1,j)) then
      ix(l)=i
      iy(l)=j
```

```
      ipart(l)=data(3,i-1,j)
      data(3,i,j)=data(3,i-1,j)
      l=l+1
      else
      ix(l)=i
      iy(l)=j
      ipart(l)=nnum+1
      data(3,i,j)=nnum+1
      l=l+1
      nnum=nnum+1
      endif
41     continue
      nn=0
      do 40 i=2,256
      do 40 j=2,256
      if (data(im,i,j) .ne. 250) goto 40
      if (data(im,i,j) .ne. data(im,i,j-1) .and.
&      data(im,i,j) .ne. data(im,i-1,j-1) .and.
&      data(im,i,j) .ne. data(im,i-1,j) .and.
&      data(im,i,j) .ne. data(im,i-1,j+1)) then
      ix(l)=i
      iy(l)=j
      ipart(l)=nnum+1
       data(3,i,j)=nnum+1
      l=l+1
      nnum=nnum+1
       if (nnum .gt. 350) return
       if (l .gt. 20000) return
      else
      if (data(im,i,j) .eq. data(im,i,j-1)) then
      ix(l)=i
      iy(l)=j
      ipart(l)=data(3,i,j-1)
      data(3,i,j)=data(3,i,j-1)
      if (data(3,i,j) .ne. data(3,i-1,j) .and. data(3,i-1,j) .ne.
&      0) then
      igreat=data(3,i-1,j)
      ilow=data(3,i,j)
      call compare(i,j,l,ilow,igreat,nnum)
      endif
      endif
      if (data(im,i,j) .eq. data(im,i-1,j-1)) then
      ix(l)=i
      iy(l)=j
      ipart(l)=data(3,i-1,j-1)
      data(3,i,j)=data(3,i-1,j-1)
      if (data(3,i,j) .ne. data(3,i,j-1) .and. data(3,i,j-1) .ne.
&      0) then
      igreat=data(3,i,j-1)
      ilow=data(3,i,j)
```

```
      call compare(i,j,l,ilow,igreat,nnum)
      endif
      if (data(3,i,j) .ne. data(3,i-1,j) .and. data(3,i-1,j) .ne.
&       0) then
      igreat=data(3,i-1,j)
      ilow=data(3,i,j)
      call compare(i,j,l,ilow,igreat,nnum)
      endif
      endif
      if (data(im,i,j) .eq. data(im,i-1,j)) then
      ix(l)=i
      iy(l)=j
      ipart(l)=data(3,i-1,j)
      data(3,i,j)=data(3,i-1,j)
      if (data(3,i,j) .ne. data(3,i,j-1) .and. data(3,i,j-1) .ne.
&       0) then
      igreat=data(3,i,j-1)
      ilow=data(3,i,j)
      call compare(i,j,l,ilow,igreat,nnum)
      endif
      endif
      if (data(im,i,j) .eq. data(im,i-1,j+1)) then
      ix(l)=i
      iy(l)=j
      ipart(l)=data(3,i-1,j+1)
      data(3,i,j)=data(3,i-1,j+1)
      if (data(3,i,j) .ne. data(3,i,j-1) .and. data(3,i,j-1) .ne.
&       0) then
      igreat=data(3,i,j-1)
      ilow=data(3,i,j)
      call compare(i,j,l,ilow,igreat,nnum)
      endif
      if (data(3,i,j) .ne. data(3,i-1,j) .and. data(3,i-1,j) .ne.
&       0) then
      igreat=data(3,i-1,j)
      ilow=data(3,i,j)
      call compare(i,j,l,ilow,igreat,nnum)
      endif
      endif
      l=l+1
       if (l .gt. 20000) return
      endif
40     continue
       l=l-1
       im=1
       if (nnum .gt. 350) return
      do 70 ik=1,l
       in=ipart(ik)
      rnumxc(in)=rnumxc(in)+float(ix(ik)*data(num,ix(ik),iy(ik)))
      rnumyc(in)=rnumyc(in)+float(iy(ik)*data(num,ix(ik),iy(ik)))
```

```
      rden(in)=rden(in)+float(data(num,ix(ik),iy(ik)))
       iarea(in)=iarea(in)+1
70     continue
c
       do 69 n=1,nnum
       rxc(n)=rnumxc(n)/rden(n)
       if (rxc(n) .gt. 256.) rxc(n)=255.8
       if (rxc(n) .lt. 1.) rxc(n)=0.8
       ryc(n)=rnumyc(n)/rden(n)
       if (ryc(n) .gt. 256.) ryc(n)=255.8
       if (ryc(n) .lt. 1.) ryc(n)=0.8
       rrad(n)=sqrt(float(iarea(n))/3.14159)
69     continue
      return
      end
c
c  ------------COMPARE PARTICLE NUMBERS------------
c
      subroutine compare(i,j,l,ilow,igreat,nnum)
c
      integer*2 data(4,256,256)
      dimension ipart(20000),ix(20000),iy(20000)
      common /imag/ data,ix,iy,ipart
c
      if (ilow .gt. igreat) then
      do 64 k=1,l
      if (ipart(k) .eq. ilow) then
      ipart(k)=igreat
       else if (ipart(k) .gt. ilow) then
       ivar=ipart(k)-ilow
       ipart(k)=ilow+(ivar-1)
       endif
64     continue
      do 66 k=1,i-1
      do 66 n=1,256
      if (data(3,k,n) .eq. ilow) then
      data(3,k,n)=igreat
       else if (data(3,k,n) .gt. ilow) then
       ivar=data(3,k,n)-ilow
       data(3,k,n)=ilow+(ivar-1)
       endif
66     continue
      do 68 k=i,i
      do 68 n=1,j
      if (data(3,k,n) .eq. ilow) then
      data(3,k,n)=igreat
       else if (data(3,k,n) .gt. ilow) then
       ivar=data(3,k,n)-ilow
       data(3,k,n)=ilow+(ivar-1)
       endif
```

```
68      continue
        ilow=igreat
        nnum=nnum-1
       else
       do 65 k=1,l
       if (ipart(k) .eq. igreat) then
       ipart(k)=ilow
         else if (ipart(k) .gt. igreat) then
         ivar=ipart(k)-igreat
         ipart(k)=igreat+(ivar-1)
         endif
65      continue
       do 67 k=1,i-1
       do 67 n=1,256
       if (data(3,k,n) .eq. igreat) then
       data(3,k,n)=ilow
         else if (data(3,k,n) .gt. igreat) then
         ivar=data(3,k,n)-igreat
         data(3,k,n)=igreat+(ivar-1)
         endif
67      continue
       do 69 k=i,i
       do 69 n=1,j
       if (data(3,k,n) .eq. igreat) then
       data(3,k,n)=ilow
         else if (data(3,k,n) .gt. igreat) then
         ivar=data(3,k,n)-igreat
         data(3,k,n)=igreat+(ivar-1)
         endif
69      continue
        igreat=ilow
        nnum=nnum-1
       endif
       return
       end
c
c   ----------CALCULATE THE HISTOGRAM OF THE IMAGE----------
c
       subroutine histo(im,x,y,kk,rmax)
c
       integer*2 data(4,256,256)
       dimension y(256),x(256)
         dimension ix(20000),iy(20000),ipart(20000)
       common /imag/ data,ix,iy,ipart
       data n /256/
c
         kk=0
         rmax=0.
       do 10 i=1,256
       do 20 j=1,256
```

```
      if (im .eq. 1) ival=data(im,i,j)
      if (im .eq. 2) ival=data(im,i,j)
      if (im .eq. 3) ival=data(im,i,j)
      if (im .eq. 4) ival=data(im,i,j)
      y(ival+1)=y(ival+1)+1
20     continue
10     continue
      do 30 i=1,n
      x(i)=i-1
       if (y(i) .eq. 0.) kk=kk+1
       rmax=amax1(rmax,y(i))
30     continue
      return
      end
c
c  -----CALCULATE  BOTH MEANS WHEN SMOOTHING IS NOT REQUIRED-----
c
       subroutine mean0(y,rmax,mu1,mu2)
       dimension y(256)
c
       mu1=256
       rmax2=0.
       do 100 i=1,256
       if (y(i) .eq. rmax) mu1=i-1
       if (i .gt. mu1+10) rmax2=amax1(rmax2,y(i))
100    continue
       do 101 i=mu1+10,256
       if (y(i) .eq. rmax2) mu2=i-1
101    continue
       return
       end
c
c  -------CALCULATE THE FIRST MEAN OF THE SMOOTHED IMAGE-------
c
       subroutine mean1(y,mu)
       dimension y(256)
c
       rmu=0.
       do 100 i=2,256
       rmu=amax1(rmu,y(i))
100    continue
       do 101 i=2,256
       if (y(i) .eq. rmu) then
       mu=i
       goto 200
       endif
101    continue
200    return
       end
c
```

