Experiments in Fluids (2006) 40: 301–313
DOI 10.1007/s00348-005-0068-7

RESEARCH ARTICLE

Nicholas T. Ouellette · Haitao Xu
Eberhard Bodenschatz

A quantitative study of three-dimensional Lagrangian particle tracking algorithms

Received: 17 December 2004 / Revised: 11 October 2005 / Accepted: 11 October 2005 / Published online: 15 November 2005

Abstract A neural network particle finding algorithm and a new four-frame predictive tracking algorithm are proposed for three-dimensional Lagrangian particle tracking (LPT). A quantitative comparison of these and other algorithms commonly used in three-dimensional LPT is presented. Weighted averaging, one-dimensional and two-dimensional Gaussian fitting, and the neural network scheme are considered for determining particle centers in digital camera images. When the signal to noise ratio is high, the one-dimensional Gaussian estimation scheme is shown to achieve a good combination of accuracy and efficiency, while the neural network approach provides greater accuracy when the images are noisy. The effect of camera placement on both the yield and accuracy of three-dimensional particle positions is investigated, and it is shown that at least one camera must be positioned at a large angle with respect to the other cameras to minimize errors. Finally, the problem of tracking particles in time is studied. The nearest neighbor algorithm is compared with a three-frame predictive algorithm and two four-frame algorithms. These four algorithms are applied to particle tracks generated by direct numerical simulation both with and without a method to resolve tracking conflicts. The new four-frame predictive algorithm with no conflict resolution is shown to give the best performance. Finally, the best algorithms are verified to work in a real experimental environment.

N. T. Ouellette (✉) · H. Xu · E. Bodenschatz
Laboratory of Atomic and Solid State Physics, Clark Hall, Cornell University, Ithaca, NY, USA
E-mail: nto2@cornell.edu

E. Bodenschatz
Max Planck Institute for Dynamics and Self-Organization, Göttingen, Germany

1 Introduction

Over the past decade, Lagrangian particle tracking (LPT) has become widely used in experimental fluid dynamics. In an LPT experiment, the flow of interest is seeded with tracer particles that are then imaged to reconstruct the fluid motion. While the related Eulerian measurement technique of particle image velocimetry (PIV) commonly uses a very high seeding density and calculates average velocity vectors for clusters of particles based on the assumption that nearby particles move similarly (Adrian 1991), LPT uses a lower seeding density but finds individual, longer particle tracks that may be used to calculate both Eulerian and Lagrangian quantities. While PIV systems generally measure two-dimensional velocities, multiple cameras are often used in LPT systems, enabling the stereoscopic reconstruction of particle tracks in three dimensions (3D) (Maas et al. 1993; Malik et al. 1993). Such particle tracks may be used to calculate flow properties, including the Lagrangian statistics of turbulent flows (La Porta et al. 2001; Voth et al. 2002). Lagrangian data is of fundamental importance for understanding turbulent mixing and scalar dispersion as well as for setting the parameters of stochastic models of turbulence (Yeung 2002).

The construction of tracks in three-dimensional LPT may be separated into three tasks, which we will analyze individually. Each task is computationally challenging, and no optimal algorithms have yet been found. We present here a quantitative study under a wide range of conditions of many of the commonly used algorithms for three-dimensional LPT systems. We compare these algorithms with newly developed methods.

The first task of an LPT system is to process the images from the cameras to yield the positions of the tracer particles in the image space of each camera. Several algorithms for finding particles in images are explored in Sects. 2 and 3, and we introduce a neural network scheme that is shown to outperform the other algorithms in situations where the signal to noise ratio is poor. After finding the two-dimensional particle positions, the second task is to reconstruct the three-dimensional coordinates, since each camera images only a two-dimensional projection of the measurement volume. The effects of camera placement on the stereo-matching process are discussed in Sect. 4. We here consider only the case of four cameras, which is considered to be a good number for three-dimensional LPT (Maas 1996). Finally, to create tracks the particles must be followed in time through the sequence of images. The tracking problem is discussed in Sect. 5, including a comparison of several algorithms from both the fluid dynamics and machine vision communities. In addition, it is shown in Sect. 5 that our new modification of a commonly used tracking algorithm can significantly improve its performance. We note that the stereoscopic matching and tracking steps may be interchanged so that particles are first tracked in two dimensions and then the tracks are matched together to create three-dimensional tracks. We, however, prefer to track in three dimensions rather than in two dimensions, since the particle seeding density is decreased by a factor of the added dimension, simplifying the tracking problem.

Finally, in Sect. 6, we demonstrate the validity of our LPT algorithms by measuring the statistics of Lagrangian velocity and acceleration in a high Reynolds number experimental flow and comparing them with previous well-known results.

2 Particle finding problem

The ideal algorithm for determining the centers of particles in camera images must meet several criteria:

1. *Sub-pixel accuracy*: the fidelity of the calculated tracks to the actual particle trajectories in the flow is due in a large part to the accuracy of the particle finding algorithm.
2. *Speed*: because of the very high data rate from cameras used to image a high Reynolds number flow, some consideration must be payed to the speed and efficiency of the algorithm.
3. *Overlap handling*: even for moderate particle seeding densities, there will be instances of particle images overlapping one another in the field of view of a single camera.
4. *Robustness to noise*: the cameras used in a LPT system will record noisy images.

Many algorithms have been proposed, including weighted averaging (Maas et al. 1993; Maas 1996; Doh et al. 2000), function fitting (Cowen and Monismith 1997; Mann et al. 1999), and template matching (Guezennec et al. 1994), as well as the standard PIV technique of image cross correlation (Adrian 1991; Westerweel 1993). In this section, we compare weighted average and both one-dimensional and two-dimensional Gaussian fitting. We also introduce a new particle finding algorithm using neural networks.

Before one can design a particle finding algorithm, one must make assumptions as to what constitutes a particle in an image. For all results presented in this article, we assume that every local maximum in intensity above a threshold represents a particle.

2.1 Weighted averaging

Weighted averaging, often referred to as a "center of mass" calculation, is both simple and commonly used in LPT systems. In a typical weighted averaging scheme, the digital image from a camera is first segmented into groups of pixels representing single particles. The center (x_c, y_c) of the particle is then determined by averaging the positions of its component pixels weighted by their intensity gray values. If we let $I(x,y)$ represent the pixel gray value at position (x, y), the horizontal coordinate of the particle center is given by

$$x_c = \frac{\sum_p x_p I(x_p, y_p)}{\sum_p I(x_p, y_p)}, \tag{1}$$

where the sums run over all pixels in a particular group. The vertical coordinate is defined similarly.

In this analysis, we used a weighted averaging scheme given by Maas and co-workers (Maas et al. 1993; Maas 1996) that attempts to handle overlapping particles. As above, we assume that every local intensity maximum represents a particle. Groups of contiguous pixels containing N local intensity maxima are assumed to contain N particles. In a preprocessing step, such groups of pixels are split into N subgroups, each containing only a single maximum. The pixel groups are split according to the assumption that the grayscale intensity of a particle image should continuously drop as the distance from the center increases. Local intensity minima are arbitrarily assigned to the pixel subgroup containing the minimum's brightest neighboring pixel.

This weighted averaging method is efficient and simple to implement computationally, as well as having some overlap handling capabilities. As will be demonstrated below, however, both the accuracy of the method and its performance on noisy images are poor compared to the others tested.

2.2 Gaussian fitting

If the functional form of the intensity profile of a particle image were known, fitting this function to each particle image would result in a very accurate determination of the particle centers. In general, however, this function is not known. A common approach is thus to approximate this intensity profile by a Gaussian (Mann et al. 1999).

$$I(x,y) = \frac{I_0}{2\pi\sigma_x\sigma_y} \exp\left\{-\frac{1}{2}\left[\left(\frac{x-x_c}{\sigma_x}\right)^2 + \left(\frac{y-y_c}{\sigma_y}\right)^2\right]\right\}. \tag{2}$$

To determine the particle centers in a group of pixels with N local maxima, N Gaussians are fit to the group.

