

CENTRO DE INVESTIGACIONES EN ÓPTICA

MASTER THESIS

---

**Comparison of 3 techniques used  
for the estimation of displacement  
in PIV**

---

*Author:*

Rafael Guillermo González  
Acuña

*Supervisor:*

Bernardino Barrientos  
García

*A thesis submitted in fulfilment of the requirements  
for the degree of Master in engineering*

*in the*

Grupo de Metrología Óptica  
CENTRO DE INVESTIGACIONES EN ÓPTICA

*“The measure of love is to love without measure.”*

Augustine of Hippo

# ACKNOWLEDGEMENTS

I want to thank God for giving us an amazing and elegant world which we can explore, to my mother for giving me life and love. . .

Besides, I would like to thank Centro de investigaciones en Óptica and CONACyT for the received support during the last 2 years of the degree. Similarly, I am grateful to Dr. Bernardino Barrientos, for his supervision work, and Dr. Mariano Cerca and Dr. Miguel Mora, for their reviewing task, which really improved the thesis manuscript.

for anyone seeking wisdom. . .

# Contents

<b>1</b>	<b>Introduccion</b>	<b>8</b>
1.1	PIV . . . . .	8
1.2	Methods for obtaining displacement . . . . .	9
1.2.1	The correlation method . . . . .	9
1.2.2	The Lucas-Kanade algorithm . . . . .	9
1.2.3	Hybrid Lucas-Kanade algorithm . . . . .	10
<b>2</b>	<b>Particle image velocimetry by correlation of images</b>	<b>12</b>
2.1	Introduction . . . . .	12
2.2	Mathematical background . . . . .	13
2.2.1	Statistical interpretation of correlation . . . . .	13
2.2.2	Fourier interpretation of correlation . . . . .	14
2.3	PIV evaluation . . . . .	16
2.3.1	PIV step by step . . . . .	16
2.3.2	Observations when FFT is applied to PIV . . . . .	17
2.3.3	Observations in PIV evaluation . . . . .	18
2.4	Peak detection . . . . .	18
2.5	Simulations and results . . . . .	19
2.5.1	Simulations . . . . .	20
2.5.2	Results with the artificial particle image generator . . . . .	21
<b>3</b>	<b>The Lucas-Kanade algorithm</b>	<b>24</b>
3.1	Introduction . . . . .	24
3.2	Mathematical background . . . . .	24
3.2.1	Suppositions . . . . .	24
3.2.2	Theoy . . . . .	25

3.2.3	The optical flow model . . . . .	26
3.3	Evaluation of the Lucas-Kanade algorithm . . . . .	28
3.3.1	Lucas-Kanade algorithm step by step . . . . .	28
3.3.2	Highlights of the Lucas-Kanade algorithm . . . . .	29
3.4	Simulations and results . . . . .	29
<b>4</b>	<b>Artificial intelligence applied to the Lucas-Kanade algorithm</b>	<b>32</b>
4.1	Introduction . . . . .	32
4.2	Introduction to artificial intelligence . . . . .	32
4.3	The particle swarm optimization algorithm . . . . .	33
4.3.1	Theory . . . . .	33
4.3.2	Implementation example . . . . .	34
4.4	The hybrid algorithm . . . . .	35
4.5	Description of the implemented code . . . . .	36
4.5.1	Pseudocode of the hybrid algorithm . . . . .	36
4.5.2	An example of the implementation of the hybrid code . . . . .	37
<b>5</b>	<b>Results</b>	<b>40</b>
5.1	Introduction . . . . .	40
5.2	Description of the numerical analysis . . . . .	40
5.2.1	Relative error in percentage . . . . .	40
5.2.2	Subpixel displacement . . . . .	45
5.3	Results of the correlation method . . . . .	45
5.3.1	Constant displacement for correlation . . . . .	47
5.3.2	Sinusoidal displacement . . . . .	53
5.3.3	Vortex-like displacement . . . . .	60
5.4	Results of the Lucas-Kanade algorithm . . . . .	67
5.4.1	Constant displacement for Lucas-Kanade algorithm . . . . .	67
5.4.2	Sinusoidal displacement . . . . .	74
5.4.3	Vortex-like displacement . . . . .	80
5.5	Results of hybrid Lucas-Kanade . . . . .	86
5.6	Experimental results . . . . .	86
5.7	Conclusions . . . . .	90
5.8	Future work . . . . .	91

# ABSTRACT

Results of numerical simulations related to the application of three processing techniques in PIV are presented. The compared techniques are the standard correlation method via Fourier transform, an optical flow method (Lucas-Kanade method) and a combination of the latter and a artificial intelligence algorithm. The parameters under analysis are the particle density, the diameter of the particles, the range of displacement, the type of displacement (constant, vortex-like and sinusoidal), the noise of displacement, the size of the subimage, the intensity variations, and the profile of the particles (rectangle, Gaussian and triangular). Plots of percentage relative errors are included for each parameter. The main findings are that the correlation technique yields the best results in general and that the other two techniques show their best performance when applied to low-contrast images.

# SUMMARY

In the velocimetry laboratory there is interest in developing a professional software for the computation of displacement fields from images mainly of particle image velocimetry (PIV) and digital image correlation (DIC). Currently, for that task, there are basically two numerical methods, which have their strengths and their weaknesses. These two algorithms are studied in this work in order to assess their performance and do a comparison. The algorithms are the correlation technique, the Lucas-Kanade algorithm and a hybrid Lucas-Kanade, which incorporates an artificial intelligence algorithm.

The correlation technique is the traditional technique in PIV and DIC. It uses the concept of correlation to measure the displacement of the particles, and this statistical measure of the displacement can be done by applying the Fourier transform. The Lucas-Kanade algorithm computes the displacement by performing a least squares minimization routine. The hybrid Lucas-Kanade includes an algorithm called particle swarm optimization algorithm. The idea here is that the particle swarm optimization algorithm helps Lucas Kanade to improve and optimize its result.

To compare these three algorithms, several parameters are evaluated: size of the scan window, size of the diameter, magnitude of the displacement, noise level, contrast of the image, etc. The evaluation considers three types of displacement distribution: constant displacement, sinusoidal displacement and vortex-like displacement.

The main results of this work are that the correlation technique in general yields more accurate results and that the hybrid Lucas-Kanade is especially accurate for low-contrast images (but with heavier processing). Therefore, it is worth to incorporate both algorithms in a future version of processing software.



# Chapter 1

## Introduccion

### 1.1 PIV

Particle image velocimetry (PIV) is an optical method of flow visualization used in research. It is used to obtain instantaneous velocity measurements, displacements and related properties in fluids.

The fluid is seeded with tracer particles which, for sufficiently small particles, are assumed to faithfully follow the flow dynamics. The fluid with entrained particles is illuminated so that particles are visible. The motion of the seeding particles is used to calculate speed and direction (the velocity field) of the flow being studied.

Figure 1.1 describes a typical set-up for PIV recording in a wind tunnel. A plane within a particular flow is illuminated in two instants by a high-intensity light source, such as a laser. The light scattered by the particles is recorded generally on two separate and consecutive video frames. The displacement undergone by the particles between the video frames is to be determined and then by considering the time between the two acquisitions, the local velocity can be obtained.[2]

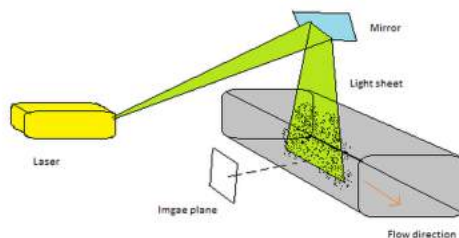


Figure 1.1: Typical set-up for PIV recording.

## 1.2 Methods for obtaining displacement

In this experimental technique, the calculation of the displacement vector for each pixel of an image is of sheer importance. This is quite a complex problem since matching one particle in two different frames is a task which may require heavy processing. In this work we deal with three algorithms to carry out this task: the correlation method, which is based on the covariance of corresponding subwindows of an image; least squares method (Lucas-Kanade algorithm), that uses the information of a surrounding pixels to calculate the displacement vector  $\vec{d}$  of one pixel; and a hybrid version of the latter, where an artificial intelligence code is applied.

The comparison of these three algorithms is the core of this work. So it is important to understand their performance and find their advantages and disadvantages.

The parameters under analysis are the particle density, the diameter of the particles, the range of displacement, the type of displacement (constant, vortex-like and sinusoidal), the noise of displacement, the size of the subimage, the intensity variations, and the profile of the particles (rectangle, Gaussian and triangular).

The displacement of the tracer particles  $\vec{d}$  can be expressed in its components  $u$  and  $v$ , where  $u$  is the horizontal component and  $v$  is the vertical component.

### 1.2.1 The correlation method

The correlation method works with two images  $I_1$  and  $I_2$ . This method finds the similarity between samples of the two images by the computation of the correlation.

This method has two ways to do it, the first one is using the correlation by its definition and another is by exploiting the Fourier transform properties (there is a property of the Fourier transform that simplifies the correlation, the convolution property of the Fourier transform). Both ways are equivalent, in Chapter 2 it is explained the two ways and the corresponding characteristic of theses.[1,2,3]

### 1.2.2 The Lucas-Kanade algorithm

The Lucas-Kanade method finds the displacement  $\vec{d}$  by assuming the two images  $I_1(x, y)$  and  $I_2(x, y)$  are copies of one another, which means that the derivatives from the images are equal and the images are function of the displacement  $d$ , so  $I_2(x, y)$  is a displaced copy of  $I_1$ , i.e.,  $I_2(x, y) = I_1(x + u, y + v)$ . With this in mind, if we want to minimize the discrepancy between the real  $I_2(x, y)$  and the proposed  $I_2$ ,  $I_1(x + u, y + v)$ , this can be carried out by applying the least squares method, and the solution to the produced system of equations yields directly  $u$  and  $v$ . [6,7]

When setting the system of differential equations,  $I_1(x + u, y + v)$  is expressed as Taylor series expanded to only 3 terms of the series (up to the linear term). The approximation works effectively because the solution is found recursively. For more information see the Chapter 3. [4,5]

### **1.2.3 Hybrid Lucas-Kanade algorithm**

The Lucas-Kanade method has the ability to incorporate an artificial intelligence algorithm to minimize the error function.[6,7,8]

The hybrid Lucas-Kanade corresponds to the original Lucas Kanade incorporating a swarm optimization algorithm. Particle swarm algorithm optimizes a problem by having a population of candidate solutions, here particles, and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and displacement. Each particle displacement is influenced by its local best-known position, and to be guided toward the best known position, it is updated taking into account the displacements of surrounding particles. This process is expected to move the swarm of particle displacements towards the least-error solution.

# Bibliography

- [1] M. Raffel, C. Willert and J. Kompenhans, *Particle Image Veocimetry A Practical Guide*. (Springer, 1998).
- [2] R. Adrian and J. Westerweel, *Particle Image Velocimetry*. ( Cambridge University Press, 2011).
- [3] J. Westerweel, *Digital Particle Image Velocimetry Theory and Application*. (Delft University Press, 1993).
- [4] B. Lucas and T. Kanade (1981), “*An iterative image registration technique with an application to stereo vision.*” Worshop, Canada (Proceedings of Imaging Understanding 1981)
- [5] S. Birchfield , “*Derivation of Kanade-Lucas-Tomasi tracking equation*” (Private communication, 1997)
- [6] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. (Addison-Wesley,1989).
- [7] M. Mitchell, *An Introduction to Genetic Algorithms*. (Bradford, 1998).
- [8] V. Gazi and K. Passimo, *Swarm Stability and Optimization*. (Springer,2011).

# Chapter 2

## Particle image velocimetry by correlation of images

### 2.1 Introduction

The correlation method calculates the displacement  $\vec{d}$  of particles using two images. The first image  $I_1$  is at time  $t$ , while the second Image  $I_2$  is at time  $t + dt$ , where  $dt$  is a small portion of time.[1,2]

The method works with images in gray scale. Normally the scale is from 0 to 255, where 0 is associated with the lowest value of light intensity and 255 represents the highest intensity value. In this way, the white pixels in the images indicate particles and the black background represents void. [3,2]

The main objective of the correlation method is to obtain the displacement of every particle using the two images. Recognizing every particle in the two images and then calculating the displacement is a really complex problem since an image can contain thousands of similar particles. We can divide the image in small subimages, with size of  $16 \times 16$  pixels, for example, and work with these subimages. Now the idea is to correlate the subimages of  $I_1$  with the corresponding subimages of  $I_2$ . By the correlation result, we can determine the displacement of the particle; this will be discussed in detail along this Chapter. In Section 2.2 it is explained how the correlation statistically works. In Section 2.3 the correlation PIV method is described from the point of view of the Fourier transform; Section 2.4 describes how to find the peak of correlation and in Section 2.5 the results of a preliminary simulation are presented.