```
c  -------CALCULATE THE SECOND MEAN OF THE SMOOTHED IMAGE-------
c
      subroutine mean(y,mu,il,ih)
      dimension y(256),yy(256),ys(256)
c
      iter=0
      do 100 i=il,ih
      yy(i)=y(i)
100   continue
150   sum=0.
      do 200 i=il+1,ih-1
      ys(i)=(yy(i+1)+yy(i)+yy(i-1))/3.
      sum=sum+(ys(i)-yy(i))**2
200   continue
      ys(256)=(yy(256)+yy(255))/2.
      sum=sum+(ys(256)-yy(256))**2
      sum=sqrt(sum/(float(ih-il-2)))
      do 250 i=il,ih
      yy(i)=ys(i)-sum
      if (yy(i) .lt. 0.) yy(i)=0.
250   continue
280   ndet=0
      do 300 i=il+1,ih-1
      if (yy(i) .gt. yy(i-1) .and. yy(i) .gt. yy(i+1)) then
      ndet=ndet+1
      mu=i
      endif
300   continue
      if (ndet .gt. 2) goto 150
      if (ndet .lt. 1) then
      if (iter .gt. 10) then
      mu=il
      return
      endif
      sum=sum/2.
      iter=iter+1
      do 350 i=il,ih
      yy(i)=ys(i)-sum
350   continue
      goto 280
      endif
      return
      end
c
c  ------------CALCULATE THE STANDARD DEVIATION------------
c
      subroutine stddev(y,mu,sig,a,err,cc)
      dimension y(256)
c
      sx=0.
```

```
      sxx=0.
      sy=0.
      sxy=0.
      np=0
      i1=mu-10
      i2=mu+10
      if (i1 .lt. 1 ) i1=1
      if (i2 .gt. 256) i2=256
      do 100 i=1,256
      if (y(i) .eq. 0.) goto 100
      if (i .lt. i1 .or. i .gt. i2) goto 100
      sx=sx+(i-mu)**2
      sxx=sxx+((i-mu)**2)**2
      sxy=sxy+alog(y(i))*(i-mu)**2
      sy=sy+alog(y(i))
      np=np+1
100   continue
      den=sx**2-sxx*np
      cc=(sy*sx-sxy*np)/den
      if (cc .gt. 0.) return
      rlna=(sx*sxy-sxx*sy)/den
      sig=sqrt(-(1./cc))
      a=exp(rlna)
      do 200 i=i1,i2
      var1=a*exp((-(i-mu)**2)/sig**2)
      var=(y(i)-var1)**2
      err=err+var
200   continue
      return
      end
c
c  ----------MATCH THE PARTICLES IN PAIRS----------
c
      subroutine test(nnum,ipartnum,im)
c
      integer*2 data(4,256,256)
      dimension ix(20000),iy(20000),ipart(20000)
      dimension rxc(999),ryc(999),iarea(999),rrad(999)
      dimension iiarea(999),xcoord(999),ycoord(999),rad(999)
      dimension x(256),y(256),ii(999),in(999),iix(999),iiy(999)
      dimension rad1(256,256),rad2(256,256),radius(999)
      dimension xi(256),yj(256),xk(256),yl(256),ik(256,256),
&           xx(999),yy(999)
      common /imag/ data,ix,iy,ipart
      common/subr/rxc,ryc,iarea,rrad
      common/wri/iiarea,xcoord,ycoord,rad
c
      im=1
      if (nnum .gt. 350) goto 922
      do 400 i=1,999
```

```
      in(i)=0
      iix(i)=0
      iiy(i)=0
      radius(i)=0.
      xx(i)=0.
      yy(i)=0.
      ii(i)=0
400   continue
      do 401 i=1,256
      xk(i)=0.
      yl(i)=0.
      xi(i)=0.
      yj(i)=0.
401   continue
      do 80 i=1,256
      do 80 j=1,256
      data(2,i,j)=0
      data(3,i,j)=0
      data(4,i,j)=0
      rad1(i,j)=0.
      rad2(i,j)=0.
      ik(i,j)=0
80    continue
      do 73 n=1,nnum
      k = int(rxc(n)+.5)
      l = int(ryc(n)+.5)
      data(3,k,l)=2
      xk(k)=rxc(n)
      yl(l)=ryc(n)
      rad2(k,l)=rrad(n)
73    continue
      do 74 n=1,ipartnum
      i = int(xcoord(n)+.5)
      j = int(ycoord(n)+.5)
      data(2,i,j)=1
      xi(i)=xcoord(n)
      yj(j)=ycoord(n)
      rad1(i,j)=rad(n)
74    continue
      k=0
      sum=0.
      sumsq=0.
      do 77 i=1,256
      do 77 j=1,256
      data(4,i,j)=data(2,i,j)+data(3,i,j)
      if (data(4,i,j) .ne. 0) then
      k=k+1
      if (data(4,i,j) .eq. 1) then
      radius(k)=rad1(i,j)
      ik(i,j)=k
```

```
      xx(k)=xi(i)
      yy(k)=yj(j)
      else if (data(4,i,j) .eq. 2) then
      radius(k)=rad2(i,j)
      ik(i,j)=k
      xx(k)=xk(i)
      yy(k)=yl(j)
      else if (data(4,i,j) .eq. 3) then
      ik(i,j)=k
      eer=(xi(i)-xk(i))**2+(yj(j)-yl(j))**2
      sumsq=sumsq+eer
      sum=sum+sqrt(eer)
      radius(k)=0.
      end if
      in(k)=data(4,i,j)
      data(4,i,j)=13
      iix(k)=i
      iiy(k)=j
      end if
77    continue
      ipf=0
      do 75 i=1,k
      if (in(i) .eq. 3) then
      l=0
      in(i)=13
      ipf=ipf+1
      else if (in(i) .eq. 2) then
      l=0
      ivar=int(radius(i)+.5)
      idummy1=iix(i)-ivar
      if (idummy1 .lt. 1) idummy1=1
      idummy2=iix(i)+ivar
      if (idummy2 .gt. 256) idummy2=256
      idummy3=iiy(i)-ivar
      if (idummy3 .lt. 1) idummy3=1
      idummy4=iiy(i)+ivar
      if (idummy4 .gt. 256) idummy4=256
      do 305 n=idummy1,idummy2
      do 305 m=idummy3,idummy4
      if (data(2,n,m) .eq. 1) then
      l=l+1
      in(ik(n,m))=11
      eer=(xi(n)-xx(i))**2+(yj(m)-yy(i))**2
      sumsq=sumsq+eer
      sum=sum+sqrt(eer)
      data(4,n,m)=11
      end if
305   continue
      data(4,iix(i),iiy(i))=12
      if (l .ne. 0) then
```

```
      in(i)=12
      ipf=ipf+1
      end if
      end if
75    continue
      l=0
      ll=0
      do 310 i=1,k
      if (in(i) .ne. 11 .and. in(i) .ne. 12 .and. in(i) .ne. 13) then
      l=l+1
      ii(l)=i
      end if
310   continue
      if (l .eq. 0) goto 200
      do 308 i=1,l
      if (in(ii(i)) .eq. 1) then
      ivar=int(2.*radius(ii(i))+.5)
      idummy1=iix(ii(i))-ivar
      if (idummy1 .lt. 1) idummy1=1
      idummy2=iix(ii(i))+ivar
      if (idummy2 .gt. 256) idummy2=256
      idummy3=iiy(ii(i))-ivar
      if (idummy3 .lt. 1) idummy3=1
      idummy4=iiy(ii(i))+ivar
      if (idummy4 .gt. 256) idummy4=256
      do 309 n=idummy1,idummy2
      do 309 m=idummy3,idummy4
      if(data(4,n,m) .eq. 2) then
      ll=ll+1
      in(ik(n,m))=12
      eer=(xi(n)-xx(ii(i)))**2+(yj(m)-yy(ii(i)))**2
      sumsq=sumsq+eer
      sum=sum+sqrt(eer)
      data(4,n,m)=14
      end if
309   continue
      if (ll .ne. 0) then
      ipf=ipf+ll
      in(ii(i))=14
      end if
      ll=0
      end if
308   continue
      km=0
      nm=0
      do 99 i=1,k
      if (in(i) .eq. 1) km=km+1
      if (in(i) .eq. 2) nm=nm+1
99    continue
200   rmean=sum/float(ipf)
```

```
      du=(sumsq/float(ipf))-rmean**2
      if (du .lt. 0.) du=0.
      rsigm=sqrt(du)
      goto 923
922   ipf=-1
      km=-1
      nm=-1
      rmean=-1.0
      rsigm=-1.0
923   write(20,253)ipf,km,nm,rmean,rsigm
253   format(4x,i3,1x,i3,1x,i3,1x,f7.4,1x,f7.4)
800   return
      end
c
c  ----------CALCULATE RANDOMLY DISTRIBUTED DEVIATES----------
c
      function ran1(idum)
c
      dimension r(97)
      parameter (m1=259200,ia1=7141,ic1=54773,rm1=1./m1)
      parameter (m2=134456,ia2=8121,ic2=28411,rm2=1./m2)
      parameter (m3=423000,ia3=4561,ic3=51349)
      data iff/0/
c
      if (idum .lt. 0 .or. iff .eq. 0) then
      iff=1
      ix1=mod(ic1-idum,m1)
      ix1=mod(ia1*ix1+ic1,m1)
      ix2=mod(ix1,m2)
      ix1=mod(ia1*ix1+ic1,m1)
      ix3=mod(ix1,m3)
      do 200 j=1,97
      ix1=mod(ia1*ix1+ic1,m1)
      ix2=mod(ia2*ix2+ic2,m2)
      r(j)=(float(ix1)+float(ix2)*rm2)*rm1
200   continue
      idum=1
      end if
      ix1=mod(ia1*ix1+ic1,m1)
      ix2=mod(ia2*ix2+ic2,m2)
      ix3=mod(ia3*ix3+ic3,m3)
      j=1+(97*ix3)/m3
      if (j .gt. 97 .or. j .lt. 1) pause
      ran1=r(j)
      r(j)=(float(ix1)+float(ix2)*rm2)*rm1
      return
      end
c
c  ----------CALCULATE NORMALLY DISTRIBUTED DEVIATES----------
c
```