This method is highly accurate and can handle overlapping particles. It is, however, computationally very expensive, requiring roughly a factor of four more computer time than the other methods studied. In addition, this method requires large particle images. Inspection of Eq. 2 reveals that each of these Gaussians requires the fitting of five parameters: I_0, σ_x, σ_y, x_c, and y_c. There must at minimum, then, be five pixels in every particle group, and many more than this to get an accurate fit. This problem is compounded for the case of overlapping particles, since a group with N local maxima must contain at least $5N$ pixels. Small particle images must therefore either be ignored or fit with some less accurate method.

These issues can be mitigated by using an approximation to the full two-dimensional Gaussian fitting scheme. Instead of fitting a Gaussian to the full particle pixel group, one can fit two one-dimensional Gaussians (Cowen and Monismith 1997). One Gaussian will determine the horizontal position of the particle and the second will determine the vertical position. This method uses the local maximum pixel as well as the four points directly adjacent horizontally and vertically. We label the coordinates of the horizontal points x_1, x_2, and x_3, labeling from left to right. Solving the system of equations

$$I_i = \frac{I_0}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x_i - x_c}{\sigma}\right)^2\right] \tag{3}$$

for $i = 1, 2, 3$ gives a horizontal particle coordinate of

$$x_c = \frac{1}{2}\frac{(x_1^2 - x_2^2)\ln(I_2/I_3) - (x_2^2 - x_3^2)\ln(I_1/I_2)}{(x_1 - x_2)\ln(I_2/I_3) - (x_2 - x_3)\ln(I_1/I_2)}, \tag{4}$$

with the vertical position of the particle defined analogously.

This 1D Gaussian Estimator retains much of the accuracy of the full two-dimensional fitting scheme while requiring fewer data points. In addition, it can be made significantly more computationally efficient by noting that the arguments of the required logarithms in Eq. 4 are all pixel intensities. In a digital image, there is a finite set of possible pixel intensities. The required logarithms may then be precomputed, cutting the computational work down to merely a few multiplications.

2.3 Neural networks

In addition to the two algorithms described above, we have also investigated the use of a neural network approach to particle finding. Neural networks have been used before in LPT to solve the tracking problem (Grant and Pan 1997; Chen and Chwang 2003; Labonté 1999, 2001) and to perform stereoscopic matching (Grant et al. 1998). Carosone et al. (1995) applied the Kohonen neural network to the problem of distinguishing the images of isolated particles from those of overlapping particles. Here, we used a new neural network scheme that solves the particle finding problem fully.

The neural network was shown square segments of images centered on local intensity maxima and trained to find single particles. This approach was used rather than showing the network full images since neural networks must have a standardized number of inputs and outputs. If the network were shown entire images, the number of outputs would vary since the number of particles in each image is in general unknown. Additionally, by showing the network standardized sections of images, overlap was trivially handled by training the network to find single particles near the center of each window of pixels.

The network used was a fully connected feed-forward network with an input layer of 81 neurons, corresponding to a 9×9 pixel window, a single hidden layer of 60 neurons, and an output layer of 2 neurons, corresponding to the horizontal and vertical position of the particle center. The 9×9 window was chosen so that the network was sure to see full particle images; changes to the size of this window should be relatively unimportant. The neurons were implemented using standard logistic activation functions and the network was trained using the standard backpropagation training algorithm augmented with a momentum term (Mitchell 1997).

While the training of the network was slow, it is a one-time cost. The subsequent computation of particle centers is a fast operation. As mentioned above, the network easily handles the problem of overlap. Additionally, as will be shown below, the network was very robust to noisy images, its performance degrading slowly as the noise level was increased.

3 Particle finding results

3.1 Methodology

The four algorithms described above were tested using computer generated synthetic images. Images were created assuming in turn two different intensity profiles for the light scattered by the particles. In each case, the underlying intensity profile was discretized by integrating it over each pixel.

The first intensity profile used was Gaussian. While this choice of profile certainly introduces a bias towards the two Gaussian particle finding algorithms tested, the intensity profile in a real experimental system is well-approximated by a Gaussian (Westerweel 1993; Cowen and Monismith 1997). The second intensity profile used was the Airy pattern generated by diffraction from a circular aperture. This profile is given by $(J_1(x)/x)^2$, where $J_1(x)$ is the cylindrical Bessel function of order 1 and x is the distance from the particle center. The ratio of the width of the Gaussian and the radius of the first peak of the Airy pattern was fixed at 0.524 in order in equalize the energy in the 2D Airy pattern and the 2D Gaussian. While choosing the Airy pattern and Gaussian in this way is not typical for studies of tracer particle imaging (Westerweel 1993), equalizing the energy in the two profiles in this way ensures that they are very different, and so provide a good test of the algorithms on different types of images. For each intensity profile, particle images were generated by imposing a pixel grid on the profile and integrating over each pixel.

Several sets of images were generated for each intensity profile, varying in turn the particle seeding density, particle image size, noise variance, and dynamic range of intensity levels. The images generated were 256×256 pixels in size with each pixel intensity stored in an 8 bit grayscale value, for a total of 256 gray levels. The particle intensity maxima were drawn from a Gaussian distribution centered at a gray value of 180. Portions of sample images generated with both Gaussian and Airy pattern intensity profiles are shown in Fig. 1.

The neural network used was trained using examples of images with only Gaussian intensity profiles, but with varying particle density, particle size, mean noise, and noise fluctuations.

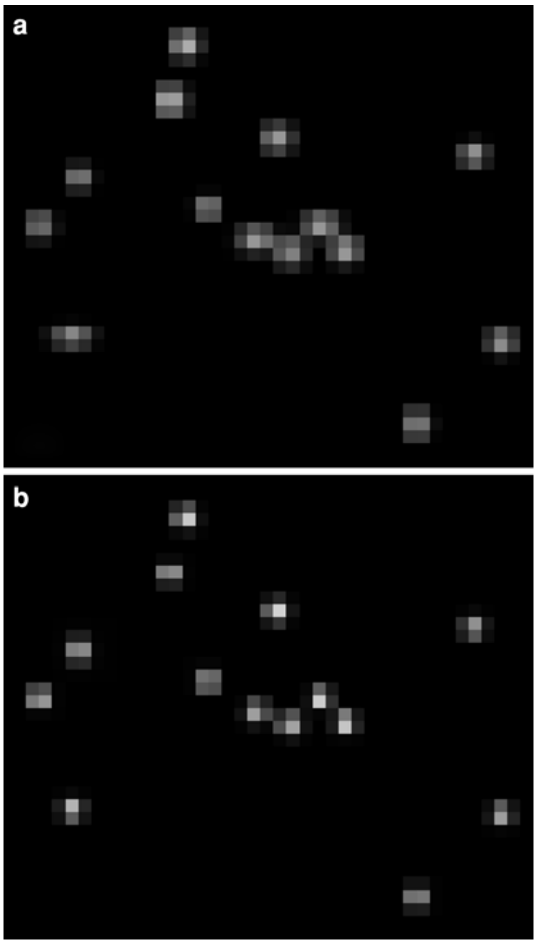


Fig. 1 Particle images at identical locations generated with **a** Gaussian intensity profiles and **b** Airy pattern intensity profiles

To determine the relative performance of the different image processing algorithms, errors were calculated by taking the difference between the found particle center and the actual center for both the horizontal and vertical coordinates. The distributions of these errors were symmetric with mean zero in all cases. The error distributions were also equivalent for both the vertical and horizontal components, showing no bias. In this section, we quantify the performance of the different algorithms based on the parameter Δ, defined by

$$\int_{-\Delta/2}^{\Delta/2} P(x)\mathrm{d}x = 0.8, \tag{5}$$

where $P(x)$ is the probability density function of the error as a function of distance from the true particle center. Δ gives an estimate of the pixel error associated with each test.

3.2 Particle seeding density

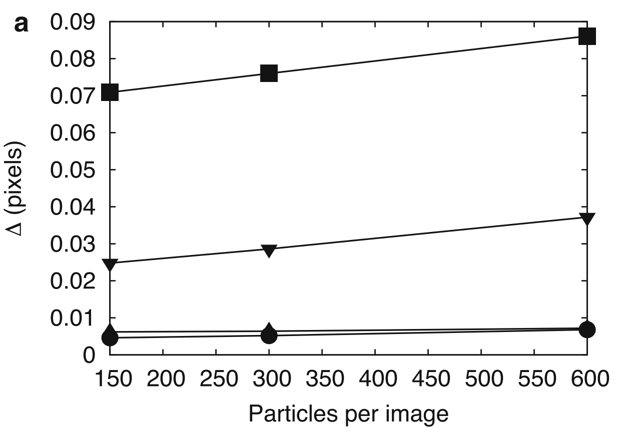
Increasing the particle seeding density introduces a greater likelihood of overlap into the image data. Overlapping particles are inherently more difficult to handle than isolated particles, and so one would expect the accuracy of an algorithm to decrease in general as the seeding density increases.