## 2.2 Mathematical background

### 2.2.1 Statistical interpretation of correlation

Rather than estimating the displacement  $\vec{d}$  analytically the method of choice is to locally find the best match between the subimage  $I_{s1}$  and the subimage  $I_{s2}$  in a statistical way. In the introduction of this chapter we mentioned that it is complex to determinate  $\vec{d}$  for every particle. So the analytical estimation is avoided and is replaced by a statistical computation. Estimating the displacement  $\vec{d}$  statistically can be done through the use of the discrete cross-correlation function.[1,4]

To understand the statistical background of the cross-correlation function, Equation 2.1, is necessary to recall it.

$$R(x, y) = \sum_{i=-k}^k \sum_{j=-l}^l I_{s1}(i, j) I_{s2}(i + x, j + y) \quad (2.1)$$

$R(x, y)$  is the measurement of correlation of the two samples at the current pixel of index  $(x, y)$ . The size of the neighborhood is determined by  $k$  and  $l$ . The variables  $I_{s1}$  and  $I_{s2}$  are the subimages (intensity values) extracted from the original image  $I_1$  and image  $I_2$  respectively.  $I_{s2}$  is larger than  $I_{s1}$ , essentially the template  $I_{s1}$  is linearly shifted around the sample  $I_{s2}$  without extending over the edges of  $I_{s2}$ .

This expression is called the discrete cross-correlation. The cross-correlation function statistically measures the degree of match between the two samples for a given shift. The highest value in the correlation plane can be used as a direct estimate of the particle displacement.[1,5]

This is shown graphically in Figure 2.1.

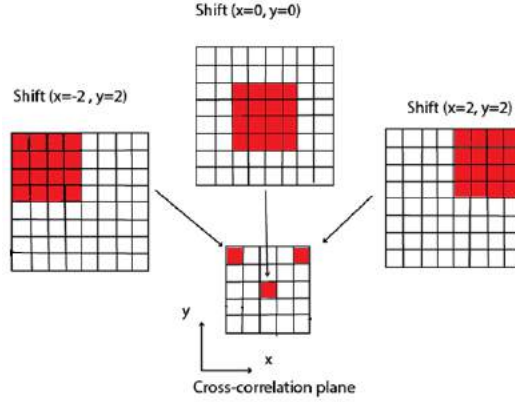


Figure 2.1: Example of the formation of the correlation plane by direct cross correlation: here a  $4 \times 4$  pix is correlated with a larger  $8 \times 8$  pix sample to produce  $5 \times 5$  pix correlation plane.

For shift values at which particles align each other.[1]

The cross-correlation will give a peak in the correlation plane when the particle in subimage  $I_{s1}$  finds itself in subimage  $I_{s2}$ . This happens since the cross-correlation is just a sum of the products of the intensities between the two subimages. The correlation of a particle and the void is zero.

Determining the coordinates of the peak of correlation, yields directly the components of the corresponding subimage displacement. If in the subimages there are more particles with negligible intensity, the information of their displacement will be lost, since the peak of the correlation is just formed by the correlation of the brightest particles.

### 2.2.2 Fourier interpretation of correlation

The Fourier transform has many properties that are useful, like: Linearity, time shifting, scaling factor, etc. But there is a crucial property of the Fourier transform for our purpose, the correlation property. To explain it, first we must need to recall what the Fourier transform is.

The Fourier transform is defined by the following equation

$$F(\omega) = \int_{-\infty}^{+\infty} f(x)e^{-2\pi i\omega x} dx \quad (2.2)$$

Where  $f(x)$  is the function to be transformed,  $F(\omega)$  is the transformation of

the function  $f(x)$ , the exponential is the kernel of the Fourier transform,  $i$  is the imaginary number and  $x$  is the spatial variable while  $\omega$  is the variable of frequency.

So the Fourier transform of a function  $f(x)$  is just the decomposition of the signal in its frequency components. The Fourier transform is just the Fourier complex series applied to a function that has an infinity period.

There is an anti-transformation operation, which has the following form.

$$f(x) = \int_{-\infty}^{+\infty} F(\omega) e^{2\pi i \omega x} d\omega \quad (2.3)$$

On the other side, the convolution in one dimension is defined as,

$$f(x) * g(x) = \int_{-\infty}^{+\infty} f(x - x') g(x') dx' \quad (2.4)$$

The correlation is almost the same as the convolution but it has a difference, a minus sign; this difference makes a different concept. The correlation is represented in Equation (2.1).

$$f(x) \circ g(x) = \int_{-\infty}^{+\infty} f(x + x') g(x') dx' \quad (2.5)$$

Now when we apply the Fourier transform to the correlation, we will prove the result is given by the product of the Fourier transforms of both functions.

First we define the two functions to have the following forms in terms of their Fourier transform; respectively:

$$f(x, y) = \mathcal{F}^{-1}[F(\omega_x, \omega_y)] = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} F(\omega_x, \omega_y) e^{2\pi i (x\omega_x + y\omega_y)} d\omega_x d\omega_y \quad (2.6)$$

$$g(x, y) = \mathcal{F}^{-1}[G(\omega_x, \omega_y)] = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} G(\omega_x, \omega_y) e^{2\pi i (x\omega_x + y\omega_y)} d\omega_x d\omega_y \quad (2.7)$$

On the other hand, the correlation for these functions of two dimensions has the following form,

$$f(x, y) \circ g(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x + x', y + y') g(x', y') dx' dy' \quad (2.8)$$

Replacing Equation (2.6) in Equation (2.8) we get Equation (2.9).



$$f(x, y) \circ g(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} g(x', y') \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} F(\omega_x, \omega_y) e^{2\pi i[\omega_x(x+x')+\omega_y(y+y')]} \\ \times dw_x dw_y dx' dy' \quad (2.9)$$

Reordering the Equation (2.9) we have:

$$f(x, y) \circ g(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} F(\omega_x, \omega_y) dw_x dw_y \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} g(x', y') e^{2\pi i[\omega_x(x+x')+\omega_y(y+y')]} \\ \times dx' dy' \quad (2.10)$$

In Equation (2.10) the kernel has a positive sign; the integration inside the brackets will give the conjugate of the Fourier transform of  $g(x', y')$ .

$$f(x, y) \circ g(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} F(\omega_x, \omega_y) G^*(\omega_x, \omega_y) e^{2\pi i[w_x(x)+w_y(y)]} dw_x dw_y \quad (2.11)$$

Equation (2.11) can be represented by the following form:

$$f(x, y) \circ g(x, y) = \mathcal{F}^{-1}[F(\omega_x, \omega_y) G^*(\omega_x, \omega_y)] \quad (2.12)$$

Therefore, the transform of correlation of the images  $g(x, y)$  and  $f(x, y)$  is

$$\mathcal{F}[f(x, y) \circ g(x, y)] = F(\omega_x, \omega_y) G^*(\omega_x, \omega_y) \quad (2.13)$$

The transform of the correlation of the functions  $g(x, y)$  and  $f(x, y)$  is just the multiplication of the Fourier transform of the function  $f(x, y)$  with the conjugate of the Fourier transform of the function  $g(x, y)$ .

This demonstration is for continuous data, but when integrals are replaced by summations it can be applied to discrete data as well. Then, this equation can be used for our purposes.

## 2.3 PIV evaluation

### 2.3.1 PIV step by step

This Section is the core of the chapter; it describes how to evaluate the PIV with digital data. The digital PIV evaluation consists in working with stored computer

images that are represented in a matrix form where every element of the matrix is associated with a value of intensity (as earlier mentioned in this Chapter).

Working with this matrix (image) in software platforms is a great advantage as the speed of the computers is getting higher. Also the problem becomes a digital image processing problem [4,3].

The steps for the digital PIV evaluation algorithm are the following:

**Step 1** Get the two Images  $I_1$  and  $I_2$  and store them on a computer in a gray color scale.

**Step 2** It is recommended to filter the images with a mean neighbor filter. The neighbor filter is a filter that computes the average of every pixel with its closest neighbors pixels; it helps to reduce the amount of noise in the images.[2,10]

**Step 3** Divide the averaged images in small samples. Remember that the correlation can be done by two ways, by its definition, Equation (2.1), or by the Fourier transform. If the correlation is done by its definition, the samples of image  $I_2$  need to be larger in size than the samples of image  $I_1$  (see Section 2.2.1). If the correlation is carried out by the Fourier transform the samples of the image  $I_1$  and the image  $I_2$  must have the same size (see Section 2.2.2).

**Step 4 Using the correlation by its definition:** Calculate the correlation for every pair of samples using Equation (2.1).

**Step 4' Using the Fourier correlation:** Apply the FFT to every sample of the images  $I_1$  and  $I_2$ .

**Step 5' Using the Fourier correlation:** Compute the product of the Fourier transform of the subimage  $I_{sf1}$  with the corresponding conjugate of the Fourier transform of subimage  $I_{sf2}$ . Compute this operation for every pair of samples. Apply the inverse FFT.

**Step 5 Using the Fourier correlation:** Find the coordinates of the correlation peak which correspond directly to the displacement. The displacement is given by the distance from the center of the correlation peak to the center of the correlation image. (see Section 2.4).

### 2.3.2 Observations when FFT is applied to PIV

The use of the FFT for the computation of the cross-correlation plane has a number of characteristics whose effects are important. Next, we list the main factors which influence the performance of the Fourier method.

**Fixed sample size:** The FFT computational efficiency is mainly derived by recursively implementing a symmetry property between the even and odd coefficients of the discrete Fourier transform. In the case of digital PIV evaluation of the data

is supposed to have a base-2 dimension.[1]

**Periodicity of the data:** The definition of the Fourier transform is an integral over a domain extending from minus infinity to plus infinity. In practice when the FFT is computed over finite domains, it is assumed the data is periodic, that is, the images continuously repeat themselves in horizontal and vertical directions from minus infinity to plus infinity. [1,4]

**Aliasing:** Since the data input is supposed to be periodic, the FFT correlation gets periodic correlation data. If the data of the image is length  $N$  and the sampling period is less than 2 pix, then the correlation peak will be folded back into the correlation plane and the resulting data appears on the opposite side of the correlation image.[1]

**Bias error:** Another effect of the periodicity of the data is that the correlation estimates are biased. This has to do with the geometrical shape of the window, which is finite and it does have a Fourier transform as well, which weighs the Fourier transform of the data. [1]

### 2.3.3 Observations in PIV evaluation

There are some occasions when the digital PIV fails, the situations are listed in the following list.

**Extremely noisy images:** When the images are corrupted by noise, it is difficult for the method to yield accurate results. So it is really important to filter the images before hand or if necessary, repeat the capture of images.[2,10]

**High particle density images:** When the image has an extremely large amount of particles and the particles overlap each other, the correlation will give wrong values of the displacement estimations.[1,4,6]

**High speed particles:** The code gets wrong results when the particle is inside the sample of image  $I_1$  but is not inside the corresponding sample of image  $I_2$ . This happens when the fluid speed is too large and  $dt$  is not short enough in comparison with the time that it takes the particles to cross the sample of image  $I_2$ . [7,8,9]

## 2.4 Peak detection

The position of the correlation peak is very important in the digital PIV evaluation; this peak can be measured to subpixel accuracy. Since the input data is discretized, the correlation value exists only for integral shifts. The peak value permits only an accuracy of 1/2 pixel for the displacement.

To increase the accuracy of the measured displacement, there are a variety of methods. Centroids, which are defined by the ratio between the first order moment and the zeroeth order moment, where the larger the quantity of values contributing to the calculation of the moment, the better. The most common of these methods is called three-point estimator. It consists of only three neighbor values to estimate a component of displacement.

The three-point estimator method consists in 3 steps, the steps are described below.

**Step 1** Scan the correlation peak  $R(i, j)$  and store its integer coordinates  $(i, j)$ .

**Step 2** Extract the neighboring four correlation values:  $R(i - 1, j)$ ,  $R(i, j - 1)$ ,  $R(i + 1, j)$  and  $R(i, j + 1)$ .

**Step 3** Use three points in each direction to apply the three point estimator, generally a peak centroid or a fitting to a Gaussian curve.

The equation for the centroid and the Gaussian curve are estimated as,

Peak centroid:

$$x_0 = \frac{(i - 1)R_{(i-1,j)} + iR_{(i,j)} + (i + 1)R_{(i+1,j)}}{R_{(i-1,j)} + R_{(i,j)} + R_{(i+1,j)}} \quad (2.14)$$

$$y_0 = \frac{(j - 1)R_{(i,j-1)} + jR_{(i,j)} + (j + 1)R_{(i,j+1)}}{R_{(i,j-1)} + R_{(i,j)} + R_{(i,j+1)}} \quad (2.15)$$

Gaussian peak fit:

$$x_0 = i + \frac{\ln R_{(i-1,j)} - \ln R_{(i+1,j)}}{2\ln R_{i-1,j} - 4\ln R_{i,j} + 2\ln R_{(i+1,j)}} \quad (2.16)$$

$$y_0 = j + \frac{\ln R_{(i,j-1)} - \ln R_{(i,j+1)}}{2\ln R_{i,j-1} - 4\ln R_{i,j} + 2\ln R_{(i,j+1)}} \quad (2.17)$$

These four equations give the coordinates in  $x$  and  $y$  for the peak centroid and the Gaussian peak, respectively. [1]

## 2.5 Simulations and results

In this Section, by a typical example, it is described how the simulations are done in this work.