```
      function gasdev(idum)
      data iset/0/
c
      if (iset .eq. 0) then
1     v1=2.*ran1(idum)-1
      v2=2.*ran1(idum)-1
      r=v1**2+v2**2
      if (r .ge. 1.) goto 1
      fac=sqrt(-2.*log(r)/r)
      gset=v1*fac
      gasdev=v2*fac
      iset=1
      else
      gasdev=gset
      iset=0
      end if
      return
      end
```

APPENDIX B

PROGRAM "VELOCITY"

APPENDIX B

```
                                        program velocity
c
      integer*2 data(4,256,256)
        dimension rx(5,200),ry(5,200),rad1(200)
        dimension xx(5,200),yy(5,200),rrad(5,200),xx1(5,200),yy1(5,200)
        dimension x1(32,32),x2(32,32),vi(4,4),vj(4,4)
      dimension ix(20000),iy(20000),ipart(20000)
        dimension x(256),y(256),in(100),nnum(10),u(200),v(200)
        dimension ip(100),ra(100),amp1(100),par(100),bkg(100),amp2(100)
      common /imag/ data,ix,iy,ipart
        common /cen/ rx,ry,rad1
        common /ken/ xx,yy,rrad,xx1,yy1
        common /vlc/x1,x2
        common /men/ u,v
c
        im=1
        nim=50
        idum=-500
        open(unit=20,file='velocity.dat',form='formatted')
        call readfile(iset,kk,nn,mm,in,ip,ra,amp1,par,bkg,amp2)
        print *,' Supply the various velocities'
        print *,' vel1,vel2,vel3,vel4'
        read *,vel1,vel2,vel3,vel4
        print *,' Supply the various vortex core sizes in the form'
        print *,'ar1,ar2,ar3'
        read *,ar1,ar2,ar3
        print *,' Supply the time interval between successive frames'
        read *,delt
        do 1 mm=1,kk
        do 3 kh=1,4
        if (kh .eq. 1) vel=vel1
        if (kh .eq. 2) vel=vel2
        if (kh .eq. 3) vel=vel3
        if (kh .eq. 4) vel=vel4
        do 4 kf=1,3
        if (kf .eq. 1) ar=ar1
        if (kf .eq. 2) ar=ar2
        if (kf .eq. 3) ar=ar3
        do 2 iu=1,nim
        num=0
        index=in(mm)
        ipartnum=ip(mm)
```

```
      radlev=ra(mm)
      amplrad=amp1(mm)
      partlev=par(mm)
      bkgndlev=bkg(mm)
      ampl=amp2(mm)
      call partgen(iu,idum,im,index,ipartnum,radlev,amplrad,partlev,
&     bkgndlev,ampl)
      iq=1
      call sub(iq,1,im,ipartnum,num)
      if (num .gt. 250) goto 908
      call velo(ipartnum,vel,delt,ar)
      nnum(1)=num
      ih1=num
      ii=2
      call velocity(ii,ipartnum,vel,delt,ar)
      call compim(ii,idum,im,index,ipartnum,radlev,amplrad,partlev,
&     bkgndlev,ampl)
      iq=2
     call sub(iq,ii,im,ipartnum,num)
      if (num .gt. 250) goto 908
      nnum(2)=num
      ih2=num
      i2=0
      np=32
      do 82 ll=1,32
      do 82 lj=1,32
      x1(ll,lj)=0.
      x2(ll,lj)=0.
82    continue
      do 81 ll=1,4
      do 81 lj=1,4
      vi(ll,lj)=0.
      vj(ll,lj)=0.
81    continue
c
c  Perform cross-correlation
c
      do 85 io=1,4
      i1=i2+1
      i2=io*64
      ia=0
      do 80 ih=1,4
      ib=ia+1
      ia=ih*64
      it=0
      il=0
c
c  Divide the image into 16 blocks 16 X 16 in dimensions
c
      do 105 i=i1,i2,2
```

```
      it=it+1
      do 104 j=ib,ia,2
      il=il+1
      x1(it,il)=0.25*(data(2,i,j)+data(2,i+1,j)+data(2,i,j+1)+
&     data(2,i+1,j+1))
      x2(it,il)=0.25*(data(1,i,j)+data(1,i+1,j)+data(1,i,j+1)+
&     data(1,i+1,j+1))
104   continue
      il=0
105   continue
      call velavg(np,delt,vx,vy)
      vi(io,ih)=2.*vx
      vj(io,ih)=2.*vy
80    continue
85    continue
50    continue
      do 10 i=1,nnum(1)
      uint=0.
      vint=0.
      ibx=int((xx(1,i)/64.)+.5)
      if (ibx .eq. 0) ibx=1
      iby=int((yy(1,i)/64.)+.5)
      if (iby .eq. 0) iby=1
      call interp1(vi,xx(1,i),yy(1,i),uint)
      call interp2(vj,xx(1,i),yy(1,i),vint)
      xx1(1,i)=uint*delt+xx(1,i)
      yy1(1,i)=vint*delt+yy(1,i)
10    continue
      ib=0
c
c  Disregard the particles which exit the frame
c
      do 12 i=1,nnum(1)
      if (xx1(1,i) .gt. 256. .or. yy1(1,i) .gt. 256. .or. xx1(1,i) .lt.
&     0. .or. yy1(1,i) .lt. 0.) then
      ib=ib+1
      do 11 n=i,nnum(1)
      xx1(1,n)=xx1(1,n+1)
      xx(1,n)=xx(1,n+1)
      yy1(1,n)=yy1(1,n+1)
      yy(1,n)=yy(1,n+1)
      u(n)=u(n+1)
      v(n)=v(n+1)
      rrad(1,n)=rrad(1,n+1)
11    continue
      i=i-1
      end if
12    continue
      nnum(1)=nnum(1)-ib
908   call test(1,2,nnum(1),nnum(2),delt,vel,ar,im,num)
```

```
2       continue
4       continue
3       continue
1       continue
        close(unit=20)
900     stop
        end
c
c  ----------INPUT THE DATA----------
c
        subroutine readfile(iset,k,n,m,in,ip,ra,amp1,par,bkg,amp2)
c
        dimension ip(100),ra(100),amp1(100),par(100),bkg(100),amp2(100)
        dimension in(100)
        character*80 name
c
        k=0
        write(*,100)
100     format(' Supply the file name you want to read:  '$)
        read *,name
        write(*,103)
103     format(' From what set do you want to start ?.')
        read *,n
        write(*,104)
104     format(' Until what set do you want to read ?.')
        read *,m
        open(unit=11,file=name,status='old')
        read(11,105)iset
105     format(1x,i3)
        do 101 i=1,iset
        if (i .ge. n .and. i .le. m) then
        k=k+1
        read(11,102)in(k),ip(k),ra(k),amp1(k),par(k),bkg(k),amp2(k)
102     format(1x,i3,1x,i3,1x,f4.1,1x,f4.1,1x,f5.1,1x,f5.1,1x,f4.1)
        write(*,102)in(k),ip(k),ra(k),amp1(k),par(k),bkg(k),amp2(k)
        else
        read(11,106)ii,ir,ri,anp1,pir,blg,anp2
106     format(1x,i3,1x,i3,1x,f4.1,1x,f4.1,1x,f5.1,1x,f5.1,1x,f4.1)
        endif
101     continue
        close(unit=11)
        return
        end
c
c  ----------GENERATE AN IMAGE IN TIME FRAME A----------
c
        subroutine partgen(iu,idum,im,index,ipartnum,radlev,amplrad,
&       partlev,bkgndlev,ampl)
c
        integer*2 data(4,256,256)
```