The performance of the four algorithms compared are shown in Fig. 2. For each of the seeding densities, the mean width of the particle image intensity profile was fixed at a half-width of 0.8 pixels. No noise was added to any of these images in these data sets.

As expected, the error of all the algorithms increased moderately with the seeding density. The accuracy of the Gaussian methods and the neural network also decreased when Airy pattern intensity profiles were used. The error increases slowly with seeding density for all methods, suggesting that a density of 600 particles per image is acceptable for LPT. The percentage of particles successfully identified, however, decreases markedly as the seeding density increases, dropping as low as 68% for the weighted averaging method, as shown in Fig. 3. For this reason, we consider seeding densities of only 300 particles per 256×256 pixel image for the rest of this analysis, since all methods tested maintain a yield of over 90% for a seeding density of 300 particles.

3.3 Particle image size

As with increasing the particle seeding density, increasing the size of particle images makes overlap more common. Larger particle images, however, also provide more information to the particle finding algorithm. The change in accuracy as a function of particle image size should therefore not be monotonic.



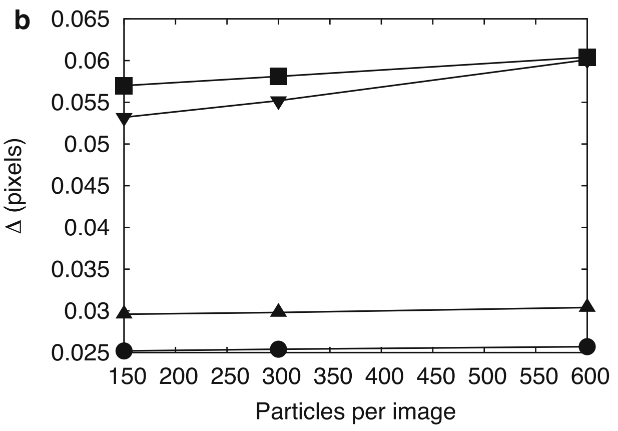


Fig. 2 The error of the four algorithms as a function of the number of particles per 256×256 pixel image. **a** Generated from images with Gaussian intensity profiles. **b** Generated from images with Airy pattern intensity profiles. Symbols: ■ weighted averaging, ● 2D Gaussian fit, ▲ 1D Gaussian Estimator, ▼ neural network

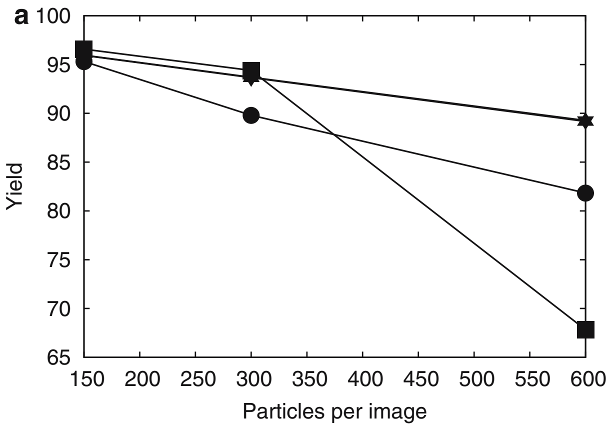
The performance of the four algorithms as a function of particle image size is shown in Fig. 4. For each value of the image size, the seeding density was fixed at 300 particles per image, and no noise was added to any images.

For the data sets using Gaussian intensity profiles, the error showed a minimum at an image half-width of 0.7–0.9 pixels for both Gaussian methods and for the neural network. The behavior of the weighted averaging algorithm is more varied. The peak in error at an image half-width of 0.7 may perhaps be explained by the introduction of overlap while at the same time having very small clusters of pixels representing particle images.

For the images generated with Airy pattern intensity profiles, the error reached a peak at an image half-width of 0.9–1.0 pixels. The decrease in error as the particles became larger than this for the Gaussian methods and the neural network may be traced to the fact that larger Airy patterns appear more Gaussian around their peaks. The weighted averaging method was observed to perform better on the Airy pattern intensity profiles than the Gaussian profiles, most likely because the extremely bright central peak of the Airy pattern weights the central pixel very heavily and is less affected by the outlying pixels of neighboring overlapping particles.

3.4 Noise level

After investigating the effects of particle image size and seeding density, these parameters were fixed at an intensity profile half-width of 0.8 pixels and roughly 300 particles per image, and noise was added in steps to the entire image. We used additive Gaussian noise in our tests. While we did not explicitly model each of the many types of noise that may occur in an LPT system (including errors from the camera sensor, thermal noise, the detection of scattered light, and many others), the central limit theorem says that as the various types of errors inherent in such a system are compounded, the noise distribution will approach a Gaussian. In addition, while the Poissonian shot noise associated with individual photon detection may dominate in low-light situations, other noise sources dominate when the light level is high and the overall noise may be well-approximated by a Gaussian (Westerweel 2000). High light



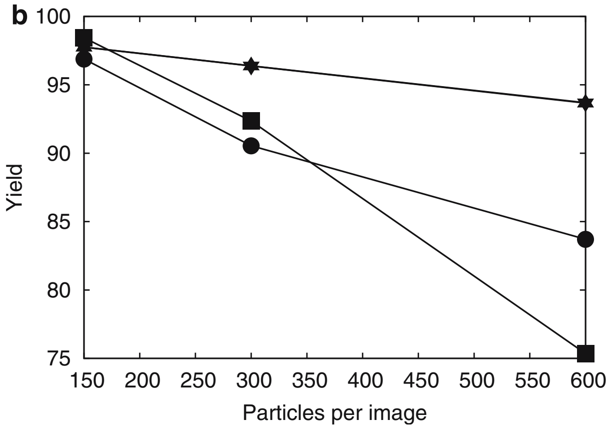


Fig. 3 The yield of the four algorithms as a function of the number of particles per 256×256 pixel image, defined as the percentage of particles successfully identified. **a** Generated from images with Gaussian intensity profiles, **b** Generated from images with Airy pattern intensity profiles. Symbols: ■ weighted averaging, ● 2D Gaussian fit, ▲ 1D Gaussian Estimator, ▼ neural network

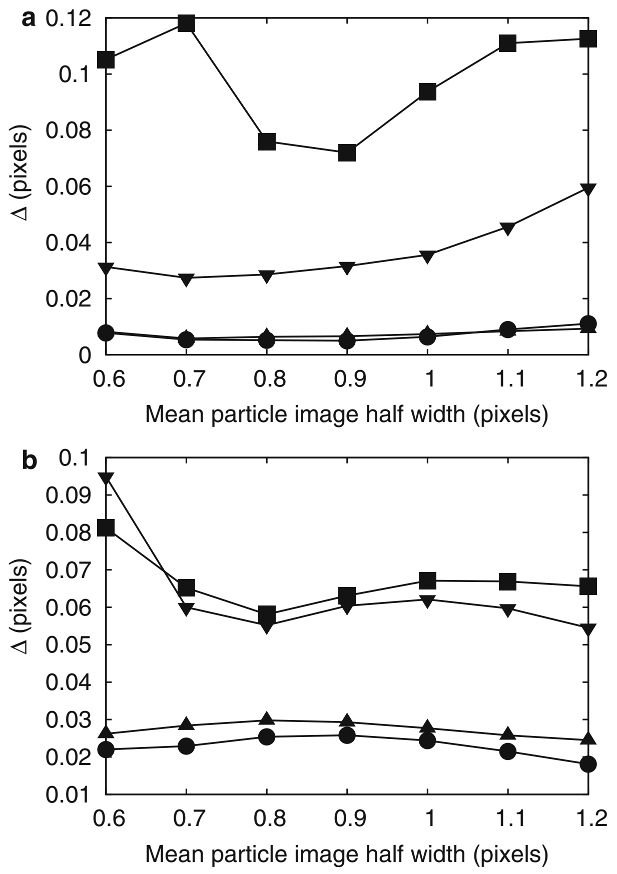


Fig. 4 The error of the four algorithms as a function of particle size. **a** Generated from images with Gaussian intensity profiles **b** Generated from images with Airy pattern intensity profiles. Symbols: ■ weighted averaging, ● 2D Gaussian fit, ▲ 1D Gaussian Estimator, ▼ neural network

levels are common when powerful lasers are used to image the tracer particles.