### 2.5.1 Simulations

The artificial particle image generator is a `Matlab` code that generates images with artificial particles with known parameters: shape, diameter, dynamic range, spatial density and image bit depth. The artificial particle image generator presented here works with gray scale images and individual particles described by a Gaussian intensity profile,

$$E(x, y) = E_0 e^{\frac{-8[(x-x_0)^2 + (y-y_0)^2]}{d_r^2}} \quad (2.18)$$

$E(x, y)$  is the intensity at point  $(x, y)$ . The center of the particles is located at  $(x_0, y_0)$  with a peak of intensity of  $E_0$ , and  $d_r$  is the diameter of the particle.

The code first receives the positions of the center of every artificial particle, then with Equation (2.18) it renders the intensity of every particle along the image. As it can be seen in Equation (2.18) the intensity turns to zero for a position far from the center of the particle, so the next step is to sum all the plots in one image (matrix), like a superposition of waves. The result is a single image that contains the corresponding intensity of every artificial particle.

The above description explains how the artificial particle image generator code creates the initial image  $I_1$ ; now to get the second image  $I_2$ , the particle image generator code moves the center of every artificial particle with a previously known displacement. Then the code repeats the same steps that generated the first image  $I_1$ .

The following pictures are typical images  $I_1$  and  $I_2$  that are generated by the artificial particle image generator code.

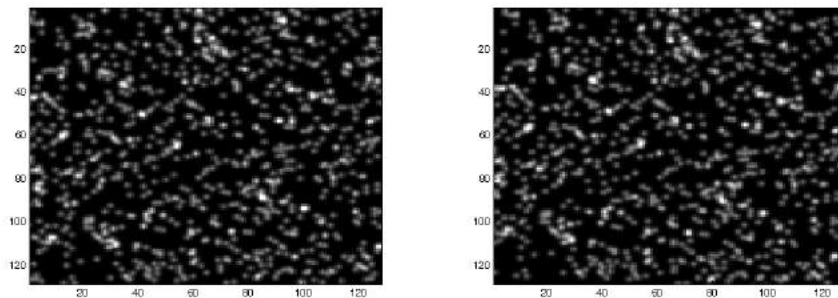


Figure 2.2: Typical PIV images, without and with displacement.

### 2.5.2 Results with the artificial particle image generator

Several tests are performed; the tests varied in the size of the image, the density of the particles, the size of the samples and the displacement supposed by the artificial particle image generator.

All the tests performed have the particularity that the displacement may change from one test to another but is equal for every particle. This is because it is necessary to realize if the code is working correctly. Since one image may contain many particles, sometimes thousands of them, the testers will not recognize if the values of the displacement computed by the code are correct if the displacement  $d$  is different for every particle. The code behaves very well for all the samples, the values that the code yields for the displacement are correct, sometimes changing a little bite (less than one pix) since the code computes the peak detection using the centroid peak Equation (2.14 and 2.15).

An example of these tests assumes two images  $I_1$  and  $I_2$  of size  $1600 \times 1600$  pix. Images  $I_1$  and  $I_2$  have 10000 artificial particles with diameter of one pixel. The displacement given to  $I_2$  particles is 3 pix to the right in the horizontal direction and 4 pix in the vertical direction. So every particle has a displacement  $d$  of 5 pix. The algorithm divides the two images in 160 pairs of subimages of  $16 \times 16$  pixels. The subimages are correlated by Fourier transform and part of a result is shown in Figure 2.3.

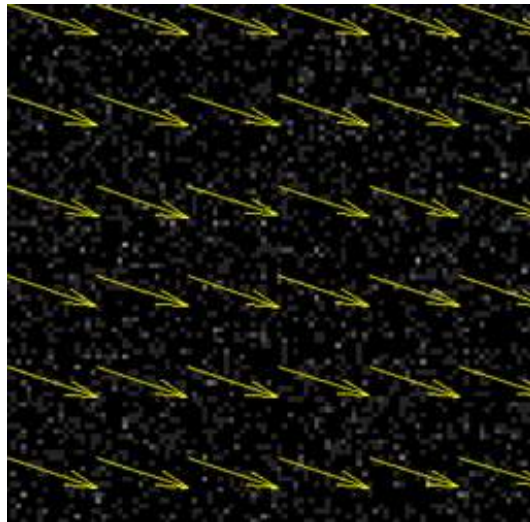


Figure 2.3: Displacement vectors obtained in a typical test performed with a constant displacement.

The results of the displacement of every subwindow of the initial test, is show in Table 2.1.

Table 2.1: Displacements in pix

u	3	3.01	2.99	3	3	3	3.02	3	3	3	2.98	3	3	3	2.98	3
v	3.97	4	4	4	4.02	4.05	4	4	4	4	4	4	4	4	4	4

This example serves to realize the software is working properly; a more detailed analysis of this algorithm is presented in Chapter 5 along with a comparison with other codes.

# Bibliography

- [1] M. Raffel, C. Willert and J. Kompenhans, *Particle Image Veocimetry A Practical Guide*. (Springer, 1998).
- [2] B. Jahne, *Digital Image Processing*. (Springer, 2005).
- [3] W. Lu and A. Antoniou, *Two-Dimensional Digital Filters*. (Marcel Dekker, Inc., 1992).
- [4] R. Adrian and J. Westerweel, *Particle Image Velocimetry*. ( Cambridge University Press, 2011).
- [5] J. Westerweel, *Digital Particle Image Velocimetry Theory and Application*. (Delft University Press, 1993).
- [6] M. Jahanmiri, *Particle Image Velocimetry: Fundamentals and Its Applications: Fundamentals and modern techniques of particle image velocimetry for fluid dynamics research*. (Lap Lambert, 2012).
- [7] M. Boas, *Mathematical Methods in the Physical Science*. (Wiley, 2005).
- [8] A. Smits and T. Lim, *Flow Visualization: Techniques and Examples*. (Imperial Collage Press, 2012).
- [9] A. Schroder and C. Willert, *Particle Image Velocimetry: New Developments and Recent Applications*. (Springer, 2008).
- [10] R. Gonzalez and R. Woods, *Digital Image Processing*. (Prentice Hall, 2007)



# Chapter 3

## The Lucas-Kanade algorithm

### 3.1 Introduction

Similar to the correlation method, the Lucas-Kanade algorithm has the goal; to calculate the displacement of particles using 2 related images. The first image  $I_1(x, y)$  is at time  $t$ , while the second Image  $I_2(x, y)$  is at time  $t + dt$ , where  $dt$  is the time between the images. [1,2]

The Lucas-Kanade algorithm uses a system of differential equations to find the displacement components  $u$  and  $v$ . [3,4]

In Section 3.2 it is explained the mathematical background of the Lucas-Kanade algorithm, in Section 3.3 we describe the Lucas-Kanade algorithm step by step along with some highlights, and in Section 3.4 the results of the simulation by `Matlab` are included.

### 3.2 Mathematical background

#### 3.2.1 Suppositions

The goal of the Lucas-Kanade algorithm is to find the components of displacement  $u$  and  $v$ . To do this, the code has some suppositions. The first image  $I_1$  is similar to the second image  $I_2$ . [1,5] Also,  $u$  and  $v$  are relatively small, and  $I_1(x + u, y + v)$  must be similar to  $I_2(x, y)$ . The objective of the code is to find  $u$  and  $v$  such that the difference between  $I_1(x + u, y + v)$  and  $I_2(x, y)$  is a minimum. [1,2,3]

### 3.2.2 Theoy

Equation (3.1) is the squared *Error* over a neighborhood of size  $N \times M$  pixels, where  $I_1(x + u, y + v)$  is the first image evaluated at the displacement components  $u$  and  $v$  and  $I_2(x, y)$  is the second image,[6]

$$Error^2 = \sum_{i=1}^N \sum_{j=1}^M [I_1(x + i + u, y + j + v) - I_2(x + i, y + j)]^2 \quad (3.1)$$

To minimize this Equation, it is better to express  $I_1(x + u, y + v)$  in a Taylor series (we take the first three terms),

$$I_1(x + u, y + v) = I_1(x, y) + u \frac{\partial I_1(x, y)}{\partial x} + v \frac{\partial I_1(x, y)}{\partial y} \quad (3.2)$$

Replacing Equation (3.2) in Equation (3.1), we get

$$Error^2 = \sum_{i=1}^N \sum_{j=1}^M [I_1(x+i, y+j) + u \frac{\partial I_1(x+i, y+j)}{\partial x} + v \frac{\partial I_1(x+i, y+j)}{\partial y} - I_2(x+i, y+j)]^2 \quad (3.3)$$

To minimize the function, we apply a derivative respect to  $u$  and  $v$ , and equalize to zero,

$$\begin{aligned} \frac{\partial Error}{\partial u} &= 2 \sum_{i=1}^N \sum_{j=1}^M [I_1(x+i, y+j) + u \frac{\partial I_1(x+i, y+j)}{\partial x} + v \frac{\partial I_1(x+i, y+j)}{\partial y} - I_2(x+i, y+j)] \\ &\quad \times \frac{\partial I_1(x+i, y+j)}{\partial x} = 0 \end{aligned} \quad (3.4)$$

and

$$\begin{aligned} \frac{\partial Error}{\partial v} &= 2 \sum_{i=1}^N \sum_{j=1}^M [I_1(x+i, y+j) + u \frac{\partial I_1(x+i, y+j)}{\partial x} + v \frac{\partial I_1(x+i, y+j)}{\partial y} - I_2(x+i, y+j)] \\ &\quad \times \frac{\partial I_1(x+i, y+j)}{\partial y} = 0 \end{aligned} \quad (3.5)$$

For shortening the notation, we omit the indices of the summations. Expanding the two sums,

$$\sum \sum I_1 \frac{\partial I_1}{\partial x} + u \sum \sum \left( \frac{\partial I_1}{\partial x} \right)^2 + v \sum \sum \frac{\partial I_1}{\partial x} \frac{\partial I_1}{\partial y} - \sum \sum I_2 \frac{\partial I_1}{\partial x} = 0 \quad (3.6)$$

and

$$\sum \sum I_1 \frac{\partial I_1}{\partial y} + v \sum \sum \left( \frac{\partial I_1}{\partial y} \right)^2 + u \sum \sum \frac{\partial I_1}{\partial x} \frac{\partial I_1}{\partial y} - \sum \sum I_2 \frac{\partial I_1}{\partial y} = 0 \quad (3.7)$$

What we get is a system of differential equations that can be expressed in matrix form,

$$\begin{bmatrix} \sum \sum \left( \frac{\partial I_1}{\partial x} \right)^2 & \sum \sum \frac{\partial I_1}{\partial x} \frac{\partial I_1}{\partial y} \\ \sum \sum \frac{\partial I_1}{\partial x} \frac{\partial I_1}{\partial y} & \sum \sum \left( \frac{\partial I_1}{\partial y} \right)^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum \sum I_2 \frac{\partial I_1}{\partial x} - \sum \sum I_1 \frac{\partial I_1}{\partial x} \\ \sum \sum I_2 \frac{\partial I_1}{\partial y} - \sum \sum I_1 \frac{\partial I_1}{\partial y} \end{bmatrix} \quad (3.8)$$

Equation (3.8) is similar to a typical system of linear equations with the form of Equation (3.9),

$$\mathbf{Ax} = \mathbf{b} \quad (3.9)$$

To solve Equation (3.9), we multiply both sides of the equation by the inverse matrix of  $\mathbf{A}$ ,

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum \sum \left( \frac{\partial I_1}{\partial x} \right)^2 & \sum \sum \frac{\partial I_1}{\partial x} \frac{\partial I_1}{\partial y} \\ \sum \sum \frac{\partial I_1}{\partial x} \frac{\partial I_1}{\partial y} & \sum \sum \left( \frac{\partial I_1}{\partial y} \right)^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum \sum I_2 \frac{\partial I_1}{\partial x} - \sum \sum I_1 \frac{\partial I_1}{\partial x} \\ \sum \sum I_2 \frac{\partial I_1}{\partial y} - \sum \sum I_1 \frac{\partial I_1}{\partial y} \end{bmatrix} \quad (3.10)$$

The solution of system of the differential equations is  $u$  and  $v$ , this  $u$  and  $v$  are just for the neighborhood of length  $N \times M$ .

### 3.2.3 The optical flow model

In this Section it is explained an alternative form of the Lucas-Kanade algorithm, called the optical flow model. The optical flow model has a simpler notation and it is easier to program than the minimization approach described in the previous section, but both are equivalent, as it is shown next.