```
      dimension rx(5,200),ry(5,200),iiarea(200)
      dimension ix(20000),iy(20000),ipart(20000),rad1(200)
     common /imag/ data,ix,iy,ipart
      common /cen/ rx,ry,rad1
c
      do 5 i=1,ipartnum
      rx(1,i)=0.
      ry(1,i)=0.
5     continue
      do 6 i=1,256
      do 6 j=1,256
      var=bkgndlev+ampl*gasdev(idum)
      if (var .lt. 0.) then
      var=0.
      else if (var .gt. 255.) then
      var=255.
      end if
      data(im,i,j)=int(var)
6     continue
      do 10 n=1,ipartnum
      rx(1,n)=ran1(idum)/0.0039063
      if (rx(1,n) .gt. 256.) rx(1,n)=255.8
      if (rx(1,n) .lt. 1.) rx(1,n)=0.8
      ry(1,n)=ran1(idum)/0.0039063
      if (ry(1,n) .gt. 256.) ry(1,n)=255.8
      if (ry(1,n) .lt. 1.) ry(1,n)=0.8
      rad1(n)=radlev+amplrad*gasdev(idum)
      if (rad1(n) .lt. 1.5) rad1(n)=1.5
10    continue
      do 13 n=1,ipartnum
      iiarea(n)=0
      dummy1=rx(1,n)-rad1(n)
      dummy2=rx(1,n)+rad1(n)
      dummy3=ry(1,n)-rad1(n)
      dummy4=ry(1,n)+rad1(n)
      if (dummy1 .lt. 0.) dummy1=1.
      if (dummy3 .lt. 0.) dummy3=1.
      if (dummy2 .gt. 256.) dummy2=256.
      if (dummy4 .gt. 256.) dummy4=256.
      do 11 i=int(dummy1),int(dummy2)
      do 11 j=int(dummy3),int(dummy4)
      x=abs(float(i)-rx(1,n))
      y=abs(float(j)-ry(1,n))
      argu=x**2+y**2
      radi=sqrt(argu)
      if (radi .le. rad1(n)) then
      data(im,i,j)=int(partlev+ampl*gasdev(idum))
      if (data(im,i,j) .gt. 255) data(im,i,j)=255
      if (data(im,i,j) .lt. 0) data(im,i,j)=0
      iiarea(n)=iiarea(n)+1
```

```
        end if
11      continue
13      continue
        write(20,103)index,iu,ipartnum,radlev,amplrad,partlev,
&       bkgndlev,ampl
103     format(1x,i3,1x,i3,1x,i3,1x,f4.1,1x,f4.1,1x,f5.1,1x,f5.1,
&       1x,f4.1)
        return
        end
c
c  ----------PROCESS AN IMAGE----------
c
      subroutine sub(iq,k,im,ipartnum,num)
c
      integer*2 data(4,256,256)
        dimension rx(2,200),ry(2,200),rad1(200)
        dimension xx(2,200),yy(2,200),rrad(2,200),xx1(2,200),yy1(2,200)
      dimension ix(20000),iy(20000),ipart(20000)
      dimension rnumxc(200),rnumyc(200),rden(200),
&               iarea(200),iiarea(200),x(256),y(256)
        dimension ygen(256),xgen(256),m(11),er(11),sig(11),a(11)
      common /imag/ data,ix,iy,ipart
        common /cen/ rx,ry,rad1
        common /ken/ xx,yy,rrad,xx1,yy1
c
      do 31 i=1,200
      rnumxc(i)=0.
      rnumyc(i)=0.
      rden(i)=0.
      xx(k,i)=0.
      yy(k,i)=0.
        iarea(i)=0
31     continue
      do 32 i=1,20000
      ipart(i)=0
      ix(i)=0
      iy(i)=0
32     continue
      do 33 i=1,256
      do 33 j=1,256
      data(3,i,j)=0
      data(4,i,j)=0
33     continue
        do 29 i=1,256
        x(i)=0.
        y(i)=0.
        xgen(i)=0.
        ygen(i)=0.
29      continue
c
```

```
c   Determine if the image needs smoothing or not
c
        call histo(1,x,y,jk,rmax)
        perc=(float(jk)/256.)*100.
        if (perc .gt. 90.) then
        call mean0(y,rmax,mu1,mu2)
        sig1=0.
        sig2=0.
        ithresh=mu1+int(float(mu2-mu1)/2.)
        nu=4
        do 760 i=1,256
        do 760 j=1,256
        data(nu,i,j)=data(1,i,j)
760     continue
        goto 744
        endif
       nu=4
c
c   Smooth the image
c
       do 10 i=2,255
       do 10 j=2,255
       rdata=(1./9.)*(data(im,i-1,j-1)+data(im,i-1,j)+data(im,i-1,j+1)
&       +data(im,i,j-1)+data(im,i,j+1)+data(im,i,j)
&       +data(im,i+1,j-1)+data(im,i+1,j)+data(im,i+1,j+1))
       data(nu,i,j)=ifix(rdata)
10      continue
        do 34 i=1,1
        do 34 j=1,256
        data(nu,i,j)=data(nu,i+1,j)
34      continue
        do 35 i=1,256
        do 35 j=256,256
        data(nu,i,j)=data(nu,i,j-1)
35      continue
        do 36 i=256,256
        do 36 j=1,256
        data(nu,i,j)=data(nu,i-1,j)
36      continue
        do 37 i=1,256
        do 37 j=1,1
        data(nu,i,j)=data(nu,i,j+1)
37      continue
c
c   Find the threshold value
c
        call histo(nu,x,y,jk,rmax)
        call mean1(y,mu1)
        mu=mu1
        do 557 i=1,11
```

```
      m(i)=0
      er(i)=0.
      sig(i)=0.
      a(i)=0.
557   continue
      do 550 i=1,11
      err=0.
      mu1=mu-6+i
      if (mu1 .lt. 1) mu1=1
      call stddev(y,mu1,sig1,a1,err,cc)
      if (cc .gt. 0.) err=1.0E20
      m(i)=mu1
      er(i)=err
      sig(i)=sig1
      a(i)=a1
550   continue
      err=amin1(er(1),er(2),er(3),er(4),er(5),er(6),er(7),er(8),
&     er(9),er(10),er(11))
      do 551 i=1,11
      if (er(i) .eq. err) then
      mu1=m(i)
      sig1=sig(i)
      a1=a(i)
      endif
551   continue
      do 500 i=1,256
      dummy=a1*exp(-(x(i)-mu1)**2/sig1**2)
      y(i)=y(i)-dummy
      if (y(i) .lt. 0) y(i)=0
500   continue
      ilow = mu1+int(1.5*sig1+.5)
      call mean(y,mu2,ilow,256)
      mu=mu2
      do 556 i=1,11
      m(i)=0
      er(i)=0.
      sig(i)=0.
      a(i)=0.
556   continue
      do 552 i=1,11
      err=0.
      mu2=mu-6+i
      if (mu2 .gt. 256) mu2=256
      call stddev(y,mu2,sig2,a2,err,cc)
      if (cc .gt. 0.) err=1.0E35
      m(i)=mu2
      er(i)=err
      sig(i)=sig2
      a(i)=a2
552   continue
```

```
      err=amin1(er(1),er(2),er(3),er(4),er(5),er(6),er(6),er(7),er(8)
&     er(9),er(10),er(11))
      do 553 i=1,11
      if (er(i) .eq. err) then
      mu2=m(i)
      sig2=sig(i)
      a2=a(i)
      endif
553   continue
      do 600 i=mu1,mu2
      p1 = a1*exp(-(i-mu1)**2/sig1**2)
      p2 = a2*exp(-(i-mu2)**2/sig2**2)
      if (p1 .le. p2) goto 650
600   continue
650   ithresh=i
      p=(p1+p2)/2
c
c  Threshold the image.
c
744    iim=0
      do 21 i=1,256
      do 21 j=1,256
      if (data(nu,i,j) .lt. ithresh) then
        if (iq .eq. 1) data(2,i,j)=0
        if (iq .eq. 2) data(1,i,j)=0
      else
      if (iq .eq. 1) data(2,i,j)=250
        if (iq .eq. 2) data(1,i,j)=250
        iim=iim+1
      endif
21     continue
        perc=(float(iim)/256.0**2)*100.
        write(20,559)mu1,sig1,mu2,sig2,ithresh,perc
559     format(4x,i3,1x,f7.4,1x,i3,1x,f7.4,1x,i3,1x,f7.4)
c
c  Identify and locate the particles.
c
      num=0
      l=1
      if (iq .eq. 1) im=2
      if (iq .eq. 2) im=1
      if (data(im,1,1) .ne. 250) goto 30
      ix(1)=1
      iy(1)=1
      ipart(1)=1
      data(3,1,1)=num+1
      l=l+1
      num=num+1
30     continue
      do 42 i=1,1
```

```
      do 42 j=2,256
      if (data(im,i,j) .ne. 250) goto 42
      if (data(im,i,j) .eq. data(im,i,j-1)) then
      ix(l)=i
      iy(l)=j
      ipart(l)=data(3,i,j-1)
      data(3,i,j)=data(3,i,j-1)
      l=l+1
      else
      ix(l)=i
      iy(l)=j
      ipart(l)=num+1
      data(3,i,j)=num+1
      l=l+1
      num=num+1
      endif
42     continue
      do 41 j=1,1
      do 41 i=2,256
      if (data(im,i,j) .ne. 250) goto 41
      if (data(im,i,j) .eq. data(im,i-1,j)) then
      ix(l)=i
      iy(l)=j
      ipart(l)=data(3,i-1,j)
      data(3,i,j)=data(3,i-1,j)
      l=l+1
      else
      ix(l)=i
      iy(l)=j
      ipart(l)=num+1
      data(3,i,j)=num+1
      l=l+1
      num=num+1
      endif
41     continue
      nn=0
      do 40 i=2,256
      do 40 j=2,256
      if (data(im,i,j) .ne. 250) goto 40
      if (data(im,i,j) .ne. data(im,i,j-1) .and.
&      data(im,i,j) .ne. data(im,i-1,j-1) .and.
&      data(im,i,j) .ne. data(im,i-1,j) .and.
&      data(im,i,j) .ne. data(im,i-1,j+1)) then
      ix(l)=i
      iy(l)=j
      ipart(l)=num+1
       data(3,i,j)=num+1
      l=l+1
      num=num+1
       if (num .gt. 200) return
```