The mean of the Gaussian noise was fixed at zero, and the standard deviation was changed. The effects of changing the noise standard deviation on the performance of the four algorithms are shown in Fig. 5.

As was expected, when the noise standard deviation was increased, the accuracy of all the methods decreased. It is important to note, however, that the performance of the neural network degraded more slowly than that of the other algorithms, beating the 1D Gaussian Estimator for high levels of the noise when Gaussian intensity profiles were used and coming close to both Gaussian methods when Airy pattern intensity profiles were used.

3.5 Mean noise

In addition to changing the standard deviation of the noise, the mean noise was also varied. The noise standard deviation was fixed at 5.6, corresponding to 2.5 bits of noise, a typical value for CMOS cameras. As above the intensity profile half-width was fixed at 0.8 pixels with roughly 300 particles per image. The effects of changing the mean noise are shown in Fig. 6.

The weighted averaging method and both Gaussian methods showed sensitivity to the mean noise level, albeit less sensitivity than they did to changes in the noise standard deviation. The neural network, however, showed almost no change in accuracy as the mean noise was increased in images using Gaussian intensity profiles and only a slow decrease in accuracy when Airy pattern intensity profiles were used. Coupled with its slower degradation as the noise standard deviation was increased, the neural network method appears to be much more robust to noise than the other three methods analyzed.

3.6 Summary

From the analysis in this section, we may draw several conclusions. Figures 4, 5, and 6 make it clear that the

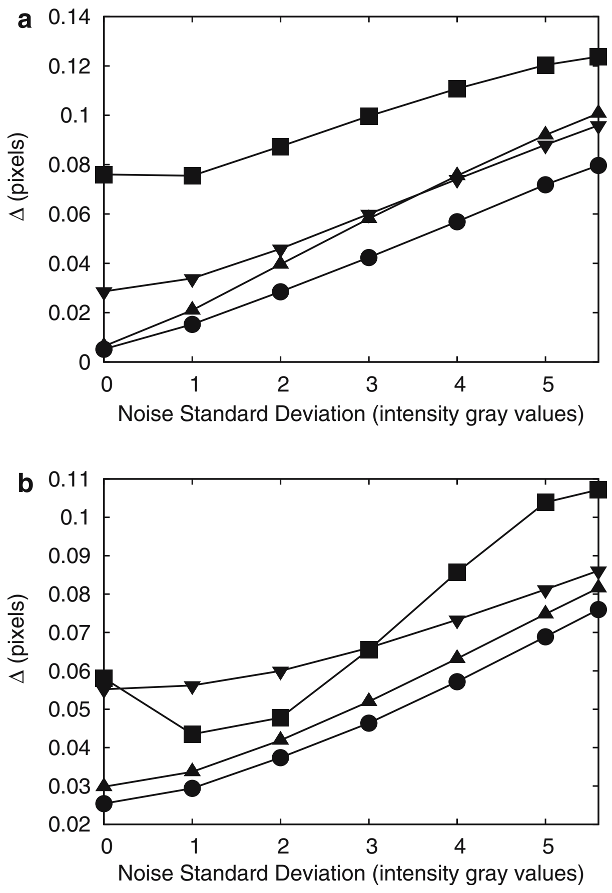


Fig. 5 The error of the four methods as a function of the standard deviation of the added white Gaussian noise. **a** Generated from images using Gaussian intensity profiles. **b** Generated from images using Airy pattern intensity profiles. Symbols: ■ weighted averaging, ● 2D Gaussian fit, ▲ 1D Gaussian Estimator, ▼ neural network

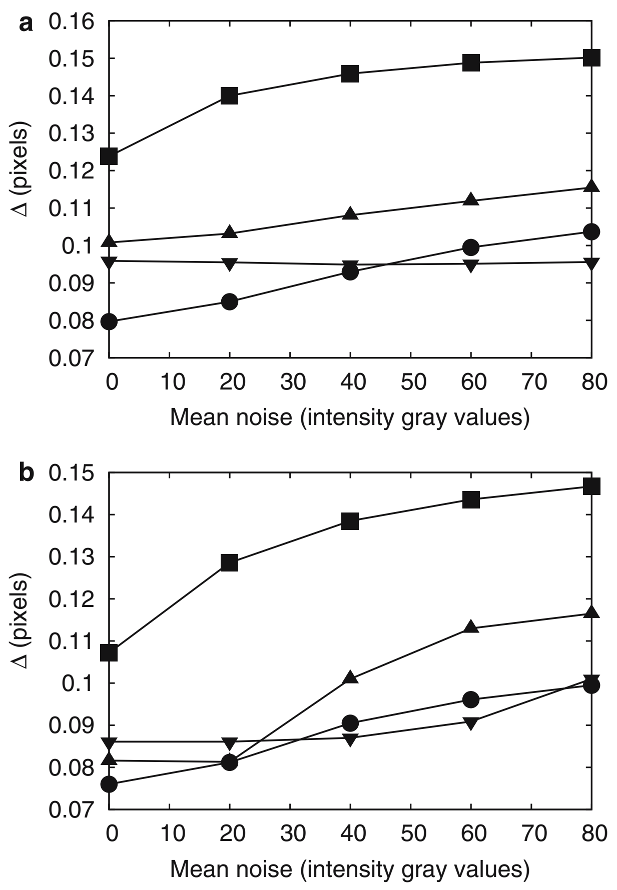


Fig. 6 The error of the four algorithms as a function of the mean noise level. While the weighted averaging and both Gaussian methods lose accuracy as the mean noise increases, the neural network is essentially unaffected. **a** Generated from images using Gaussian intensity profiles. **b** Generated from images using Airy pattern intensity profiles. Symbols: ■ weighted averaging, ● 2D Gaussian fit, ▲ 1D Gaussian Estimator, ▼ neural network

weighted averaging algorithm is significantly less accurate in all cases than the other three algorithms tested. Its main advantages are speed and simplicity of implementation. The 1D Gaussian Estimator, however, may be implemented so that it runs with comparable, if not superior, efficiency, and is therefore preferable to the weighted averaging method. In addition, since the accuracy of the two Gaussian methods used is comparable and the 1D Gaussian Estimator is significantly more efficient to implement, it is preferable to the two-dimensional Gaussian fit.

While the neural network did not perform as well as the 1D Gaussian Estimator in the absence of noise, it was far more robust to changes in both the mean level and fluctuations in the noise. It is also important to note that the accuracy of the neural network was roughly the same when applied to the Gaussian intensity profile images and the Airy pattern images even though it had only been trained on Gaussian images. Therefore, when trained on synthetic images, the network can be used to find particles in real experimental images.

In summary, the 1D Gaussian Estimator should be the preferred particle finding algorithm when image noise is negligible, while a neural network approach is superior when noise levels are appreciable.