The same suppositions of the Lucas-Kanade algorithm are taken for the optical flow model.

- The first image  $I_1(x, y)$  is similar to the second image  $I_2(x, y)$ .
- The displacement  $u$  and  $v$  are relatively small.
- The first image  $I_1(x, y)$  evaluated at  $x + u$  and  $y + v$  is equal to  $I_2(x, y)$ ,  $I_1(x + u, y + v) = I_2(x, y)$ . [4,6]

Then, by assuming an exact differential (with small displacements) for the first image,

$$I_1(x + u, y + v, t + dt) = I_1(x, y, t) + \frac{\partial I_1}{\partial x}u + \frac{\partial I_1}{\partial y}v + \frac{\partial I_1}{\partial t}t \quad (3.11)$$

Since the displaced image is assumed to be an exact copy of the first one, then

$$\frac{\partial I_1}{\partial x}u + \frac{\partial I_1}{\partial y}v + \frac{\partial I_1}{\partial t}t = 0 \quad (3.12)$$

or

$$\frac{\partial I_1}{\partial x}u + \frac{\partial I_1}{\partial y}v = -\frac{\partial I_1}{\partial t}t \quad (3.13)$$

The derivative with respect to time is just the difference between the first and the second image. Taking this into consideration, from Equation (3.13), we have,

$$\frac{\partial I_1}{\partial x}u + \frac{\partial I_1}{\partial y}v = -(I_2 - I_1) \quad (3.14)$$

Equation (3.14) is expressed for a single pixel; for a neighborhood of  $n$  pixels, the following matrix form is obtained,

$$\begin{bmatrix} \frac{\partial I_{1_1}}{\partial x} & \frac{\partial I_{1_1}}{\partial y} \\ \vdots & \vdots \\ \frac{\partial I_{1_n}}{\partial x} & \frac{\partial I_{1_n}}{\partial y} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} I_{1_1} - I_{2_1} \\ \vdots \\ I_{1_n} - I_{2_n} \end{bmatrix} \quad (3.15)$$

Equation (3.15) is similar to,

$$\mathbf{A}_1 \mathbf{x}_1 = \mathbf{b}_1 \quad (3.16)$$

The system can be solved by using the pseudoinverse of  $\mathbf{A}_1$ . First, we need to multiply both sides of the equation by  $(\mathbf{A}_1)^T$  to get a quadratic matrix  $\mathbf{A}_1(\mathbf{A}_1)^T$  on the left side of the equation:

$$\mathbf{A}_1^T \mathbf{A}_1 \mathbf{x}_1 = \mathbf{A}_1^T \mathbf{b}_1 \quad (3.17)$$

Then, multiply both sides by the inverse of  $\mathbf{A}_1(\mathbf{A}_1)^T$  and the system is solved,

$$\mathbf{x}_1 = (\mathbf{A}_1^T \mathbf{A}_1)^{-1} \mathbf{A}_1^T \mathbf{b}_1 \quad (3.18)$$

We can express  $(\mathbf{A}_1^T \mathbf{A}_1)$  as

$$\begin{bmatrix} \frac{\partial I_{1_1}}{\partial x} & \cdots & \frac{\partial I_{1_n}}{\partial x} \\ \frac{\partial I_{1_1}}{\partial y} & \cdots & \frac{\partial I_{1_n}}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial I_{1_1}}{\partial x} & \frac{\partial I_{1_1}}{\partial y} \\ \vdots & \vdots \\ \frac{\partial I_{1_n}}{\partial x} & \frac{\partial I_{1_n}}{\partial y} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n (\frac{\partial I_{1_i}}{\partial x})^2 & \sum_{i=1}^n \frac{\partial I_{1_i}}{\partial x} \frac{\partial I_{1_i}}{\partial y} \\ \sum_{i=1}^n \frac{\partial I_{1_i}}{\partial x} \frac{\partial I_{1_i}}{\partial y} & \sum_{i=1}^n (\frac{\partial I_{1_i}}{\partial y})^2 \end{bmatrix} \quad (3.19)$$

which has the same form as the first term of the right-hand side of Equation (3.10). Besides,  $(\mathbf{A}_1)^T \mathbf{b}_1$  yields

$$\begin{bmatrix} \frac{\partial I_{1_1}}{\partial x} & \cdots & \frac{\partial I_{1_n}}{\partial x} \\ \frac{\partial I_{1_1}}{\partial y} & \cdots & \frac{\partial I_{1_n}}{\partial y} \end{bmatrix} \begin{bmatrix} I_{1_1} - I_{2_1} \\ \vdots \\ I_{1_n} - I_{2_n} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n \frac{\partial I_{1_i}}{\partial x} (I_{1_i} - I_{2_i}) \\ \sum_{i=1}^n \frac{\partial I_{1_i}}{\partial y} (I_{1_i} - I_{2_i}) \end{bmatrix} \quad (3.20)$$

which corresponds to the second term of Equation (3.10). The optical flow model is equivalent to the Lucas-Kanade algorithm. However, Equation (3.10) has a double sum, because the neighborhood of equation is  $N \times M$ , and the terms of equation Equation (3.19) and Equation (3.20) the neighborhood is the same but it uses a one-dimensional path of length  $n = N \times M$ .

## 3.3 Evaluation of the Lucas-Kanade algorithm

### 3.3.1 Lucas-Kanade algorithm step by step

It is important to see that Equation (3.10) is an approximate solution. To get the best possible solution, the mathematical background explained in Section (3.2) must be implemented recursively. This recursive approach compensates the use of only the linear terms in the Taylor expansion. Thus, the first iteration gives a solution of the type  $I_1(x+u, y+v)$ ; then we apply the same process considering the previous answer to find a new  $I_1(x+u, y+v)$ , and so on.[7,8] This implies that the subimage composed by the  $M \times N$  particles must be deformed according to the calculated displacement, and this deformed subimage is the one used in the next iteration of the algorithm.

---

**Algorithm 1 Lucas-Kanade**

---

- 1: Get the two subimages  $I_1(x, y)$  and  $I_2(x, y)$ .
  - 2: Take the first derivative of the first image  $I_1(x, y)$  with respect to  $x$  and  $y$ .
  - 3: Obtain the summations implied in Equation (3.10).
  - 4: Calculate  $u$  and  $v$ .
  - 5: With the new  $u$  and  $v$ , interpolate  $I_1(x, y)$  to get  $I_1(x + u, y + v)$ .
  - 6: The  $I_1(x + u, y + v)$  becomes the new  $I_1(x, y)$ .
  - 7: Go to step 2 and repeat the steps until the difference in displacement for two consecutive iterations is less than a certain threshold.
- 

### 3.3.2 Highlights of the Lucas-Kanade algorithm

There are some occasions when the Lucas-Kanade algorithm fails; the situations are listed in the following list.

**Noisy images.** When the images are corrupted by noise or are saturated, it is difficult for the method to yield accurate results, since pixel information gets lost. This problem can be alleviated by preprocessing the images.[8]

**Images with low particle density.** When the image has less than 10 particles per subimage, the derivatives of  $I_1(x, y)$  may be zero, and equation 3.10 cannot give a result.[1]

**Particles with relatively large speed.** The algorithm gets misleading results when the particle is inside the neighborhood of  $I_1(x, y)$  but not inside the corresponding neighborhood of  $I_2(x, y)$ . This happens when the fluid reaches high speeds. The problem is that  $I_1(x, y)$  is no longer similar to  $I_2(x, y)$ . [1,6,7]

## 3.4 Simulations and results

Two synthetic particle images are generated, as described in the previous Chapter, and the Lucas-Kanade algorithm is applied. The images are  $128 \times 128$  pix with 1000 particles. The particle diameter is 1 pix. The displacement  $\vec{d}$  for every particle is constant, with components  $u = 1$  and  $v = 1$ . The size of the subimages is  $10 \times 10$  pix. The code processes all this information and computes the displacement. This calculated displacement is then compared with the given displacement.

Also the code draws the vectors of the displacement. In the following Figure it is shown the optical flow map of the above-mentioned example, for three iterations.

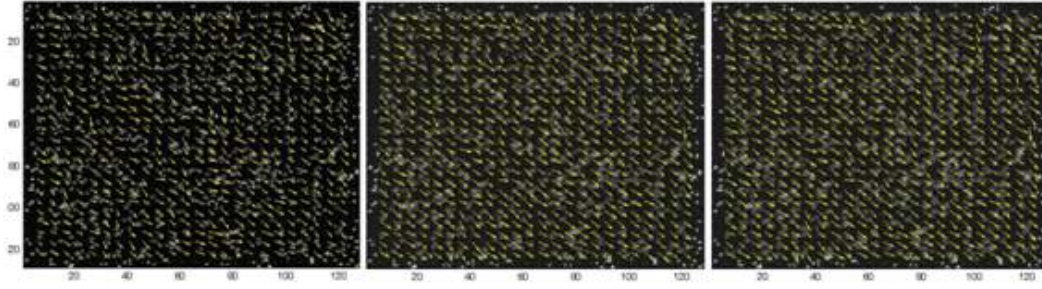


Figure 3.1: A typical displacement map obtained by the Lucas-Kanade code. Three iterations are shown.

In the next plot, Figure 3.2, we show the convergence of the values of displacement for three different pixels selected from Figure 3.1.

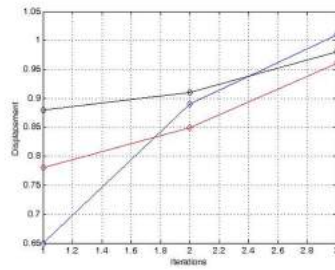


Figure 3.2: The plot of the displacement of three pixels as the number of iterations increases.

# Bibliography

- [1] B. Lucas and T. Kanade, *An iterative image registration technique with an application to stereo vision*. Workshop, Canada (Proceedings of Imaging Understanding, 1981).
- [2] S. Birchfield, *Derivation of Kanade-Lucas-Tomasi tracking equation*. Private communication, (1997).
- [3] B. Lucas, *Generalized Image Matching by the Method of Differences*. doctoral dissertation, (Carnegie-Mellon University, 1984).
- [4] J. Bouguet, *Pyramidal implementation of the affine Lucas-Kanade feature tracker description of the algorithm*. (Intel Corporation, 2001).
- [5] R. Adrian and J. Westerweel, *Particle Image Velocimetry*. (Cambridge University Press, 2011).
- [6] A. Smits and T. Lim, *Flow Visualization: Techniques and Examples*. (Imperial Collage Press, 2012).
- [7] M. Raffel, C. Willert and J. Kompenhans, *Particle Image Veocimetry: A Practical Guide*. (Springer, 1998).
- [8] B. Jahne, *Digital Image Processing*. (Springer, 2005).



# Chapter 4

## Artificial intelligence applied to the Lucas-Kanade algorithm

### 4.1 Introduction

The main idea of this Chapter is to describe the addition of an artificial intelligence algorithm to the Lucas-Kanade routine. This combination is called the hybrid method.

In Section 4.2 we give a brief description of the artificial intelligence algorithm. Then, in Section 4.3, the so-called particle swarm optimization algorithm is included. In Section 4.4, a justification of the particle swarm optimization method is presented. And finally in Section 4.5, the `Matlab` code of the hybrid algorithm is implemented and described.

### 4.2 Introduction to artificial intelligence

Artificial intelligence is the intelligence shown by software or machines; also, it is an academic field of study whose objective is to create intelligence. Some books describe this field as the study and design of intelligent agents, where an intelligent agent is a system that perceives its environment and takes actions that maximize its chance of success.[1,2,3]

There are many types of intelligent agents. The agents are just algorithms with the ability to perceive their environment and take decisions; in most of the codes, these decisions are taken according to processes that have deterministic and random steps [3,4]. Many of these algorithms are inspired by nature and they imitate how

nature solves its problems. [5,6,7]

In this work just one type of these intelligence algorithms is described, the particle swarm optimization algorithm.