```
        if (l .gt. 20000) return
      else
      if (data(im,i,j) .eq. data(im,i,j-1)) then
      ix(l)=i
      iy(l)=j
      ipart(l)=data(3,i,j-1)
      data(3,i,j)=data(3,i,j-1)
      if (data(3,i,j) .ne. data(3,i-1,j) .and. data(3,i-1,j) .ne.
&       0) then
      igreat=data(3,i-1,j)
      ilow=data(3,i,j)
      call compare(i,j,l,ilow,igreat,num)
      endif
      endif
      if (data(im,i,j) .eq. data(im,i-1,j-1)) then
      ix(l)=i
      iy(l)=j
      ipart(l)=data(3,i-1,j-1)
      data(3,i,j)=data(3,i-1,j-1)
      if (data(3,i,j) .ne. data(3,i,j-1) .and. data(3,i,j-1) .ne.
&       0) then
      igreat=data(3,i,j-1)
      ilow=data(3,i,j)
      call compare(i,j,l,ilow,igreat,num)
      endif
      if (data(3,i,j) .ne. data(3,i-1,j) .and. data(3,i-1,j) .ne.
&       0) then
      igreat=data(3,i-1,j)
      ilow=data(3,i,j)
      call compare(i,j,l,ilow,igreat,num)
      endif
      endif
      if (data(im,i,j) .eq. data(im,i-1,j)) then
      ix(l)=i
      iy(l)=j
      ipart(l)=data(3,i-1,j)
      data(3,i,j)=data(3,i-1,j)
      if (data(3,i,j) .ne. data(3,i,j-1) .and. data(3,i,j-1) .ne.
&       0) then
      igreat=data(3,i,j-1)
      ilow=data(3,i,j)
      call compare(i,j,l,ilow,igreat,num)
      endif
      endif
      if (data(im,i,j) .eq. data(im,i-1,j+1)) then
      ix(l)=i
      iy(l)=j
      ipart(l)=data(3,i-1,j+1)
      data(3,i,j)=data(3,i-1,j+1)
      if (data(3,i,j) .ne. data(3,i,j-1) .and. data(3,i,j-1) .ne.
```

```
&       0) then
        igreat=data(3,i,j-1)
        ilow=data(3,i,j)
        call compare(i,j,l,ilow,igreat,num)
        endif
        if (data(3,i,j) .ne. data(3,i-1,j) .and. data(3,i-1,j) .ne.
&       0) then
        igreat=data(3,i-1,j)
        ilow=data(3,i,j)
        call compare(i,j,l,ilow,igreat,num)
        endif
        endif
        l=l+1
          if (l .gt. 20000) return
        endif
40       continue
          l=l-1
          im=1
          if (num .gt. 250) return
        do 70 ik=1,l
          in=ipart(ik)
        rnumxc(in)=rnumxc(in)+float(ix(ik)*data(nu,ix(ik),iy(ik)))
        rnumyc(in)=rnumyc(in)+float(iy(ik)*data(nu,ix(ik),iy(ik)))
        rden(in)=rden(in)+float(data(nu,ix(ik),iy(ik)))
          iarea(in)=iarea(in)+1
70       continue
          do 69 n=1,num
          xx(k,n)=rnumxc(n)/rden(n)
          if (xx(k,n) .gt. 256.) xx(k,n)=255.8
          if (xx(k,n) .lt. 1.) xx(k,n)=.8
          yy(k,n)=rnumyc(n)/rden(n)
          if (yy(k,n) .gt. 256.) yy(k,n)=255.8
          if (yy(k,n) .lt. 1.) yy(k,n)=0.8
          rrad(k,n)=sqrt(float(iarea(n))/3.14159)
69       continue
        return
        end
c
c  ------------COMPARE PARTICLE NUMBERS------------
c
        subroutine compare(i,j,l,ilow,igreat,num)
c
        integer*2 data(4,256,256)
        dimension ipart(20000),ix(20000),iy(20000)
        common /imag/ data,ix,iy,ipart
c
        if (ilow .gt. igreat) then
        do 64 k=1,l
        if (ipart(k) .eq. ilow) then
        ipart(k)=igreat
```

```
        else if (ipart(k) .gt. ilow) then
        ivar=ipart(k)-ilow
        ipart(k)=ilow+(ivar-1)
        endif
64      continue
      do 66 k=1,i-1
      do 66 n=1,256
      if (data(3,k,n) .eq. ilow) then
      data(3,k,n)=igreat
        else if (data(3,k,n) .gt. ilow) then
        ivar=data(3,k,n)-ilow
        data(3,k,n)=ilow+(ivar-1)
        endif
66      continue
      do 68 k=i,i
      do 68 n=1,j
      if (data(3,k,n) .eq. ilow) then
      data(3,k,n)=igreat
        else if (data(3,k,n) .gt. ilow) then
        ivar=data(3,k,n)-ilow
        data(3,k,n)=ilow+(ivar-1)
        endif
68      continue
        ilow=igreat
        num=num-1
      else
      do 65 k=1,l
      if (ipart(k) .eq. igreat) then
      ipart(k)=ilow
        else if (ipart(k) .gt. igreat) then
        ivar=ipart(k)-igreat
        ipart(k)=igreat+(ivar-1)
        endif
65      continue
      do 67 k=1,i-1
      do 67 n=1,256
      if (data(3,k,n) .eq. igreat) then
      data(3,k,n)=ilow
        else if (data(3,k,n) .gt. igreat) then
        ivar=data(3,k,n)-igreat
        data(3,k,n)=igreat+(ivar-1)
        endif
67      continue
      do 69 k=i,i
      do 69 n=1,j
      if (data(3,k,n) .eq. igreat) then
      data(3,k,n)=ilow
        else if (data(3,k,n) .gt. igreat) then
        ivar=data(3,k,n)-igreat
        data(3,k,n)=igreat+(ivar-1)
```

```
        endif
69      continue
        igreat=ilow
        num=num-1
       endif
       return
       end
c
c  ----------CALCULATE THE HISTOGRAM OF THE IMAGE----------
c
       subroutine histo(im,x,y,kk,rmax)
c
       integer*2 data(4,256,256)
       dimension y(256),x(256)
         dimension ix(20000),iy(20000),ipart(20000)
       common /imag/ data,ix,iy,ipart
       data n /256/
c
        kk=0
        rmax=0.
       do 10 i=1,256
       do 20 j=1,256
       if (im .eq. 1) ival=data(im,i,j)
       if (im .eq. 2) ival=data(im,i,j)
       if (im .eq. 3) ival=data(im,i,j)
       if (im .eq. 4) ival=data(im,i,j)
       y(ival+1)=y(ival+1)+1
20      continue
10      continue
       do 30 i=1,n
       x(i)=i-1
         if (y(i) .eq. 0.) kk=kk+1
         rmax=amax1(rmax,y(i))
30      continue
       return
       end
c
c  -------CALCULATE BOTH MEANS WHEN SMOOTHING IS NOT REQUIRED-------
c
        subroutine mean0(y,rmax,mu1,mu2)
        dimension y(256)
c
        mu1=256
        rmax2=0.
        do 100 i=1,256
        if (y(i) .eq. rmax) mu1=i-1
        if (i .gt. mu1+10) rmax2=amax1(rmax2,y(i))
100     continue
        do 101 i=mu1+10,256
        if (y(i) .eq. rmax2) mu2=i-1
```

```
101     continue
        return
        end
c
c  ----------CALCULATE THE FIRST MEAN OF THE SMOOTHED IMAGE----------
c
        subroutine mean1(y,mu)
        dimension y(256)
c
        rmu=0.
        do 100 i=2,256
        rmu=amax1(rmu,y(i))
100     continue
        do 101 i=2,256
        if (y(i) .eq. rmu) then
        mu=i
        goto 200
        endif
101     continue
200     return
        end
c
c  ----------CALCULATE THE SECOND MEAN OF THE SMOOTHED IMAGE----------
c
        subroutine mean(y,mu,il,ih)
        dimension y(256),yy(256),ys(256)
c
        iter=0
        do 100 i=il,ih
        yy(i)=y(i)
100     continue
150     sum=0.
        do 200 i=il+1,ih-1
        ys(i)=(yy(i+1)+yy(i)+yy(i-1))/3.
        sum=sum+(ys(i)-yy(i))**2
200     continue
        ys(256)=(yy(256)+yy(255))/2.
        sum=sum+(ys(256)-yy(256))**2
        sum=sqrt(sum/(float(ih-il-2)))
        do 250 i=il,ih
        yy(i)=ys(i)-sum
        if (yy(i) .lt. 0.) yy(i)=0.
250     continue
280     ndet=0
        do 300 i=il+1,ih-1
        if (yy(i) .gt. yy(i-1) .and. yy(i) .gt. yy(i+1)) then
        ndet=ndet+1
        mu=i
        endif
300     continue
```