4 Stereoscopic reconstruction

In three-dimensional LPT, images of the measurement volume from different view angles are recorded simultaneously, usually in different cameras. The positions of the cameras in the world coordinate system are determined by calibration. After image processing, as described in Sects. 2 and 3, the particle centers on the image planes of the cameras together with the known camera positions can be used to reconstruct the three-dimensional coordinates of the particles. Due to the lack of distinguishing features of the tracer particle images, the only constraint that can be used in stereoscopic matching is the fact that the "lines of sight" from different cameras to a particle must intersect at the particle center. Obviously, ambiguities will arise when the number of particles in the measurement volume increases. Dracos (1996) showed that for reasonable particle seeding density and optical setup, at least three cameras are needed in order to resolve matching ambiguity. For this work, we consider a system with four pinhole cameras and we focus on the effect of the arrangement of the cameras on the stereo-matching

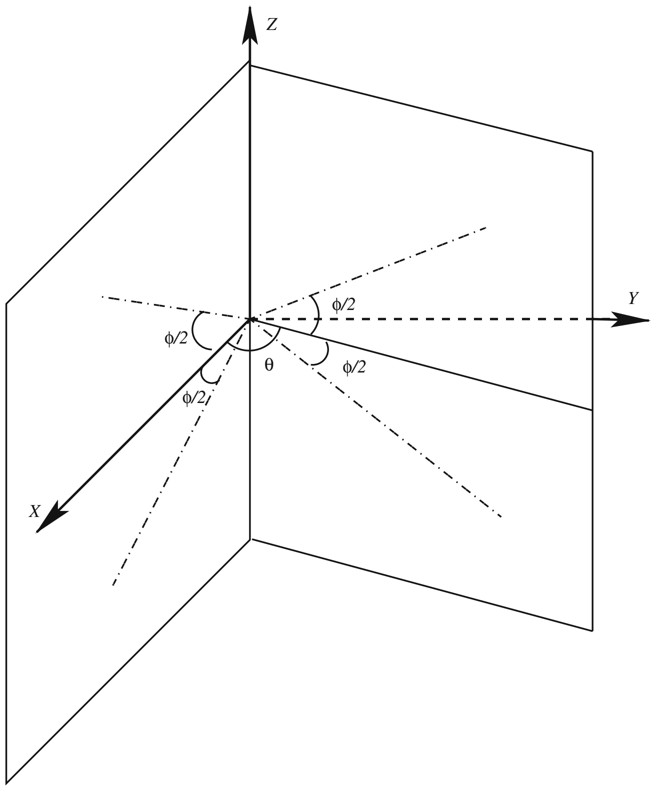


Fig. 7 Arrangement of camera axes

process. The four cameras are arranged in two vertical planes with angle θ between them. The two cameras within a vertical plane are symmetric about the horizontal plane and are separated by angle ϕ, as sketched in Fig. 7.

Our stereo-matching algorithm is a combination of those in Dracos (1996) and Mann et al. (1999) and will only be described briefly here. From a particle image on camera A, we can construct a "line of sight" passing through the particle image center and the perspective center of camera A. This line of sight is then projected onto the image plane of camera B. Any particle image on camera B that is within a tolerance ε away from the projection of the line of sight is then added into a list. In total, we have six such lists, one for each combination of two cameras. These lists are then checked for consistency. In this work, we keep only the results that are consistent among all four cameras. The particle coordinates in the three-dimensional world are then computed from the least square solution of the line of sight equations (Mann et al. 1999).

Motivated by the geometry of our experimental apparatus and the size of the cameras used, we generate for each camera 1,000 images of ~300 particles in a 10×10×10 cm^3 volume. The cameras are chosen with a magnification of 1:20 and the image resolution is 256×256; therefore, each pixel corresponds to roughly 0.4 mm in space. The particle positions in three dimensions are randomly chosen and are independent of each other. In the simulated images, the particle intensity profile half-width is fixed at 0.8 pixels and no noise was added to the images. These images are processed using the 1D Gaussian Estimator to find particle centers on the image plane, which are then fed to the stereo-matching algorithm to calculate the particle positions in three dimensions. The matching tolerance ε is chosen to be 0.5 pixels for all image sets. By comparison with the known particle three-dimensional positions, we are able to compute the yield and accuracy of the algorithm and the rms errors of the calculated particle three-dimensional positions, σ_x, σ_y, and σ_z, as summarized in Tables 1 and 2. Here yield is defined as the number of particles in three dimensions found by the matching algorithm, including potentially incorrect matches, divided by the actual number of particles that can be seen by all four cameras, while accuracy is the ratio of the number of correctly matched particles to the total number of particles found by the algorithm. We define a particle center position in three dimensions output by the algorithm as "correctly matched" if a simulated particle can be found within a sphere of radius 0.2 mm about that position, i.e., if the distance between the calculated center position to any particle is less than roughly 0.5 pixels. Because our 1D Gaussian Estimator can find particle centers on the image plane with rms error less than 0.1 pixels, a tolerance of 0.5 pixels for particle position in three dimensions is adequate and is also consistent with the tolerance ε used in stereo-matching.

From these results, we see that the yield is almost independent of camera arrangement. In fact, the yield of stereo-matching is almost entirely determined by the yield of particle center finding. For the sets of images that we simulated, the 1D Gaussian Estimator gives a yield of about 0.97, independent of camera setup. Assuming the performance of the center finding algorithm on images from four cameras is uncorrelated, the expected yield of stereo-matching is $0.97^4 = 0.89$, which is very close to the measured value for the cases of four cameras widely separated (large ϕ and θ). When either ϕ or θ is reduced, the images from different cameras become more and more similar; therefore, the yield increases slightly. However, this increase of yield is at the price of accuracy, because of the rise of ambiguity with decreasing camera separation.

Results in Table 1 indicate that as long as θ is kept at 90°, changing ϕ alone has only a small effect on stereo-matching. Both the yield and the accuracy vary slightly. The rms errors in the three-dimensional coordinates correspond to 0.05~0.07 pixels, which is at the same level of the error from particle center finding on the image plane. When ϕ is decreased, the increase of error in the x and the y direction is partially compensated by the decrease of error in the z direction. However, Table 2 suggests that when the four cameras are close to collinear, the position error along the axis that the cameras converging to can increase to more than 0.1 pixels, much higher than the error in two-dimensional particle center finding. Moreover, the accuracy decreases appreciably as θ is reduced while ϕ is fixed at 45°.

In reality, the situation will be much more complicated: the images are noisy, the particle intensity profile is not Gaussian, there will be calibration errors in the camera model, and so forth. Our simulations have by no means considered all these effects. However, an important guideline for practice that we can learn from these simulations is that at least one camera should be kept at a large angle away from the rest to reduce error in three-dimensional position.

Table 1 Effect of ϕ on stereo-matching; θ is fixed at 90°

ϕ (°)	yield (%)	accuracy (%)	σ_x (mm)	σ_y (mm)	σ_z (mm)
90	90.0	98.5	0.0241	0.0235	0.0290
60	89.8	98.5	0.0266	0.0256	0.0238
45	90.0	98.5	0.0280	0.0261	0.0220
30	90.0	98.4	0.0289	0.0268	0.0211
15	90.1	98.2	0.0289	0.0268	0.0205

Table 2 Effect of θ on stereo-matching; ϕ is fixed at 45°

θ (°)	yield (%)	accuracy (%)	σ_x (mm)	σ_y (mm)	σ_z (mm)
90	90.0	98.5	0.0280	0.0261	0.0220
60	90.0	98.3	0.0311	0.0254	0.0219
30	90.2	97.7	0.0394	0.0221	0.0208
15	90.4	97.0	0.0411	0.0200	0.0190

5 Particle tracking

Once particles have been found in the two-dimensional images and their three-dimensional positions have been constructed, they may be tracked in time. In general, tracking large numbers of particles over many frames is an example of a multidimensional assignment problem and is known to be NP-hard (Veenman et al. 2003). Instead of attempting to generate tracks using all data frames, therefore, tracking algorithms approximate the optimal solution by considering only a few frames at a time.

In order to measure the performance of a tracking algorithm quantitatively, we define the tracking error to be

$$E_{\mathrm{track}} = \frac{T_{\mathrm{imperfect}}}{T_{\mathrm{total}}}, \tag{6}$$

where $T_{\mathrm{imperfect}}$ is the number of tracks the algorithm failed to produce perfectly, when compared to the known tracks, and T_{total} is the total number of tracks in the data set (Chetverikov and Verestóy 1999). A perfectly determined track must contain no spurious points and must also begin at the same time as the actual data track, though it may not have the same length.

Here, we report E_{track} as a function of the parameter

$$\xi = \frac{\Delta r}{\Delta_0}, \tag{7}$$

where Δr is the average distance each particle moves from one frame to the next and Δ_0 is the average separation between a particle and its closest neighbor in a single frame. We note that ξ is exactly the inverse of the parameter p defined by Malik et al. (1993). When $\xi \ll 1$, the particle seeding density is low and/or the particles move slowly from frame to frame; in this limit, tracking is not difficult. When ξ becomes of order unity, however, the particle density is high and/or the particles move quickly, making tracking much more difficult.