## 4.3 The particle swarm optimization algorithm

### 4.3.1 Theory

This algorithm uses populations of solutions and iterations to find the best answer in an optimization problem. The swarm algorithm is based on the behavior of the populations of birds, when the crew is searching food; in this context the food means the solution in the optimization problem.[7,8,10]

In the particle swarm optimization, there is a small population searching the maximum value or the minimum value of a function; that function is the fit test solution. The algorithm starts evaluating which particle is in the best position  $\mathbf{x}(\mathbf{t})$  in a certain function at time  $\mathbf{t}$  (the best position refers to the position with the minimum or maximum value in the population, depending if the code is searching the minimum or the maximum value of the function). Then, the change of each particle is determined. The next step is to obtain the position of all particles for the next iteration  $\mathbf{t}+1$ . The function of change depends on the current position of the particle. Likewise, the function of position depends on the change function. In this way all the birds in the population have communication with each other. Therefore, when a bird finds food (which is equivalent to the maximum of the function), it attracts other birds to its position and in the following iteration the other birds will change their directions and will fly to the food, which means that the code converges to the fittest solution. In every iteration, the current positions are evaluated, so the leader of the crew (the bird with the best position) is changed when is required.[8,9]

So the algorithm has the following form:

---

**Algorithm 2 The particle swarm optimization algorithm**

---

- 1: Initialize the population of particles with random positions.
- 2: Evaluate the positions of the population with a fit test.
- 3: Compare the progress of the population in the current iteration; compare the performance of every bird/particle with its own historical record and find the best of all.
- 4: With the position of the best particle, compute the change speed of every bird/particle by [4,5]

$$\mathcal{V}_i(t) = \mathcal{V}_i(t-1) + p_1[x_{best_i}(t) - x_i(t-1)] + p_2[x_{bestofall}(t) - x_i(t-1)] \quad (4.1)$$

where  $p_1$  and  $p_2$  are two random numbers within 0 and 1,  $\mathcal{V}_i$  is the velocity of bird  $i$ ,  $x_i$  is the position of bird  $i$ ,  $x_{best_i}$  is the best historical position of  $i$  and  $x_{bestofall}$  is the best historical position of the population.

- 5: With the new velocities compute the new corresponding positions by

$$x_i(t) = \mathcal{V}_i(t) + x_i(t-1) \quad (4.2)$$

- 6: Go to Step 2 and repeat until all the iterations are done.
- 

### 4.3.2 Implementation example

In this Section a simple example of the use of the particle swarm optimization algorithm is presented; we want to maximize the following equation,

$$y = 8x - x^2 \quad (4.3)$$

To do it, we can derive and equalize it to zero, but we will find the maximum using the particle swarm optimization algorithm; Equation (4.3) is the fit test.

First we will use a population of four birds and we will have two iterations. We start by setting all the initial velocities and initial positions to random values.

The values for  $p_1$  and  $p_2$  were selected randomly. The initial velocities are for the first bird 0.013, the second bird 0.62, the third bird 1.06 and the last bird 0.41. In the first iteration we have, Table 4.1.

Table 4.1: Firts iteration

Initial positions	Fit test	$p_1$	$p_2$	Velocity	New positions
1	7	0.55	0.21	0.58	1.58
2.2	12.76	0.65	0.11	0.79	2.99
1.7	10.71	0.22	0.32	0.60	2.30
3.7	15.91	0.98	0.85	0.41	4.11

Here the Fit test is calculated as Equation 4.3 is evaluated with the initial random positions.

It can be seen that in the second iteration all the birds are converging to the solution, 4, see Table 4.2

Table 4.2: Second iteration

Initial positions	Fit test	$p_1$	$p_2$	Velocity	New positions
1.58	10.14	0.75	0.51	1.87	3.45
2.99	14.97	0.44	0.11	0.91	3.90
2.30	13.11	0.35	0.36	1.17	3.47
4.11	15.98	0.81	0.81	0.41	4.52

For Equation (4.3), the code converges rapidly and by the second iteration, all the birds know that the food must be at position four. This procedure is really useful when the equation is complex and where an analytical solution is not possible.

## 4.4 The hybrid algorithm

In this Section it is described the implementation of the swarm algorithm into the Lucas-Kanade code.

The motivation is the following: since the Lucas-Kanade algorithm uses only three terms of the Taylor series it needs a recursive algorithm to compute good results. The idea is to help Lucas-Kanade algorithm to find the best  $u$  and  $v$ .

The primary goal of the Lucas-Kanade algorithm is to minimize the next error function,

$$Error^2 = \sum_{i=1}^N \sum_{j=1}^M [I_1(x+i+u, y+j+v) - I_2(x+i, y+j)]^2 \quad (4.4)$$

This function can be seen as the fit test function for the particle swarm algorithm. This fit function is not analytical since the images  $I_1$  and  $I_2$  are numerical matrices. This problem can be solved by recalling that image  $I_1$  is a function of  $u$  and  $v$ , and that the Lucas-Kanade algorithm, in every iteration, updates the next image  $I_1$  by an interpolation of current  $I_1$  by considering the current values of  $u$  and  $v$ . The aim of the artificial intelligence algorithm is to find the  $u$  and  $v$  that minimizes the difference by using the interpolation image generated by  $I_1$ ,  $u$  and  $v$ , and the second image  $I_2$ . Then, the fit function implemented in this code is the following expression,

$$Fit(u, v) = Interpolation(I_1, u, v) - I_2(u, v) \quad (4.5)$$

Where *Interpolation* is the interpolation function in two dimensions of  $I_1$ .

With the fit function defined, the following steps in the particle swarm optimization algorithm are as described above.

## 4.5 Description of the implemented code

The code implemented is a hybrid of the Lucas-Kanade algorithm and the particle swarm optimization algorithm.

### 4.5.1 Pseudocode of the hybrid algorithm

First, the Lucas-Kanade algorithm code gets  $u$  and  $v$ . Then, the swarm optimization code uses a scan window that extracts the neighborhoods of  $u$  and  $v$ . Following, the code uses the current  $u$  and  $v$  to find the best version of  $u$  and  $v$  that minimizes the fit function in that neighborhood (the same process is applied to all scan windows).

So the code has the following form:

---

**Algorithm 3 The hybrid Lucas-Kanade algorithm**

---

- 1: Get  $u$  and  $v$  with the Lucas-Kanade algorithm.
- 2: Extract the current  $u$  and  $v$  belonging to a neighborhood of a certain size.
- 3: Save the originals  $u$  and  $v$  in  $u_{original}$  and in  $v_{original}$ .
- 4: Initialize the bird population of  $u$  and the bird population  $v$  with random values between  $max(u)$  and  $min(u)$  and between  $max(v)$  and  $min(v)$ , respectively.
- 5: Evaluate the populations of  $u$  and  $v$  with the fit test.
- 6: Get the best value for every element of populations and construct matrices  $u_{best}$  and  $v_{best}$  with the best elements of the  $u$  and  $v$  populations, respectively.
- 7: With  $u_{best}$ ,  $v_{best}$ ,  $u_{original}$ , and  $v_{original}$ , compute the velocity of every bird/particle.

$$\mathcal{V}_{ui}(t) = V_{ui}(t-1) + p_1(u_{best}(t) - u_i(t-1)) + p_2(u_{original}(t) - u_i(t-1)) \quad (4.6)$$

$$\mathcal{V}_{vi}(t) = V_{vi}(t-1) + p_3(v_{best}(t) - v_i(t-1)) + p_4(v_{original}(t) - v_i(t-1)) \quad (4.7)$$

Here  $p_1$ ,  $p_2$ ,  $p_3$  and  $p_4$  are random numbers within 0 and 1,  $\mathcal{V}_{ui}$  is the velocity of the bird  $i$  in the horizontal direction and  $\mathcal{V}_{vi}$  in the vertical direction.

- 8: With the new velocities compute the new corresponding  $u_i$  and  $v_i$ .

$$u_i(t) = \mathcal{V}_{ui}(t) + u_i(t-1) \quad (4.8)$$

$$v_i(t) = \mathcal{V}_{vi}(t) + v_i(t-1) \quad (4.9)$$

- 9: Go to Step 2 and repeat until all iterations are done.
- 

### 4.5.2 An example of the implementation of the hybrid code

The same test taken by Lucas-Kanade algorithm of Section 3.4 now is processed by the hybrid Lucas-Kanade code. The Lucas-Kanade works with images  $I_1(x, y)$  and  $I_2(x, y)$ , and gives an initial answer of  $u(x, y)$  and  $v(x, y)$ ; these  $u(x, y)$  and  $v(x, y)$  have a matrix form in which their elements have the answers recollected for every pixel in image  $I_1(x, y)$ .

The swarm optimization algorithm uses  $u(x, y)$  and  $v(x, y)$ . These variables are used in Equation (4.5). The code scans both matrices  $u(x, y)$  and  $v(x, y)$  and extracts parts of these matrices and generate populations of birds and the listed points in Section 4.5.1 are carried out. Normally the code extracts a neighborhood of  $4 \times 4$  pix

and produces populations of five birds with fifteen iterations. Recall that  $u(x, y)$  and  $v(x, y)$  have a size of  $128 \times 128$  pix, so the swarm optimization algorithm is applied 1024 times.

When the swarm optimization algorithm ends, the matrices  $u(x, y)$  and  $v(x, y)$  are updated and should be better displacement values than the original ones; to prove this statement, in Chapter 5 the original Lucas-Kanade algorithm and the hybrid Lucas-Kanade are tested under several conditions.

In the result presented in Fig. 4.1, the swarm optimization algorithm uses a population of five birds and 15 iterations are performed.

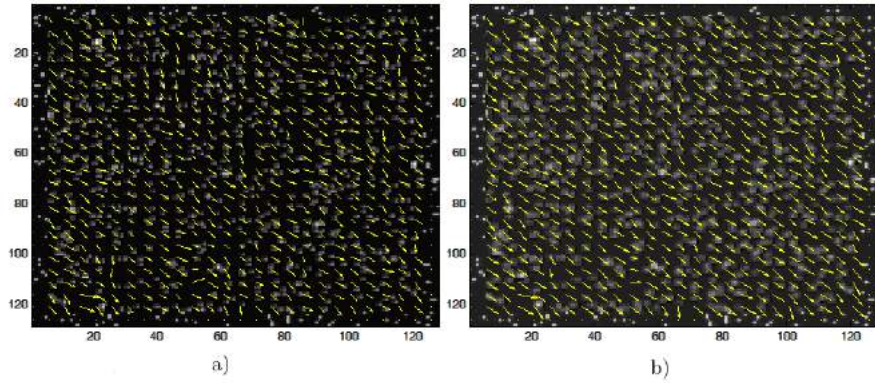


Figure 4.1: a) Lucas-Kanade algorithm result, b) hybrid Lucas-Kanade output.

# Bibliography

- [1] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. (Addison-Wesley, 1989).
- [2] M. Mitchell, *An Introduction to Genetic Algorithms*. (Bradford, 1998).
- [3] R. Haupt, *Practical Genetic Algorithms*. (Wiley, 1998).
- [4] D. Simon, *Evolutionary Optimization Algorithms*. (Wiley, 2013).
- [5] D. Floreano and C. Mattiussi, *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*. (The MIT Press, 2008).
- [6] J. Heaton, *Artificial Intelligence for Humans, Volume 1: Fundamental Algorithms*. (Independent Publishing Platform, 2013).
- [7] M. Clerc, *Particle Swarm Optimization*. (Wiley, 2006).
- [8] V. Gazi and K. Passino, *Swarm Stability and Optimization*. (Springer, 2011).
- [9] E. Chong and S. Zak, *An Introduction to Optimization*. (Wiley, 2013).
- [10] S. Russell and P. Norving, *Artificial Intelligence: A Modern Approach*. (Prentice Hall, 2009).



# Chapter 5

## Results

### 5.1 Introduction

This chapter is the core of the presented work; the three algorithms for calculating displacement maps described in Chapter 4 are tested by analyzing the effect of distinct parameters on the accuracy of the results.

In Section 5.2, a description of the evaluation method is given. Then, in Sections 5.3 and 5.4 the results related with the correlation method and the Lucas-Kanade method are presented, respectively. In these cases, synthetic particle images for numerical analysis are assumed. In Sections 5.5, the performance of the combination of the Lucas-Kanade method and an artificial intelligence algorithm is included. Likewise, in Section 5.6 the performance of the three previous algorithms is shown when applied to experimental images. In Section 5.7 we give the conclusions of this work. Finally, in Section 5.8 we list the main challenges to be analyzed in a future work.

### 5.2 Description of the numerical analysis

#### 5.2.1 Relative error in percentage

To evaluate the different methods for calculation of displacement fields we use the percentage relative error of the calculated displacement components,  $u$  and  $v$ , with respect to the corresponding given values,  $u_g$  and  $v_g$ , respectively. The error equations for each displacement component are the following.[1,2,3]

$$E_u = \sqrt{\frac{\sum_{i=1}^n (u_g^i - u^i)^2}{n}} \frac{100}{\max(u_g)} \quad (5.1)$$

$$E_v = \sqrt{\frac{\sum_{i=1}^n (v_g^i - v^i)^2}{n}} \frac{100}{\max(v_g)} \quad (5.2)$$

where  $n$  is the length of vector  $u$  and  $\max(u)$  returns the maximum value of  $u$ . This type of error is commonly used in the literature.[1]

For the numerical analysis of the methods, the following parameters are subjected to variation.

1. Size of the window, from  $16 \times 16$  pix to  $64 \times 64$  pix in the correlation method and from  $5 \times 5$  pix to the  $50 \times 50$  pix in the Lucas-Kanade algorithm.
2. Value of maximum displacement, depends of the type of displacement.
3. Diameter of the particles, from 1 pix to 3 pix.
4. Number of particles in the image, from 1000 to 3000 particles.
5. Noise level in the displacement, from 5% to 11%.
6. Contrast of the image, from 0% to 100%.
7. For the correlation method there is a test that evaluates the performance of the method when the scan window is moved half its size.
8. Number of iterations, with the Lucas-Kanade algorithm.