```
      if (ndet .gt. 2) goto 150
      if (ndet .lt. 1) then
      if (iter .gt. 10) then
      mu=il
      return
      endif
      sum=sum/2.
      iter=iter+1
      do 350 i=il,ih
      yy(i)=ys(i)-sum
350   continue
      goto 280
      endif
      return
      end
c
c  ------------CALCULATE THE STANDARD DEVIATION------------
c
      subroutine stddev(y,mu,sig,a,err,cc)
      dimension y(256)
c
      sx=0.
      sxx=0.
      sy=0.
      sxy=0.
      np=0
      i1=mu-10
      i2=mu+10
      if (i1 .lt. 1 ) i1=1
      if (i2 .gt. 256) i2=256
      do 100 i=1,256
      if (y(i) .eq. 0.) goto 100
      if (i .lt. i1 .or. i .gt. i2) goto 100
      sx=sx+(i-mu)**2
      sxx=sxx+((i-mu)**2)**2
      sxy=sxy+alog(y(i))*(i-mu)**2
      sy=sy+alog(y(i))
      np=np+1
100   continue
      den=sx**2-sxx*np
      cc=(sy*sx-sxy*np)/den
      if (cc .gt. 0.) return
      rlna=(sx*sxy-sxx*sy)/den
      sig=sqrt(-(1./cc))
      a=exp(rlna)
      do 200 i=i1,i2
      var1=a*exp((-(i-mu)**2)/sig**2)
      var=(y(i)-var1)**2
      err=err+var
200   continue
```

```
      return
      end
c
c  ----------CALCULATE THE VELOCITY FIELD----------
c
      subroutine velo(np,vel,delt,ar)
c
      dimension xx(5,200),yy(5,200),rrad(200),u(200),v(200)
      dimension xx1(5,200),yy1(5,200)
      common /ken/ xx,yy,rrad,xx1,yy1
      common /men/ u,v
c
      do 99 i=1,np
      u(i)=0.
      v(i)=0.
99    continue
      do 102 i=1,np
      cx=xx(1,i)-128.
      cy=yy(1,i)-128.
c
c  Rankine vortex flow field
c
      r=sqrt(cx**2+cy**2)
      theta=atan2(cy,cx)
      if (r .le. ar) then
      vth=vel*(r/ar)
      else
      vth=vel*(ar/r)
      end if
      u(i)=-vth*sin(theta)
      v(i)=vth*cos(theta)
c
c  Stagnation point flow field
c
      u(i)=vel*cx/128.
      v(i)=-vel*cy/128.
102   continue
      return
      end
c
c  ----------APPLY THE VELOCITY FIELD----------
c
      subroutine velocity(n,np,vel,delt,ar)
c
      dimension rx(5,200),ry(5,200),rad1(200)
      dimension xd(200),yd(200),ud(200),vd(200)
      common /cen/ rx,ry,rad1
c
      ig=0
      do 99 i=1,np
```

```
      xd(i)=0.
      ud(i)=0.
      ud(i)=0.
      vd(i)=0.
99    continue
      do 100 i=1,np
      cx=rx(n-1,i)-128.
      cy=ry(n-1,i)-128.
c
c  Rankine vortex flow field
c
      r=sqrt(cx**2+cy**2)
      theta=atan2(cy,cx)
      if (r .le. ar) then
      vth=vel*(r/ar)
      else
      vth=vel*(ar/r)
      end if
      u=-vth*sin(theta)
      v=vth*cos(theta)
c
c  Stagnation point flow field
c
      u=vel*cx/128.
      v=-vel*cy/128.
      rx(n,i)=rx(n-1,i)+u*delt
      ry(n,i)=ry(n-1,i)+v*delt
      xd(i)=rx(n-1,i)
      ud(i)=u
      yd(i)=ry(n-1,i)
      vd(i)=v
100   continue
c
c  Disregard the particles which move out of the frame
c
      do 101 i=1,np
      if (rx(n,i).lt.0. .or. rx(n,i).gt.256. .or. ry(n,i).lt.0.
&     .or. ry(n,i).gt.256.) then
      ig=ig+1
      do 200 k=i,np
      rx(n,k)=rx(n,k+1)
      ry(n,k)=ry(n,k+1)
      rx(n-1,k)=rx(n-1,k+1)
      ry(n-1,k)=ry(n-1,k+1)
      rad1(k)=rad1(k+1)
      xd(k)=xd(k+1)
      yd(k)=yd(k+1)
      ud(k)=ud(k+1)
      vd(k)=vd(k+1)
200   continue
```

```
      i=i-1
      end if
101   continue
      np=np-ig
      return
      end
c
c  ----------GENERATE AN IMAGE IN TIME FRAME B----------
c
      subroutine compim(k,idum,im,index,ipartnum,radlev,amplrad
&     ,partlev,bkgndlev,ampl)
c
      integer*2 data(4,256,256)
      dimension rx(5,200),ry(5,200),iiarea(200)
      dimension ix(20000),iy(20000),ipart(20000),rad1(200)
     common /imag/ data,ix,iy,ipart
      common /cen/ rx,ry,rad1
c
      do 6 i=1,256
      do 6 j=1,256
      var=bkgndlev+ampl*gasdev(idum)
      if (var .lt. 0.) then
      var=0.
      else if (var .gt. 255.) then
      var=255.
      end if
      data(im,i,j)=int(var)
6     continue
      do 13 n=1,ipartnum
      dummy1=rx(k,n)-rad1(n)
      dummy2=rx(k,n)+rad1(n)
      dummy3=ry(k,n)-rad1(n)
      dummy4=ry(k,n)+rad1(n)
      if (dummy1 .lt. 0.) dummy1=1.
      if (dummy3 .lt. 0.) dummy3=1.
      if (dummy2 .gt. 256.) dummy2=256.
      if (dummy4 .gt. 256.) dummy4=256.
      do 11 i=int(dummy1),int(dummy2)
      do 11 j=int(dummy3),int(dummy4)
      x=abs(float(i)-rx(k,n))
      y=abs(float(j)-ry(k,n))
      argu=x**2+y**2
      radi=sqrt(argu)
      if (radi .le. rad1(n)) then
      data(im,i,j)=int(partlev+ampl*gasdev(idum))
      if (data(im,i,j) .gt. 255) then
      data(im,i,j)=255
      else if (data(im,i,j) .lt. 0) then
      data(im,i,j)=0
      end if
```

```
        end if
11      continue
13      continue
        return
        end
c
c  ------------MATCH THE PARTICLES BETWEEN FRAMES------------
c
        subroutine test(io,iu,num1,num2,delt,vel,ar,im,num)
c
      integer*2 data(4,256,256)
        dimension xx(5,200),yy(5,200),rrad(5,200),xx1(5,200),yy1(5,200)
      dimension ix(20000),iy(20000),ipart(20000)
        dimension iix(400),iiy(400),iiarea(200),vx(1,200),u(200),v(200)
        dimension iarea(200),x(256),y(256),in(400),vy(1,200)
        dimension radius(400),radius1(400),xc(256),yc(256),ii(400)
        dimension xi(256),yj(256),xk(256),yl(256),ik(256,256),
&               xd(200),yd(200),ud(200),vd(200)
      common /imag/ data,ix,iy,ipart
        common /ken/ xx,yy,rrad,xx1,yy1
        common /men/ u,v
c
        im=1
        if (num .gt. 250) goto 922
        do 399 i=1,400
        in(i)=0
        iix(i)=0
        iiy(i)=0
        radius(i)=0.
        radius1(i)=0.
        ii(i)=0
399     continue
        do 401 i=1,256
        xk(i)=0.
        yl(i)=0.
        xi(i)=0.
        yj(i)=0.
        xc(i)=0.
        yc(i)=0.
401     continue
        do 80 i=1,256
        do 80 j=1,256
        data(2,i,j)=0
        data(3,i,j)=0
        data(4,i,j)=0
        ik(i,j)=0
80      continue
        do 73 n=1,num1
        m = int(xx1(io,n)+.5)
        l = int(yy1(io,n)+.5)
```

```
      xk(n)=xx1(io,n)
      yl(n)=yy1(io,n)
      xc(n)=xx(io,n)
      yc(n)=yy(io,n)
      data(4,m,l)=2
      data(2,m,l)=n*16+int(rrad(io,n)+.5)
73    continue
      do 74 n=1,num2
      i = int(xx(iu,n)+.5)
      j = int(yy(iu,n)+.5)
      xi(n)=xx(iu,n)
      yj(n)=yy(iu,n)
      data(4,i,j)=data(4,i,j)+1
      data(3,i,j)=n*16+int(rrad(iu,n)+.5)
74    continue
      kw=0
      ipf=0
      sum=0.
      sumsq=0.
      do 77 i=1,256
      do 77 j=1,256
      if (data(4,i,j) .ne. 0) then
      kw=kw+1
      if (data(4,i,j) .eq. 1) then
      n1=data(3,i,j)/16
      radius(kw)=data(3,i,j)-n1*16
      ik(i,j)=kw
      ii(kw)=n1
      else if (data(4,i,j) .eq. 2) then
      n2=data(2,i,j)/16
      radius(kw)=data(2,i,j)-n2*16
      ik(i,j)=kw
      ii(kw)=n2
      else if (data(4,i,j) .eq. 3) then
      n1=data(3,i,j)/16
      n2=data(2,i,j)/16
      ii(kw)=0
      ik(i,j)=kw
      vx(1,ipf)=(xi(n1)-xc(n2))/delt
      vy(1,ipf)=(yj(n1)-yc(n2))/delt
      er=(sqrt(u(n2)**2+v(n2)**2)-sqrt(vx(1,ipf)**2+vy(1,ipf)**2))**2
      sumsq=sumsq+er
      sum=sum+sqrt(er)
      xd(ipf)=xc(n2)
      yd(ipf)=yc(n2)
      ud(ipf)=vx(1,ipf)
      vd(ipf)=vy(1,ipf)
      radius(kw)=0.
      ipf=ipf+1
      end if
```