The algorithms were tested using fully three-dimensional direct numerical simulation (DNS) data of fluid particles in turbulence provided by Lance Collins at Cornell University. The DNS was carried out at a Taylor microscale Reynolds number of $R_\lambda = 52$ in a $2\pi \times 2\pi \times 2\pi$ box with periodic boundary conditions; despite this low Reynolds number, however, the individual particle trajectories undergo chaotic motion and are therefore not trivial to track. Since Δr was fixed by the DNS, the parameter ξ was varied by changing Δ_0. This was accomplished by first fitting the DNS tracks into progressively smaller periodic boxes. When a track left the box, it was cut and the remaining portion of the track was wrapped back into the box. All of these sub-tracks were then considered to start at time zero, increasing the particle density and therefore increasing ξ. This approach also simulates particles leaving the measurement volume; note, however, that since no new particles enter, ξ changes over time. Since E_{track} measures the number of tracks generated perfectly, however, the largest value of ξ defines the difficulty of the tracking problem. In addition to increasing the difficulty of the tracking problem by this method, we also included additive Gaussian noise of magnitude up to 0.5 Δr to the particle positions, with no appreciable change in the relative performance of the tracking algorithms tested.

We note that the tracking algorithms were simply fed lists of fluid particle positions, with no generation of synthetic images. The results of testing the tracking algorithms presented here are therefore independent of the center finding method chosen.

Before describing the tested algorithms individually, we present some definitions and common features of all the algorithms. Let $\mathbf{x}_i^n$ denote the ith position in the nth frame. A tracking algorithm, then, tries to find an $\mathbf{x}_j^{n+1}$ for each $\mathbf{x}_i^n$ such that $\mathbf{x}_j$ is the position of the particle in frame $n+1$ that was at position $\mathbf{x}_i$ in frame n. In order to determine which of all the $\mathbf{x}_j^{n+1}$ to choose, a tracking algorithm defines a cost ϕ_{ij}^n (which we define below for several algorithms) for each pair of $\mathbf{x}_i^n$ and $\mathbf{x}_j^{n+1}$. The optimal solution to the tracking problem would make links between $\mathbf{x}_i^n$ and $\mathbf{x}_j^{n+1}$ such that

$$\Phi = \sum_n \sum_i \sum_j \phi_{ij}^n \tag{8}$$

is minimized. As mentioned above, however, this is a multidimensional assignment problem and is intractable. The tracking algorithms considered here approximate this optimal solution. The most common approximation made is to restrict the number of frames over which Φ is optimized, sometimes referred to as a greedy matching approximation (Veenman et al. 2001). Greedy matching algorithms are preferable to iterative schemes (Sethi and Jain 1987; Ohmi and Li 2000) for the present application since such schemes are unsuitable for processing long tracks. Most tracking algorithms, including those presented here, also restrict the $\mathbf{x}_j^{n+1}$ investigated as possible matches for each $\mathbf{x}_i^n$ by imposing a limit on the distance a particle can travel from one frame to the next.

Within these approximations, a tracking algorithm is specified by two parameters: the heuristic used to calculate the cost ϕ_{ij}^n and the method used to break tracking conflicts. A conflict occurs when $\phi_{ij}^n = \phi_{kj}^n$ for $i \neq k$, so that two particles in frame n match equally well with the same particle in frame $n+1$. It is important to note that these conflicts do not, in general, arise from overlapping particle images, since we track here in three dimensions: overlapping images that were not properly identified will not pass through the stereomatching step and will consequently not be fed to the tracking algorithm.

In this study, we have compared four tracking heuristics and two methods of conflict breaking. Illustrations of the heuristics are shown in Fig. 8. The tracking heuristics used were

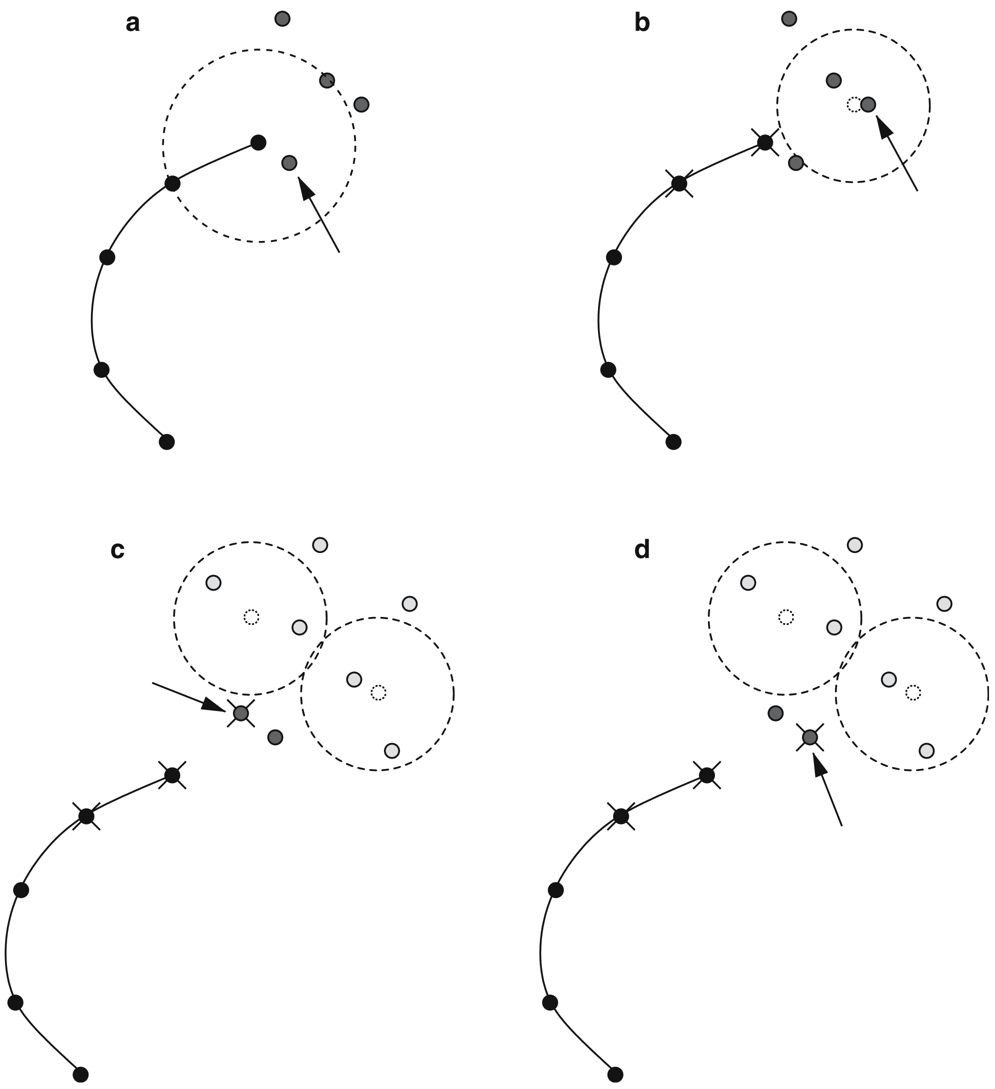


Fig. 8 Diagrams of the four tracking heuristics. The *black circles* and *line* indicate positions already joined into a track. *Dark gray circles* indicate positions a single frame into the future, while *light gray circles* signify positions two frames into the future. *Open circles* indicate estimated positions, and the *crossed circles* indicate the positions used to generate the estimates. **a** Nearest Neighbor heuristic, **b** the 3 Frame: Minimum Acceleration, **c** the 4 Frame: Minimum Change in Acceleration, and **d** the 4 Frame: Best Estimate. In each case, the *arrow* points out which position will be chosen as the next point on the track

1. *Nearest Neighbor (NN)*: the tracking cost is given by the distance between the point in frame n and the point in frame $n+1$:

$$\phi_{ij}^n = \left\|\mathbf{x}_j^{n+1} - \mathbf{x}_i^n\right\|. \tag{9}$$

2. *3 Frame: Minimum Acceleration (3MA)*: the position of the particle in frame $n-1$ is used along with the position in frame n to estimate a velocity and therefore a position $\tilde{\mathbf{x}}_i^{n+1}$ for the particle in frame $n+1$, where

$$\tilde{\mathbf{x}}_i^{n+1} = \mathbf{x}_i^n + \tilde{\mathbf{v}}_i^n \Delta t, \tag{10}$$

where $\tilde{\mathbf{v}}_i^n$ is the estimated velocity. The tracking cost is calculated for all the particles falling in a search volume surrounding the estimate, and is given by the particle acceleration (Malik et al. 1993; Dracos 1996):

$$\phi_{ij}^n = \frac{\left\|\mathrm{x}_j^{n+1} - 2\mathrm{x}_i^n + \mathrm{x}_i^{n-1}\right\|}{2\Delta t^2}, \tag{11}$$

where Δt is the time elapsed between frames. Since this method requires that a track consist of at least two points (one in frame n and one in frame $n-1$), we also specify that the first two frames in the track are joined using the nearest neighbor heuristic.