The test about **noise** level consists in adding a percentage of random noise to the images. This noise is computed with a matrix that has the size of the image, and its elements take random values between zero and one; then, this matrix is multiplied by the maximum value of the image, divided by one hundred, and multiplied by the percentage of level noise. Each of the images to be compared has an independent noise matrix.

The contrast of the image in turn refers to the ratio of the dynamic range of the signal to the average signal, i.e., first the  $DC$  or average value is obtained:

$$DC = 255 \frac{100 - \mathcal{U}}{100} \quad (5.3)$$

where  $\mathcal{U}$  is the percentage of contrast that will have the image; then the upper value of the image  $UP$  is computed, that is just the subtraction of the maximum value of the image and the integer part of the  $DC$ :

$$UP = 255 - \text{int}(DC) \quad (5.4)$$

In Equation (5.4)  $int()$  means to take the integer part of the argument.

Finally, a normalization of each element  $I(i, j)$  of the image  $I$  is done, where  $min(I)$  is the lowest value of the Image  $I$  and  $max(I)$  is the highest value of the image  $I$ :

$$I(i, j) = int[\frac{I(i, j) - min(I)}{max(I) - min(I)}UP] + DC \quad (5.5)$$

To explain test seven, it is important to remember that in the correlation method the images are divided by subwindows of a typical size of  $32 \times 32$  pix. Then, the Fourier algorithm is applied to every subwindow. The scan window moves with steps of length of its own size. Similarly, when the scan window moves with steps of half its size, the number of displacement vectors increases by 4 and the spatial resolution improves. In Figure 5.1, a graphic demonstration of the above explanation is given. In Figure 5.1 a) we have an example when the scan window moves in steps of length of its size. In b) the scan window moves half of its size; in this case three sub-windows are formed. This is an example only in the horizontal direction, but it applies identically for the vertical direction.

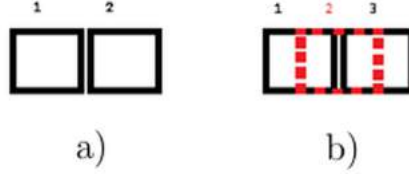


Figure 5.1: a) Scan window moves in steps of same length of its size. b) Scan window moves in steps of half its size.

For the study, first all the parameters take a-priori optimal values. Then, one parameter is varied and its influence on the accuracy of the method is obtained. Then, the optimal value of the latter parameter is taken and a second parameter is analyzed. This process is repeated until analyzing all parameters.

The a-priori values are obtained with several runs of the code with different parameters, the parameters that seems that minimizes the error.

The tests were performed for three types of displacements.

### Constant displacement

Particles experience constant displacement in the horizontal and vertical directions.

$$u = \text{const}_1 \quad v = \text{const}_2 \quad (5.6)$$

In the constant displacement tests, the second constant is bigger than the first constant,  $\text{const}_1 < \text{const}_2$ , this is because with this condition, we can compute two measurements in each test.

### Sinusoidal displacement

Particles move according to a sinusoidal function.

$$u = A \sin(x) \quad v = 0 \quad (5.7)$$

where  $A$  is the amplitude of the displacement and  $x$  is the horizontal position, see Figure 5.2.

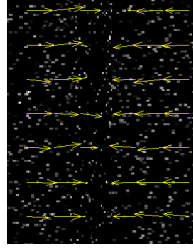


Figure 5.2: Sinusoidal displacement.

### Vortex-like displacement

The particles tend to move along concentric circles.

$$u = Ax \quad v = -Ay \quad (5.8)$$

where  $A$  is a constant;  $x$  is the horizontal position and  $y$  the vertical position, see Figure 5.3.

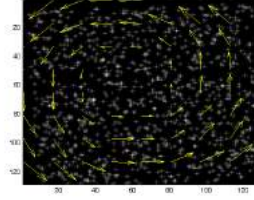


Figure 5.3: Vortex-like displacement.

Similarly, we consider three types of synthetic particles, with different profile functions:

- Gaussian particles have a exponential Gaussian profile (described in Chapter 2).[1,2,3]
- The images with binary particles assign 0 to the background and 255 to the particles, like a squared function.
- The triangular particles are similar to the binary particles but these ones contain a continuous distribution of gray levels, where the highest level of intensity is at the center of the particle and the lowest level at the edges.

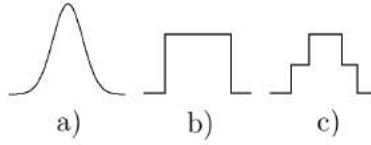


Figure 5.4: a) Gaussian profile particle. b) Binary profile particle. c) Triangular profile particle.

All tests performed in Sections 5.3, 5.4 and 5.5 assume images of  $128 \times 128$  pixels containing synthetic particles, while in Section 5.6 a test using real images is presented. Every presented result corresponds to the average of five tests with the same conditions.

The units for displacement values, size of the scan window, and the diameter of the particles are pixels [pix].

### 5.2.2 Subpixel displacement

Modeling of subpixel displacement is not straightforward for the cases of binary and triangular particles. This is due to the discrete nature of these particles, their size and their location.

In Figure 5.2 we show the process of achieving subpixel displacement for a binary particle. For example, supposing that the particle is at pixel 5 and that 1 represents the 100% of its size, of 1 pix, and that it undergoes a horizontal displacement. In Figure 5.2 a) the displacement of the particle is 1 pix, so 100% of it moves 1 pix. Now, if the displacement is of 1.8 pix, Figure 5.2 b), 20% of the particle stays in position 5 while 80% of it moves to pixel 6. Another example, Figure 5.2 c), the displacement is of 1.99 pix, therefore 99% of the particle experiences a displacement of 2 pix and 1% 1 pix.

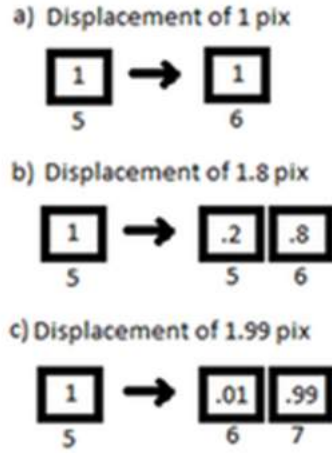


Figure 5.5: Subpixel displacement for a binary particle of diameter 1 pix.

Displacements of 0.5 pix cannot be modeled by the previous algorithm since in that case its diameter increases from 1 pix to 2 pix, and this misleads the displacement methods. For triangular particles, the same algorithm for modeling subpixel displacements can be applied.

## 5.3 Results of the correlation method

In this Section, the results obtained with the correlation method are presented. Three different types of displacement are considered. For each type of displacement, influ-

ence of the different parameters on the accuracy of the results is obtained. Images of **128x128 pix** are assumed. In the corresponding tables, values for other parameters are shown (parameter **radius** refers to the radius of the particles, the double of the radius is the diameter). Each Figure shows three plots, which correspond to different type of particle profile: **red for Gaussian, blue for binary and black for triangular**. [1,2,3]

For each testing parameter, we include a table that shows the values of the other parameters (static parameters), a figure containing the results and a description of the result.

### 5.3.1 Constant displacement for correlation

#### Size of the scan window

Table 5.1: Static parameters

# Particles	Radius [pix]	$u$ [pix]	$v$ [pix]
1000	1	2.2	2.4

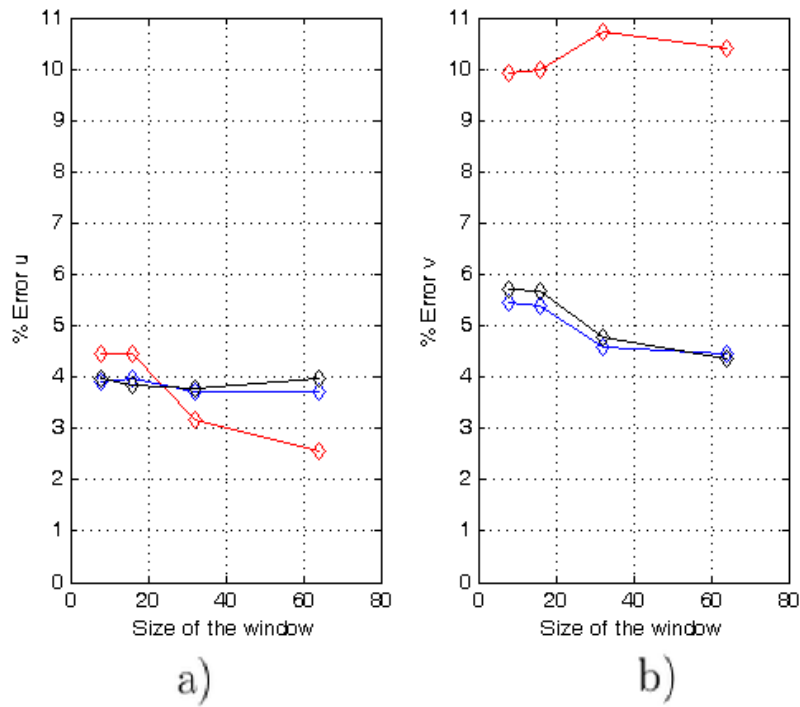


Figure 5.6: Varying the size of the window of the correlation method with constant displacement. a) Horizontal component of displacement. b) Vertical component of displacement. Red plot for Gaussian particles, blue for binary particles and black for triangular particles.

We notice that the larger the scan window, the smaller the error for the three types of the particles and for both directions. Further, the average global influence of the type of particle is relatively constant, considering the two components of displacement.



## Size of diameter of particles

Table 5.2: Static parameters

# Particles	Size of the window	$u$	$v$
1000	$16 \times 16$	2.2	2.4

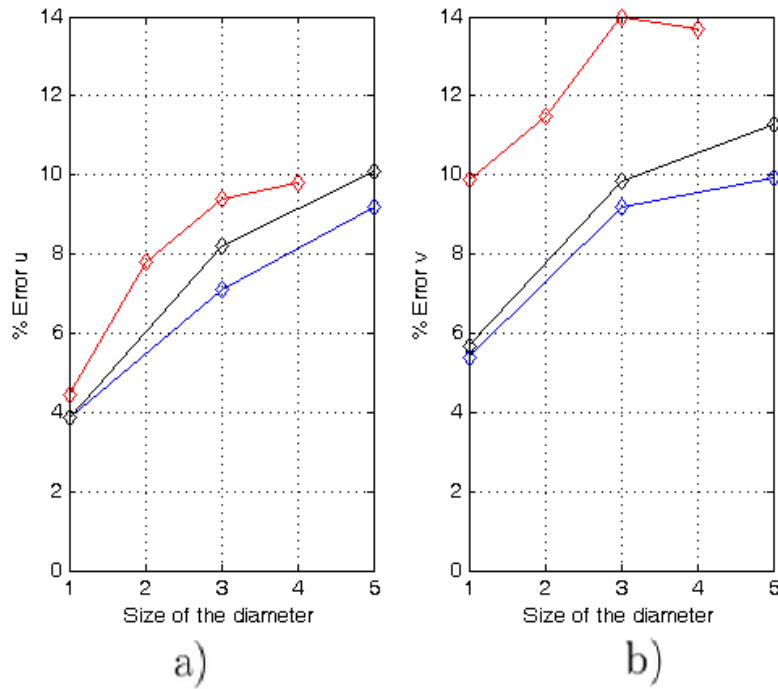


Figure 5.7: Particle diameter versus percentage relative error. a) Horizontal component of displacement. b) Vertical component of displacement.

In this case, the error increases directly with the diameter of the particles. This is due to the following effect. Correlation is based on comparing spatial structures. If the number of structures is low in each image the accuracy decreases. When the particles get large, the number of pixels with large gray level increases and the image background becomes white instead of black, and this decreases the number of spatial structures.