```
      in(kw)=data(4,i,j)
      iix(kw)=i
      iiy(kw)=j
      end if
77    continue
      do 75 i=1,kw
      if (in(i) .eq. 3) in(i)=13
75    continue
c
      k1=1
      k2=1
      perc1=(float(ipf)/float(num1))*100.
      if (perc1 .lt. 98.) then
311   l=0
      do 297 kk=k1,k2
      rr=.1*kk
      do 298 i=1,kw
      if (in(i) .eq. 2) then
      ivar=int(rr*radius(i)+.5)
      idummy1=iix(i)-ivar
      if (idummy1 .lt. 1) idummy1=1
      idummy2=iix(i)+ivar
      if (idummy2 .gt. 256) idummy2=256
      idummy3=iiy(i)-ivar
      if (idummy3 .lt. 1) idummy3=1
      idummy4=iiy(i)+ivar
      if (idummy4 .gt. 256) idummy4=256
      do 305 n=idummy1,idummy2
      do 305 m=idummy3,idummy4
      if (data(4,n,m) .eq. 1 .and. l .le. 1) then
      n1=data(3,n,m)/16
      l=l+1
      in(ik(n,m))=11
      ipf=ipf+1
      vx(1,ipf)=(xi(n1)-xc(ii(i)))/delt
      vy(1,ipf)=(yj(n1)-yc(ii(i)))/delt
      er=(sqrt(u(ii(i))**2+v(ii(i))**2)-sqrt(vx(1,ipf)**2+
&     vy(1,ipf)**2))**2
      sumsq=sumsq+er
      sum=sum+sqrt(er)
      xd(ipf)=xc(ii(i))
      yd(ipf)=yc(ii(i))
      ud(ipf)=vx(1,ipf)
      vd(ipf)=vy(1,ipf)
      end if
305   continue
      if (l .ne. 0) in(i)=12
      l=0
      end if
298   continue
```

```
297     continue
        perc1=(float(ipf)/float(num1))*100.
        if (perc1 .gt. 98.) goto 310
        if (rr .gt. 15.) goto 310
        k1=k1+1
        k2=k2+1
        goto 311
        end if
310     km=0
        nm=0
        do 70 i=1,kw
        if (in(i) .eq. 1) km=km+1
        if (in(i) .eq. 2) nm=nm+1
70      continue
        rmean=sum/float(ipf)
        du=(sumsq/(float(ipf)))-rmean**2
        if (du .lt. 0) du=0.
        rsigm=sqrt(du)
        goto 923
922     ipf=1
        km=1
        nm=1
        rmean=1.0
        rsigm=1.0
923     write(20,175)ipf,km,nm,rmean,rsigm,vel,ar
175     format(4x,i3,1x,i3,1x,i3,1x,f7.4,1x,f7.4,1x,f5.2,1x,f5.2)
        return
        end
c
c   ------COMPUTE THE AVERAGE VELOCITY BY CROSS-CORRELATION-----
c
        subroutine velavg(np,delt,vx,vy)
c
        dimension x1(32,32),x2(32,32)
        dimension cor(64,64)
        common /vlc/x1,x2
c
        call xcor2d(k,cor,np)
c
c -- Locate maximum cross-correlation
c
        cormax=0
        n=2*np
        do 200 i=1,n
        do 100 j=1,n
        if (cor(i,j) .gt. cormax) then
        cormax=cor(i,j)
        imax=i
        jmax=j
        endif
```

```
100     continue
200     continue
c
c -- Evaluate velocity
c
        xdisp=1-imax
        if (imax .gt. np) xdisp=n-imax+1
        ydisp=1-jmax
        if (jmax .gt. np) ydisp=n-jmax+1
        vx=xdisp/delt
        vy=ydisp/delt
        return
        end
c
c  ------------COMPUTE THE CROSS CORRELATION------------
c
        subroutine xcor2d(k,cor,np)
c
        dimension x1(32,32),x2(32,32)
        dimension cor(64,64)
        dimension data1(64,64),data2(64,64)
        complex data1t(64,64),data2t(64,64)
        common /vlc/x1,x2
c
        n=2*np
        do 30 i=1,n
        do 20 j=1,n
        data1(i,j)=0.
        data2(i,j)=0.
20      continue
30      continue
        do 100 i=1,np
        do 50 j=1,np
        data1(i,j)=x1(i,j)
        data2(i,j)=x2(i,j)
50      continue
100     continue
        call fft2d(data1,data1t,n)
        call fft2d(data2,data2t,n)
        do 400 i=1,n
        do 350 j=1,n
        data1t(i,j)=data1t(i,j)*conjg(data2t(i,j))
350     continue
400     continue
        call ifft2d(data1t,cor,n)
        return
        end
c
c  ------------COMPUTE THE 2-D FOURIER TRANSFORM------------
c
```

```
      subroutine fft2d(x,xt,n)
c
      dimension x(64,64)
      complex xt(64,64),temp(64)
c
      np=n/2
      do 300 i=1,np
      do 100 j=1,n
      temp(j)=cmplx(x(i,j),0.)
100   continue
      call fft(temp,n,0)
      do 200 j=1,n
      xt(i,j)=temp(j)
      xt(i+np,j)=0
200   continue
300   continue
      do 600 j=1,n
      do 400 i=1,n
      temp(i)=xt(i,j)
400   continue
      call fft(temp,n,0)
      do 500 i=1,n
      xt(i,j)=temp(i)
500   continue
600   continue
      return
      end
c
c  ----------COMPUTE THE INVERSE 2-D FOURIER TRANSFORM----------
c
      subroutine ifft2d(xt,x,n)
c
      dimension x(64,64)
      complex xt(64,64),temp(64)
c
      do 300 j=1,n
      do 100 i=1,n
      temp(i)=xt(i,j)
100   continue
      call fft(temp,n,1)
      do 200 i=1,n
      xt(i,j)=temp(i)
200   continue
300   continue
      do 600 i=1,n
      do 400 j=1,n
      temp(j)=xt(i,j)
400   continue
      call fft(temp,n,1)
      do 500 j=1,n
```

```
      x(i,j)=real(temp(j))
500   continue
600   continue
      return
      end
c
c  --INTERPOLATE BETWEEN AVERAGE VELOCITIES FOR THE VELOCITY IN X DIR--
c
      subroutine interp1(vi,x,y,f)
      dimension vi(4,4)
c
      ix1=int((x-32)/64.)+1
      if (ix1 .gt. 3) ix1=3
      ix2 = ix1+1
      iy1=int((y-32)/64.)+1
      if (iy1 .gt. 3) iy1=3
      iy2 = iy1+1
      x1=(ix1-1)*64.+32.
      x2=(ix2-1)*64.+32.
      y1=(iy1-1)*64.+32.
      y2=(iy2-1)*64.+32.
      dx=float(x2-x1)
      dy=float(y2-y1)
      fa = -(x-x2)/dx*vi(ix1,iy1)+(x-x1)/dx*vi(ix2,iy1)
      fb = -(x-x2)/dx*vi(ix1,iy2)+(x-x1)/dx*vi(ix2,iy2)
      f = -(y-y2)/dy*fa+(y-y1)/dy*fb
      return
      end
c
c  --INTERPOLATE BETWEEN AVERAGE VELOCITIES FOR THE VELOCITY IN Y DIR--
c
      subroutine interp2(vj,x,y,f)
      dimension vj(4,4)
c
      ix1=int((x-32)/64.)+1
      if (ix1 .gt. 3) ix1=3
      ix2 = ix1+1
      iy1=int((y-32)/64.)+1
      if (iy1 .gt. 3) iy1=3
      iy2 = iy1+1
      x1=(ix1-1)*64.+32.
      x2=(ix2-1)*64.+32.
      y1=(iy1-1)*64.+32.
      y2=(iy2-1)*64.+32.
      dx=float(x2-x1)
      dy=float(y2-y1)
      fa = -(x-x2)/dx*vj(ix1,iy1)+(x-x1)/dx*vj(ix2,iy1)
      fb = -(x-x2)/dx*vj(ix1,iy2)+(x-x1)/dx*vj(ix2,iy2)
      f = -(y-y2)/dy*fa+(y-y1)/dy*fb
      return
```

Wang-Chul Choi