3. *4 Frame: Minimum Change in Acceleration (4MA)*: The position of the particle in frame $n+1$ is estimated in the same way as in the 3MA algorithm. For each of the particles in the search volume in frame $n+1$, a position $\tilde{\mathbf{x}}_i^{n+2}$ in frame $n+2$ is estimated to be

$$\tilde{\mathbf{x}}_i^{n+2} = \mathbf{x}_i^n + \tilde{\mathbf{v}}_i^n(2\Delta t) + \tilde{\mathbf{a}}_i^n(2\Delta t)^2 \tag{12}$$

and particles in a search volume around it are investigated. The particle chosen as the best match in frame $n+1$ is that which leads to the smallest change in acceleration from frame $n+1$ to frame $n+2$ (Malik et al. 1993; Dracos 1996):

$$\phi_{ij}^n = \frac{1}{2\Delta t^2}\left\{\left\|\mathbf{x}_j^{n+2} - 2\mathbf{x}_j^{n+1} + \mathbf{x}_i^n\right\| - \left\|\mathbf{x}_j^{n+1} - 2\mathbf{x}_i^n + \mathbf{x}_i^{n-1}\right\|\right\}. \tag{13}$$

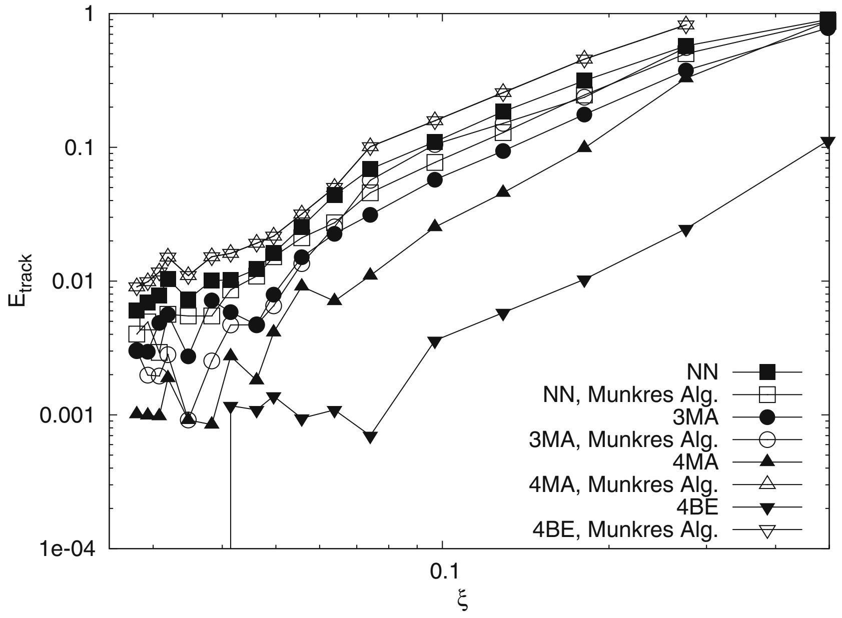


Fig. 9 The results of testing the tracking algorithms. The tracking error (Eq. 6) is plotted against the parameter ξ for each of the algorithms tested. The 4BE algorithm with no conflict breaking is clearly superior to the other algorithms tested, making zero errors for $\xi < 0.04$. In all cases except for the NN algorithm, using the Munkres algorithm to break conflicts degraded performance

As with the three frame method, the first two points on a track are joined using a nearest neighbor heuristic.

4. *4 Frame: Best Estimate (4BE)*: we have developed this algorithm as an extension of the four-frame method described above. We replace the ϕ_{ij}^n defined above by the distance between particles in frame $n+2$ and the second estimated position:

$$\phi_{ij}^n = \left\| \mathbf{x}_j^{n+2} - \tilde{\mathbf{x}}_i^{n+2} \right\|, \tag{14}$$

where $\tilde{\mathbf{x}}_i^{n+2}$ is the estimated position in frame $n+2$. This algorithm therefore makes no attempt to estimate the third time derivative along the track segment.

Conflicts were handled in two ways. The simplest way to handle conflicts is to give up: when a particle in frame $n+1$ is the best match for multiple particles in frame n, the involved tracks are stopped at frame n and a new track is considered to have begun in frame $n+1$. One may also handle conflicts by choosing the set of links from frame n to frame $n+1$ so that the total cost $\sum_{ij} \phi_{ij}^n$ is minimized (Veenman et al. 2001). This task, now a two-dimensional assignment problem, may be solved efficiently using the Munkres Algorithm (Bourgeois and Lasalle 1971).

The results of testing the different tracking algorithms are shown in Fig. 9. The 4BE algorithm with no conflict breaking cleary performed better than all the other tested algorithms, making no tracking mistakes for $\xi < 0.04$. Interestingly, when the Munkres algorithm was used to break conflicts, the 4BE algorithm performed the poorest.

Figure 9 also shows a dramatic difference in performance between the 4BE and 4MA algorithms, even though the two algorithms are very similar. This difference in accuracy, however, is readily explainable. As described above, the 4MA algorithm uses the change in acceleration as its heuristic, essentially estimating the third derivative of position. With the limited time resolution in these simulations, the third derivative is very difficult to distinguish from noise. Because of this, the minimum change in acceleration heuristic becomes unreliable for high values of ξ.

One may also see from Fig. 9 that the use of the Munkres algorithm to break tracking conflicts degrades performance for all the algorithms tested except for the NN algorithm. This suggests that the wisest choice when faced with a tracking conflict is to end all conflicting tracks and begin a new track.

6 Experimental tests

In order to ensure that our proposed algorithms work in a real experiment, we have measured the statistics of the Lagrangian velocity and acceleration in a turbulent flow with a Taylor microscale Reynolds number of $R_\lambda = 690$. Turbulence is generated in a closed cylindrical chamber containing roughly 0.1 m^3 of water by the coaxial counter-rotation of two baffled disks, as described in detail in Voth et al. (2002). Measurements are made in a subvolume of roughly 2.5×2.5×2.5 cm^3 in the center of the tank where the mean flow is negligible. The temperature of the water is controlled at 20.6 ± 0.1°C.

In our flow at $R_\lambda = 690$, the Kolmogorov length scale η is 30 µm and the Kolmogorov time scale τ_η is 0.9 ms. We seed the flow with transparent polystyrene microspheres of diameter 25 µm. The particles have a density 1.06 times that of water and are nearly neutrally buoyant. The particles are illuminated using a 90 W Nd:YAG laser pulsed at 27 kHz. We image the particles using three Phantom v7.1 CMOS cameras from Vision Re-

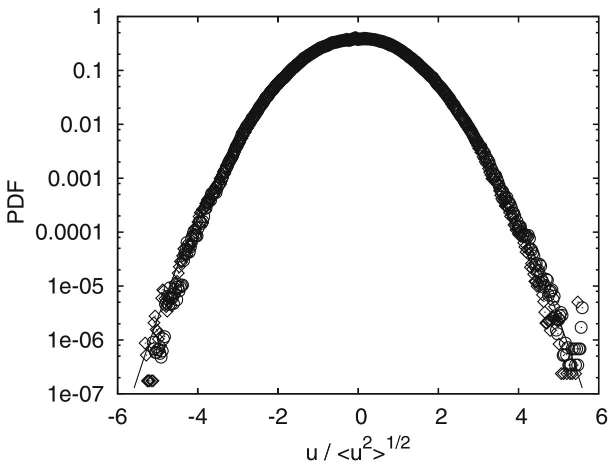


Fig. 10 PDFs of the radial velocity components. The *symbols* denote the two components of measured data, while the *solid line* is a Gaussian. As expected, the velocity data fall almost perfectly on the Gaussian