## Size of displacement

Table 5.3: Static parameters

# Particles	Size of scan window	Radius
1000	16×16	1

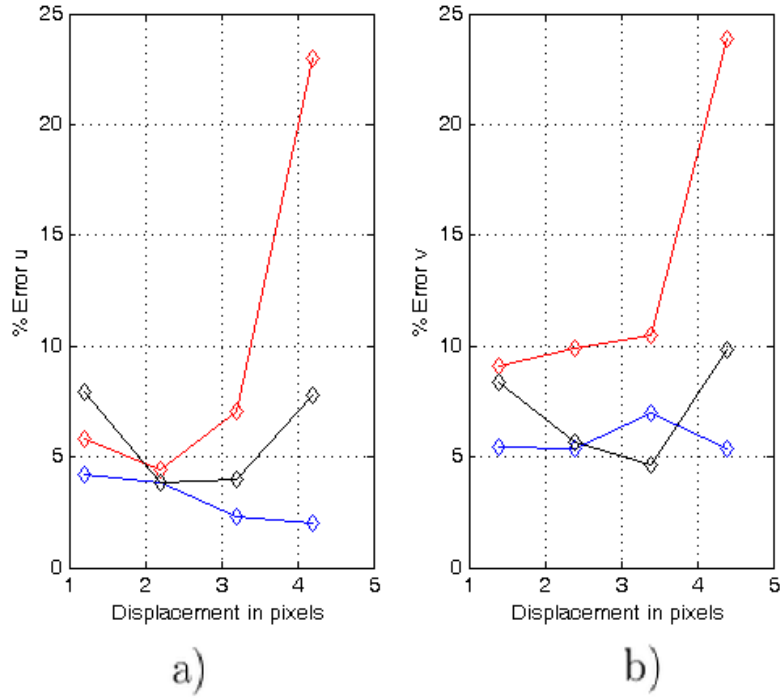


Figure 5.8: Effect of the size of displacement. a) Horizontal component of displacement. b) Vertical component of displacement.

It is observed that for large displacements the accuracy decreases. This is a consequence of the decrease in the number of similar spatial structures in both images being compared, since in the second one several particles move out of the scan window. As a rule of thumb, displacements greater than a quarter of the size of the scan window should be avoided.

## Number of particles per image

Table 5.4: Static parameters

Size of the window	Radius	$u$	$v$
16×16	1	2.2	2.4

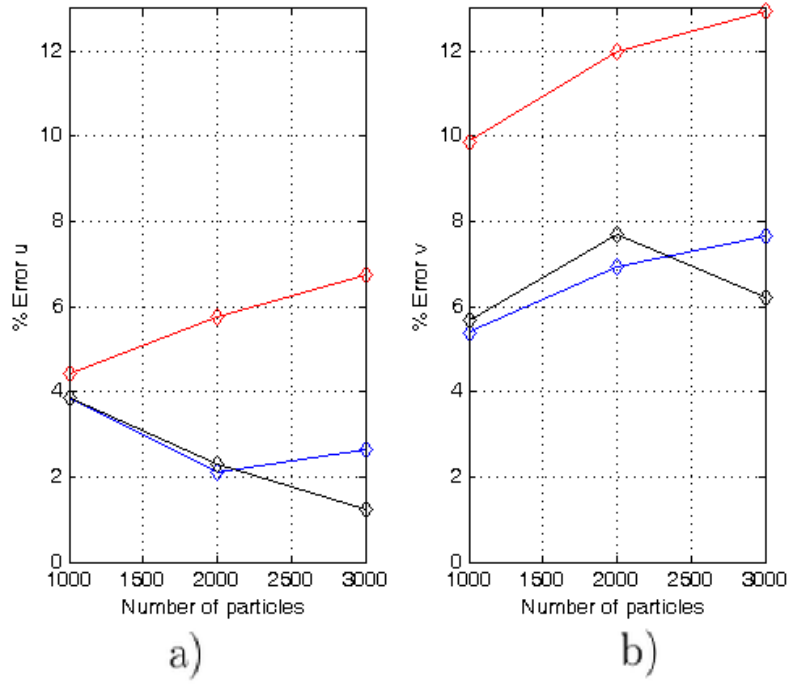


Figure 5.9: Effect of the number of particles. a) Horizontal component of displacement. b) Vertical component of displacement.

It is seen that the correlation method works better with a number of particles of at least 1000, which correspond to 15 particles per scan window. This agrees with the values found in the literature [1].

## Noise level

Table 5.5: Static parameters

Size of the window	Radius	# Particles	$u$	$v$
$16 \times 16$	1	1000	2.2	2.4

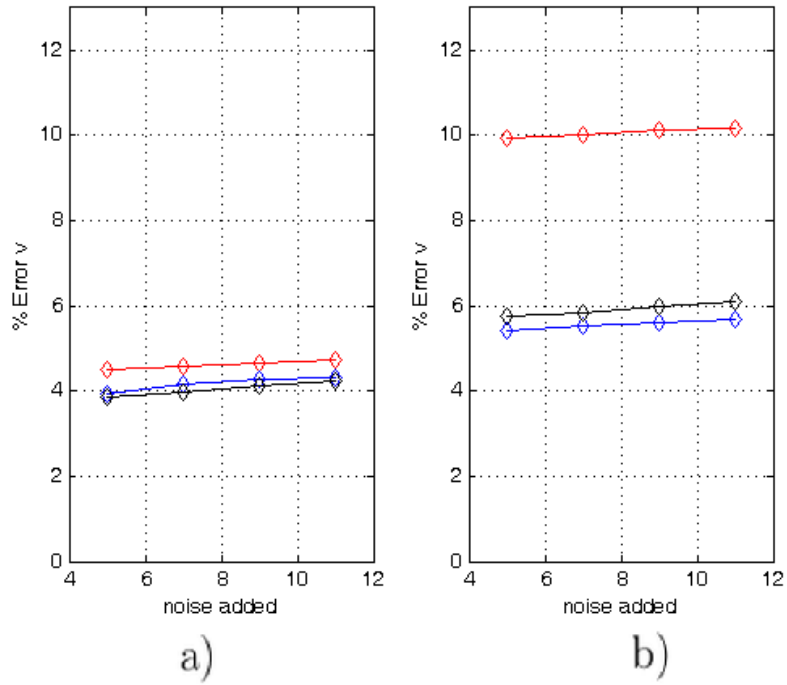


Figure 5.10: Noise level influence. a) Horizontal component of displacement. b) Vertical component of displacement.

As expected, the estimated error is affected by the level of noise. One can observe, however, that the variation of the error is not significant, which implies that the correlation method is relatively robust to this sort of noise. There is more error in  $v$  since it has bigger displacement than  $u$ .

## Contrast

Table 5.6: Static parameters

Size of the window	Radius	# Particles	$u$	$v$
$16 \times 16$	1	100	2.2	2.4

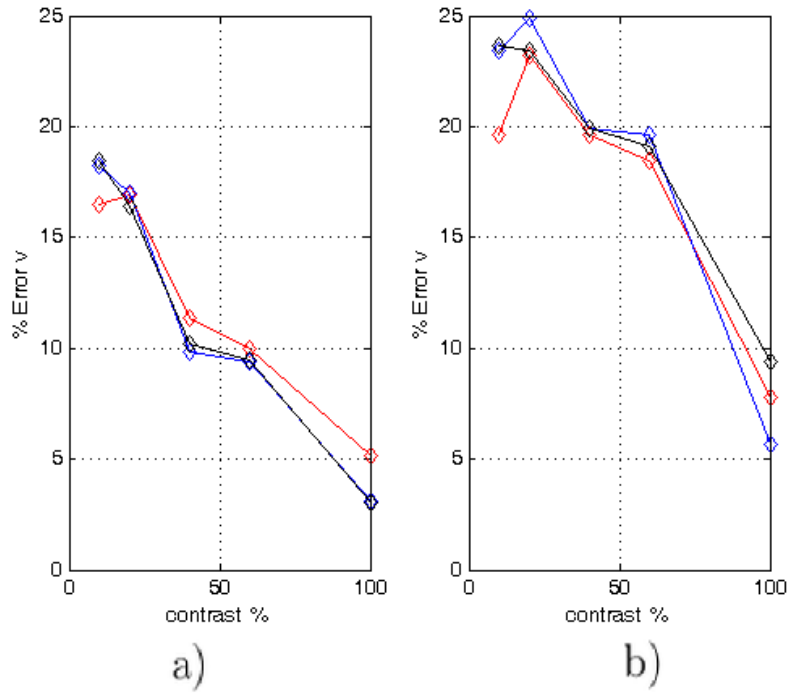


Figure 5.11: Influence of contrast. a) Horizontal component of displacement. b) Vertical component of displacement.

Unlike the previous result, the effect of the contrast on the accuracy of the results is significant. Actually, this results to be the most important parameter to take into consideration. The cause of this is the lack of spatial structures as well. When the contrast is low, the differences in intensities get smaller and the spatial structures become part of the background.

### 5.3.2 Sinusoidal displacement

It is supposed that the particles undergo horizontal displacement with a sine behavior and that one period of the sinusoidal function covers the entire image. This implies that displacement values are continuous and some are null and some point to the left and others to the right.

Analyses as those in the previous sections are presented.

#### Size of the scan window

Table 5.7: Static parameters

# Particles	Radius	u	v	A
1000	1	$A\sin(x)$	0	1

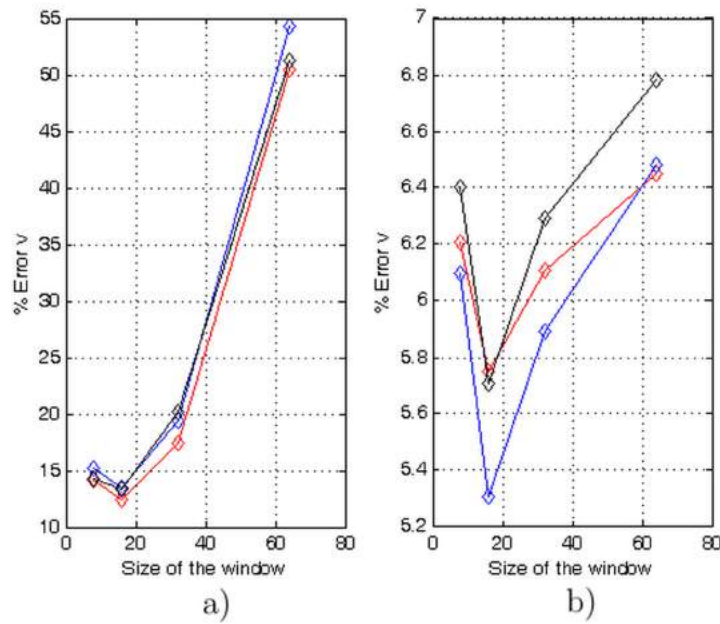


Figure 5.12: Size of scan window. a) Horizontal component of displacement. b) Vertical component of displacement.

Compared to the result obtained for constant displacement, in this case, the size of the window shows a larger effect on the accuracy. The reason for this is that images contain a continuous distribution of displacements, instead of being constant. Then, small windows do better because they do not tend to wash out the displacement information.

## Size of diameter of the particles

Table 5.8: Static parameters

# Particles	Size of the window	u	v	A
1000	16×16	$A\sin(x)$	0	1

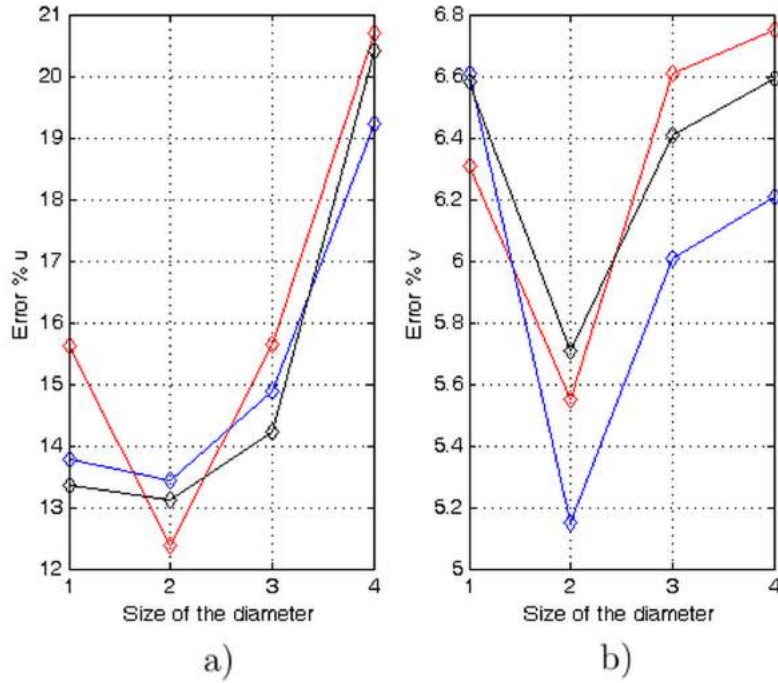


Figure 5.13: Diameter. a) Horizontal component of displacement. b) Vertical component of displacement.

There is an optimal value for the diameter of the particles, and it is around 2 pix; this for the three types of particles. The cause of this is the same as in the constant-displacement analysis. The error is larger when compared to the result with constant displacement.