```
      end
c
c  ----------CALCULATE RAMDOMLY DISTRIBUTED DEVIATES----------
c
      function ran1(idum)
c
      dimension r(97)
      parameter (m1=259200,ia1=7141,ic1=54773,rm1=1./m1)
      parameter (m2=134456,ia2=8121,ic2=28411,rm2=1./m2)
      parameter (m3=423000,ia3=4561,ic3=51349)
      data iff/0/
c
      if (idum .lt. 0 .or. iff .eq. 0) then
      iff=1
      ix1=mod(ic1-idum,m1)
      ix1=mod(ia1*ix1+ic1,m1)
      ix2=mod(ix1,m2)
      ix1=mod(ia1*ix1+ic1,m1)
      ix3=mod(ix1,m3)
      do 200 j=1,97
      ix1=mod(ia1*ix1+ic1,m1)
      ix2=mod(ia2*ix2+ic2,m2)
      r(j)=(float(ix1)+float(ix2)*rm2)*rm1
200   continue
      idum=1
      end if
      ix1=mod(ia1*ix1+ic1,m1)
      ix2=mod(ia2*ix2+ic2,m2)
      ix3=mod(ia3*ix3+ic3,m3)
      j=1+(97*ix3)/m3
      if (j .gt. 97 .or. j .lt. 1) pause
      ran1=r(j)
      r(j)=(float(ix1)+float(ix2)*rm2)*rm1
      return
      end
c
c  ----------CALCULATE NORMALLY DISTRIBUTED DEVIATES----------
c
      function gasdev(idum)
      data iset/0/
c
      if (iset .eq. 0) then
1     v1=2.*ran1(idum)-1
      v2=2.*ran1(idum)-1
      r=v1**2+v2**2
      if (r .ge. 1.) goto 1
      fac=sqrt(-2.*log(r)/r)
      gset=v1*fac
      gasdev=v2*fac
      iset=1
```

```
else
gasdev=gset
iset=0
end if
return
end
```

LIST OF REFERENCES

Adrian, R. J., 1984, "Scattering Particle Characteristics and Their Effect on Pulsed Laser Measurements of Fluid Flow: Speckle Velocimetry vs. Particle Image Velocimetry", *Applied Optics*, Vol 23, No 11, pp. 1690-1691.

Adrian, R, J., 1986, "Image Sifting Technique to Resolve Directional Ambiguity in Double Pulsed Velocimetry", *Applied Optics*, Vol 25, No 21, pp. 3855-3858.

Adrian, R. J., 1986, "Multi-Point Optical Measurements of Simultaneous Vectors in Unsteady Flows - A Review", *I. J. of Heat and Fluid Flow*, Vol. 7, No. 2, pp. 127-145.

Barford, N. C., 1985, *Experimental Measurements: Precision, Error and Truth, 2nd ed.*, John Wiley & Sons.

Bradshaw, P., 1971, *An Introduction to Turbulence and Its Measurement*, Pergamon Press LTD.

Brodkey, R. S., 1967, *The Phenomena of Fluid Motions, 3rd ed.*, Addison Wesley.

Brodkey, R. S., 1977, "Stereoscopic Visual Studies of Complex Turbulence Shear Flow", *Proceedings of the International Symposium on Flow Visualization, Tokyo, Japan.*, pp. 45-49.

Brodkey, R. S., 1986, "Image Processing and Analysis for Turbulence Research", *Chem. Eng. Educ.*, Vol. 20, pp. 202-207.

Brodkey, R. S. and Guezennec, Y. G., 1988, Report on the NSF Imaging Workshop, The Ohio

State University.

Brown, G. L. and Roshko A., 1974, "On Density Effects and Large Structure in Turbulent Mixing Layers", *J. of Fluid Mechanics*, Vol. 64, pp. 775-816.

Chang, T. P. K., Watson, A. T. and Tatterson, G. B., 1985, "Image Processing of Tracer Particle Motions as Applied to Mixing and Turbulent Flow - I. The Technique", *Chemical Engineering Science*, Vol. 40, No. 2, pp. 269-275.

Corino, E. R. and Brodkey, R. S., 1969, "A Visual Investigation of the Wall Region in Turbulent Flows", *J. of Fluid Mechanics*, Vol. 37, pp.1-30.

Dimotakis, P. E., Debussy, F. D. and Koochesfahani, M. M., 1981, "Particle Streak Velocity Field Measuremets in a Three-Dimensional Mixing Layer", *Physics of Fluids*, Vol. 24, p. 995.

Drain, L. E., 1976, *The Laser Doppler Techniques*, John Wiley & Sons.

Economikos, L., 1988, "Image Processing and Analysis of Colored Particle Motion in Turbulent Flow", *Ph.D Thesis, The Ohio State University.*

Erf, R. K., 1980, "Application of Laser Speckle to Measurement", *Laser Applications*, ed J. W. Goodman and M. Ross, Vol. 4, Academic Press.

Faugeras, O. D., 1983, *Fundamentals in Computer Vision*, Cambrige University Press.

Frost, W. and Mouldin T., 1977, *Handbook of Turbulence, Volume 1*, Plenum Press, New York and London.

Goldstein, R. J., 1983, *Fluid Mechanics Measurements*, Hemisphere Publishing Corporation.

Head, F. R. and Bandyophadyay, P., 1981, "New Aspects of Turbulent Boundary Layer Structure", *J. of Fluid Mechanics*, Vol 107, pp. 297-337.

Hinze, J. O., 1975, *Turbulence, 2nd ed.*, McGraw-Hill, New-York.

Horn, B. K. P., 1986, *Robot Vision*, MIT Press.

Kim, J. and Moin, P., 1986, "The Structure of the Velocity Field in Turbulent Channel Flow, Part 2. Study of Ensemble-Averaged Field", *J. of Fluid Mechanics*, Vol 162, pp. 339-363.

Kobaybashi, T., Ishihara, T. and Sasaki, N., 1983, "Automatic Analysis of Photographs of Trace Particles by Microcomputer System", *Third International Conference on Flow Visualization, Ann Arbor, USA*, pp 261-265.

Kobaybashi, T. and Yoshitake, Y., 1985, "An Automated Analysis Method for Determining Velocity Vectors from Pathline Photograph", *Proceedings of the International Symposium on Fluid Control Measurement, Tokyo, Japan,*, Pergamon Press, pp. 729-734.

Lakshmanan, K., 1986, "Computer Image Processing of Color Particles Markers in Flow Visualization", *Ph.D Thesis, The Ohio State University.*

Landreth, C. C., Adrian, R. J. and Yao C. S., 1987, "Double Pulsed Particle Image Velocimeter with Directional Resolution for Complex Flows", *Experiments in Fluids.*, Vol. 5, p. 105.

Levine, M. D., 1985, *Vision in Man and Mashine*, McGraw-Hill, New York.

Lourenco, L. M., 1986, "Theory and Applications of Particle Image Displacement Velocimetry", *Lecture Series 1986-09, von Karman Institute for Fluid Dynamics, Belgium, June 9-13-1986.*

Lourenco, L. M. and Krothapalli, A., 1987, "The Role of Photographic Parameters in Laser Speckle or Particle Image Displacement Velocimetry", *Experiments in Fluids*, Vol. 5, pp 29-32.

Lourenco, L. M. and Whiffen, M. C., 1984, "Laser Speckle Methods in Fluid Dynamics Applications", *Proceedings of the International Symposium on Applications of Laser Anemometry to Fluid Mechanics, Lisbon, Portugal.*

Majumdar, P. S., Fales, J. C., Algazi, R. V. and Stewart, J. E., 1987, "Teaching Computers to Track Particles. II. Auto Tracking Algorithm and Coordinate Transformation", *Journal of Imaging Science*, Vol 31, No 5, pp. 208-219.

Miller, I., 1977, *Probability and Statistics for Engineers*, Prendice-Hall, Englewood Cliffs, NJ.

Nishino, K., Kasagi, N. and Hirata, M., 1985, "Flow Visualization and Its Digital Image Processing in a Fully Developed Two Dimentional Turbulent Channel Flow", *Proceedings of the International Symposium on Fluid Control Measurement, Tokyo, Japan,*, Pergamon Press.

Nychas, S. G., Hershey, H. C. and Brodkey, R. S., 1973, "A visual study of turbulent shear flow", *J. of Fluid Mechanics*, Vol. 61, pp. 513-540.

Offen, G. R. and Kline, S. J., 1974, "Combined Dye-Streak and Hydrogen-Bubble Visual Observations of a Turbulent Boundary Layer", *J. of Fluid Mechanics*, Vol. 62, pp. 223-239.

Perry, A. E., 1982, *Hot Wire Anemometry*, Clarendon Press, Oxford.

Praturi, A. K. and Brodkey, R. S., 1978, "A Stereoscopic Visual Study of Coherent Structures in Turbulent Shear Flow", *J. of Fluid Mechanics*, Vol. 89, pp. 251-272.

Press, W. H., Flannery, P.B., Teukolsky, A. S. and Vetterling, T. W., 1986, *Numerical Reci-*

pies, Cambrige University Press.

Russ, K. M., 1988, "Particle Color Considerations in Flow System Image Acquisition", *M.S Thesis, The Ohio State University.*

Sheu, Y. -H. E., Chang, T. P. K., Tatterson, G. B. and Dickey, D. S., 1982, "A Three-Dimensional Measurement Technique for Turbulent Flows", *Chem. Eng. Commun.*, Vol. 17, pp. 67-83.

Tennekes, H. and Lumley J. L., 1972, *A First Course in Turbulence*, The MIT Press.