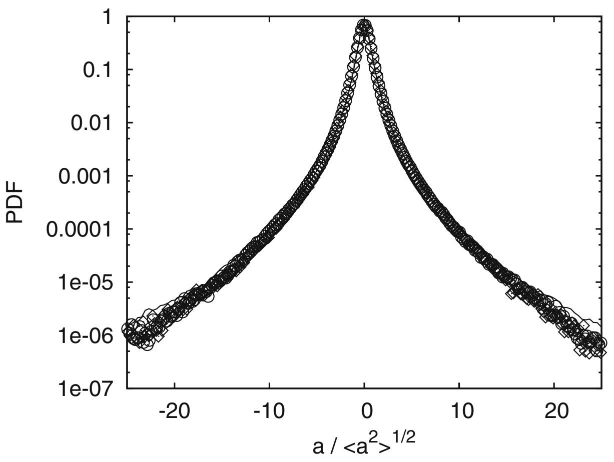


Fig. 11 PDFs of the radial acceleration components. The *symbols* denote the two components of measured data. The *solid line* is data measured by Mordant et al. (2004). The tails of our measured distribution are somewhat depressed due to our low time resolution

search, Inc. These cameras have a framerate of 27,000 pictures per second at a resolution of 256×256 pixels, giving us approximately 24 measurements per τ_η. The cameras are arranged in a single plane with an angular separation of roughly 45°.

The measured particle trajectories are used to calculate Lagrangian velocities and accelerations. Time derivatives were calculated from the tracking data by convolution with a Gaussian smoothing and differentiating kernel as described in Mordant et al. (2004).

As shown above for simulated data, the 1D Gaussian Estimator and the 4BE tracking algorithm gave the best combination of accuracy and efficiency of the algorithms tested. We have therefore used them in our experiment. Figure 10 shows the measured probability density functions (PDFs) for the two radial velocity components. As expected, the PDFs are almost perfectly Gaussian. The axial velocity component (not shown) is less Gaussian, as noted in Voth et al. (2002) for this flow.

Figure 11 shows the measured PDFs of the two radial acceleration components. The PDFs show the expected stretched exponential shape. We compare this measured data with the previous measurements of Mordant et al. (2004), who used very fast silicon strip detectors to image tracer particles in our flow. Mordant et al. (2004) took images at a rate of 70 kHz in a volume of 2×2×2 mm with effectively 512×512 pixels. We see some depression in the tails of our measured PDF, which we attribute to the poorer temporal and spatial resolution in our experiment, since Mordant et al. (2004) took nearly three times as many pictures per second.

Given that the 1D Gaussian Estimator and the 4BE algorithm produce accurate Lagrangian statistics in our high Reynolds number flow, we conclude that they can successfully be applied to real experiments.

7 Conclusions

The choice of algorithms used in a three-dimensional LPT system is important for the overall accuracy of such a system. We have proposed both a particle finding algorithm and a tracking algorithm. For particle center finding, any advantages the commonly used method of weighted averaging enjoys are outweighed by the significant gains in accuracy one can achieve with (1) for high signal-to-noise ratios the comparably efficient 1D Gaussian Estimator and (2) for low signal-to-noise ratios our new neural network scheme. For camera placement, at least one camera must be placed at a large angular separation from the others in order to increase accuracy by removing matching ambiguities. Finally, our new 4BE tracking algorithm outperforms the other algorithms tested by a large margin, and the addition of the Munkres algorithm (Bourgeois and Lasalle 1971) for breaking conflicts to a tracking method does not increase the tracking accuracy.

Additionally, we have verified that the best algorithms tested, the 1D Gaussian Estimator and the 4BE tracking algorithm, produce particle tracks with the correct flow statistics when applied to a real high Reynolds number flow.

Acknowledgements The authors would like to thank Lance Collins at Cornell University for contributing DNS data for testing the tracking algorithms presented here. This work was supported by the National Science Foundation under grants PHY-9988755 and PHY-0216406.

References

Adrian RJ (1991) Particle-imaging techniques for experimental fluid mechanics. Annu Rev Fluid Mech 23:261–304

Bourgeois F, Lassalle J-C (1971) An extension of the Munkres algorithm for the assignment problem to rectangular matrices. Commun ACM 14:802–804

Carosone F, Cenedese A, Querzoli G (1995) Recognition of partially overlapped particle images using the Kohonen neural network. Exp Fluids 19:225–232

Chen Y, Chwang AT (2003) Particle image velocimetry system with self-organized feature map algorithm. J Eng Mech ASCE 129:1156–1163

Chetverikov D, Verestóy J (1999) Feature point tracking for incomplete trajectories. Computing 62:321–338

Cowen EA, Monismith SG (1997) A hybrid digital particle tracking velocimetry technique. Exp Fluids 22:199–211

Doh D-H, Kim D-H, Choi S-H, Hong S-D, Saga T, Kobayashi T (2000) Single-frame (two-field image) 3-D PTV for high speed flows. Exp Fluids 29:S85–S98

Dracos Th (1996) Particle tracking in three-dimensional space. In: Dracos Th (ed) Three-dimensional velocity and vorticity measuring and image analysis techniques. Kluwer, Dordrecht

Grant I, Pan X (1997) The use of neural techniques in PIV and PTV. Meas Sci Technol 8:1399–1405

Grant I, Pan X, Romano F, Wang X (1998) Neural-network method applied to the stereo image correspondence problem in three-component particle image velocimetry. Appl Opt 37:3656–3663

Guezennec YG, Brodkey RS, Trigui N, Kent JC (1994) Algorithms for fully automated three-dimensional particle tracking velocimetry. Exp Fluids 17:209–219

La Porta A, Voth GA, Crawford AM, Alexander J, Bodenschatz E (2001) Fluid particle accelerations in fully developed turbulence. Nature 409:1017–1019

Labonté G (1999) A new neural network for particle-tracking velocimetry. Exp Fluids 26:340–346

Labonté G (2001) Neural network reconstruction of fluid flows from tracer-particle displacements. Exp Fluids 30:399–409

Maas H-G (1996) Contributions of digital photogrammetry to 3-D PT. In: Dracos Th (ed) Three-dimensional velocity and vorticity measuring and image analysis techniques. Kluwer, Dordrecht

Maas H-G, Gruen A, Papantoniou D (1993) Particle tracking velocimetry in three-dimensional flows—part 1. Photogrammetric determination of particle coordinates. Exp Fluids 15:133–146

Malik NA, Dracos Th, Papantoniou DA (1993) Particle tracking velocimetry in three-dimensional flows—part 2. Particle tracking. Exp Fluids 15:279–294

Mann J, Ott S, Andersen JS (1999) Experimental study of relative, turbulent diffusion. Risø National Laboratory Report Risø-R-1036(EN)

Mitchell TM (1997) Machine learning. McGraw-Hill, Boston, pp 81–127

Mordant N, Crawford AM, Bodenschatz E (2004) Experimental Lagrangian acceleration probability density function measurement. Physica D 193:245–251

Ohmi K, Li H-Y (2000) Particle-tracking velocimetry with new algorithms. Meas Sci Technol 11:603–616

Sethi IK, Jain R (1987) Finding trajectories of feature points in a monocular image sequence. IEEE Trans Pattern Anal Mach Intell 9:56–73

Veenman CJ, Reinders MJT, Backer E (2001) Resolving motion correspondence for densely moving points. IEEE Trans Pattern Anal Mach Intell 23:54–72

Veenman CJ, Reinders MJT, Backer E (2003) Establishing motion correspondence using extended temporal scope. Artif Intell 145:227–243

Voth GA, La Porta A, Crawford AM, Alexander J, Bodenschatz E (2002) Measurement of particle accelerations in fully developed turbulence. J Fluid Mech 469:121–160

Westerweel J (1993) Digital particle image velocimetry—theory and applications. PhD Dissertation, Delft University Press

Westerweel J (2000) Theoretical analysis of the measurement precision in particle image velocimetry. Exp Fluids 29:S3–S12

Yeung PK (2002) Lagrangian investigations of turbulence. Annu Rev Fluid Mech 34:115–142