## Amplitude of horizontal displacement

Table 5.9: Static parameters

# Particles	Size of the window	Radius	$u$	$v$
1000	$16 \times 16$	1	$Asin(x)$	0

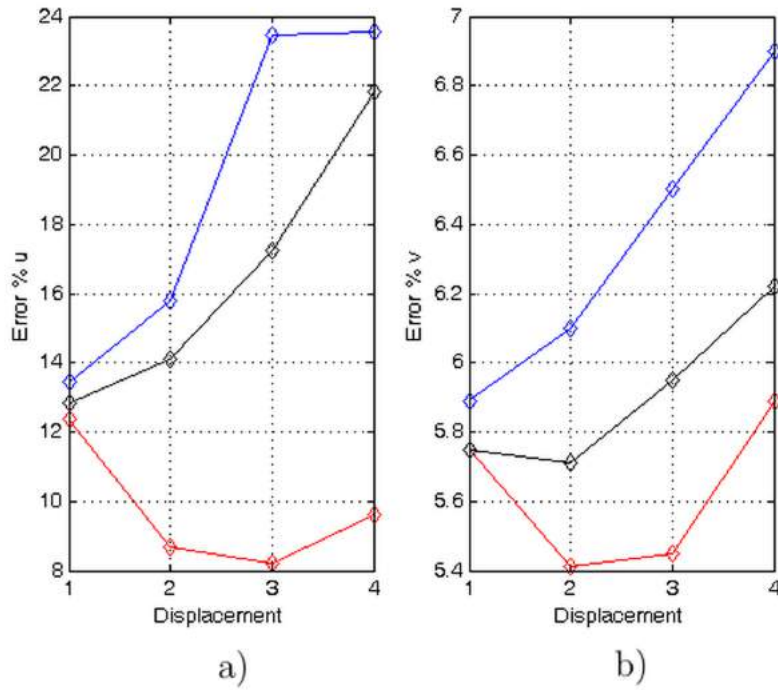


Figure 5.14: Amplitude of the displacement. a) Horizontal component of displacement. b) Vertical component of displacement.

It can be seen that for large displacement amplitude, the error is large as well. The displacement must not be larger than the quarter of the size of the scan window. It is worth noting the superior performance of the Gaussian particles.



## Number of particles per image

Table 5.10: Static parameters

Size of the window	Radius	u	v	A
16×16	1	$A\sin(x)$	0	1

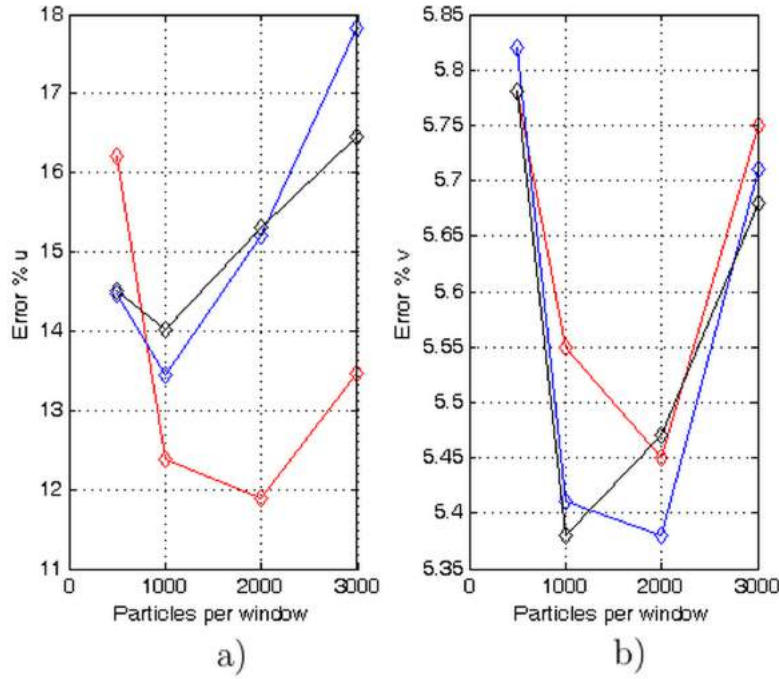


Figure 5.15: Number of particles. a) Horizontal component of displacement. b) Vertical component of displacement.

It is shown that the error does not vary significantly with the number of particles. However, as in the constant displacement, an optimal number of 1000 particles is found.

## Noise level

Table 5.11: Static parameters

# Particles	Window size	Radius	$u$	$v$	$A$
1000	$16 \times 16$	1	$A \sin(x)$	0	1

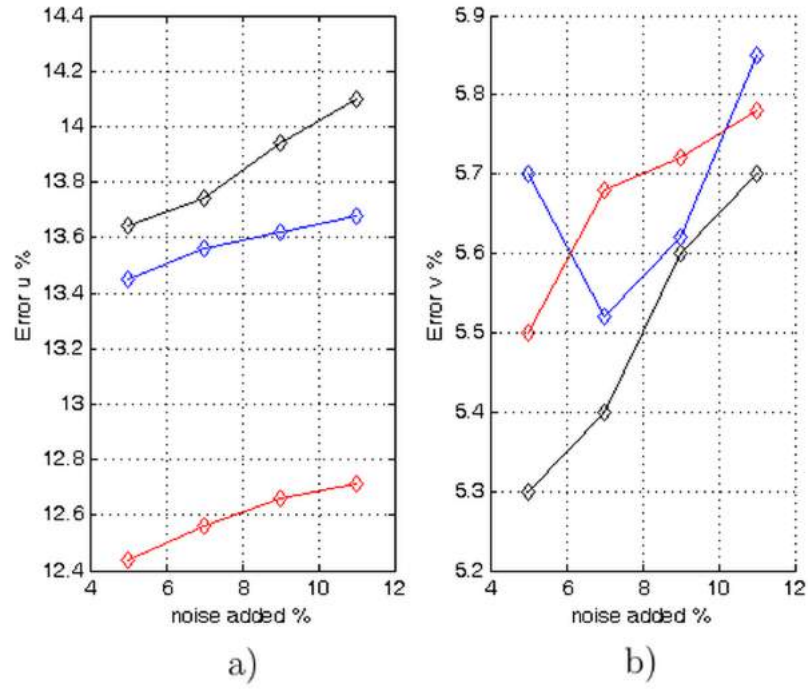


Figure 5.16: Noise level. a) Horizontal component of displacement. b) Vertical component of displacement.

As in the case of the constant displacement, the noise level effect on the precision of the results is almost constant. Also, the Gaussian particles yield the best results.

## Displacement of the scan window to half its size

Table 5.12: Static parameters

# Particles	Radius	u	v	A
1000	1	$A\sin(x)$	0	1

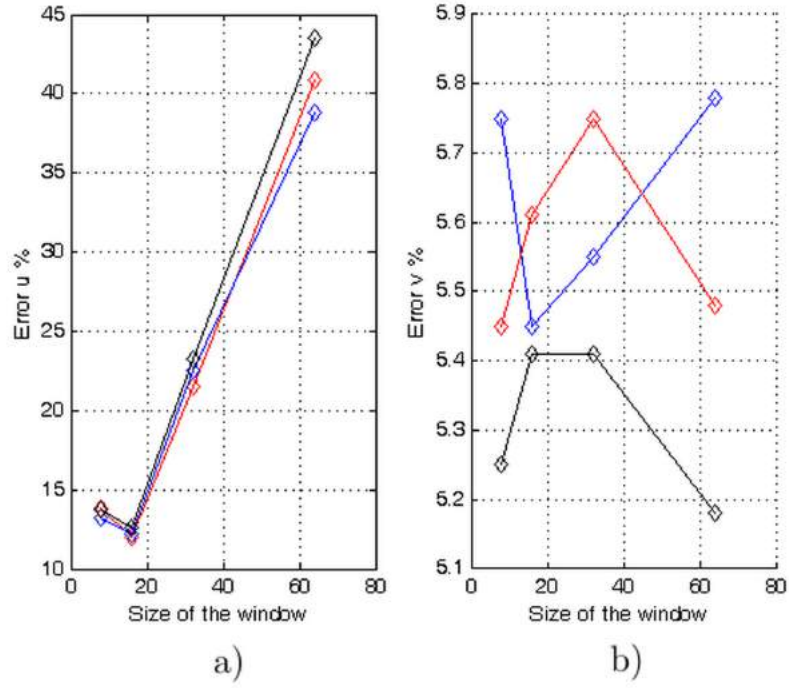


Figure 5.17: Displacement of scan window to half its size. a) Horizontal component of displacement. b) Vertical component of displacement.

Comparing these results with those in Figure 5.12, we can say that a slight improvement is obtained when the scan window moves half its size.

## Contrast

Table 5.13: Static parameters

# Particles	Window size	Radius	$u$	$v$	$A$
1000	$16 \times 16$	1	$A \sin(x)$	0	1

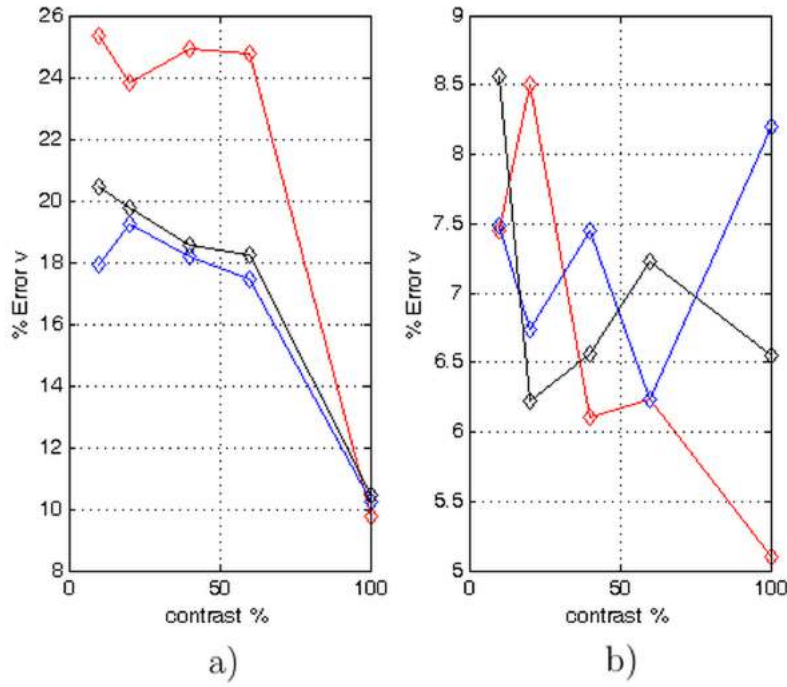


Figure 5.18: Contrast. a) Horizontal component of displacement. b) Vertical component of displacement.

The error is affected by the contrast more severely than in the case of the constant displacement, where errors on the order of 7% were observed.

### 5.3.3 Vortex-like displacement

This sort of displacement field resembles a whirlpool. The important features of this type of field are the mixture of displacements, negative and positive, and the incorporation of a continuous set of amplitude values. Then, applying this displacement field to the correlation algorithm, it may seem to be more challenging for its performance. The amplitude of the maximum displacement is given by  $A$ . For example, if  $A = 1/25$ , then the horizontal maximum displacement in pix is  $Ay = (1/25)(128/2) = 2.56$  pix.

#### Size of the scan window

Table 5.14: Static parameters

# Particles	Radius	u	v	A
1000	1	$Ay$	$-Ax$	$1/25$

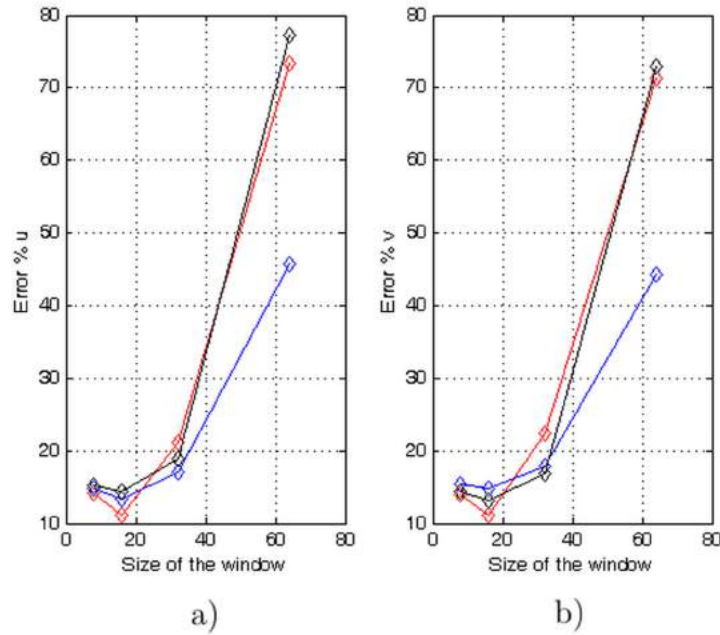


Figure 5.19: Size of the window of the correlation method. a) Horizontal component of displacement. b) Vertical component of displacement.

It can be seen that the correlation method has an optimal size for the scan window for this type of displacement; this size is  $16 \times 16$  pixels. As expected, the error in this case is greater than for the other types of displacement.

## Size of the diameter of the particles

Table 5.15: Static parameters

# Particles	Window size	u	v	A
1000	16×16	$Ay$	$-Ax$	$1/25$

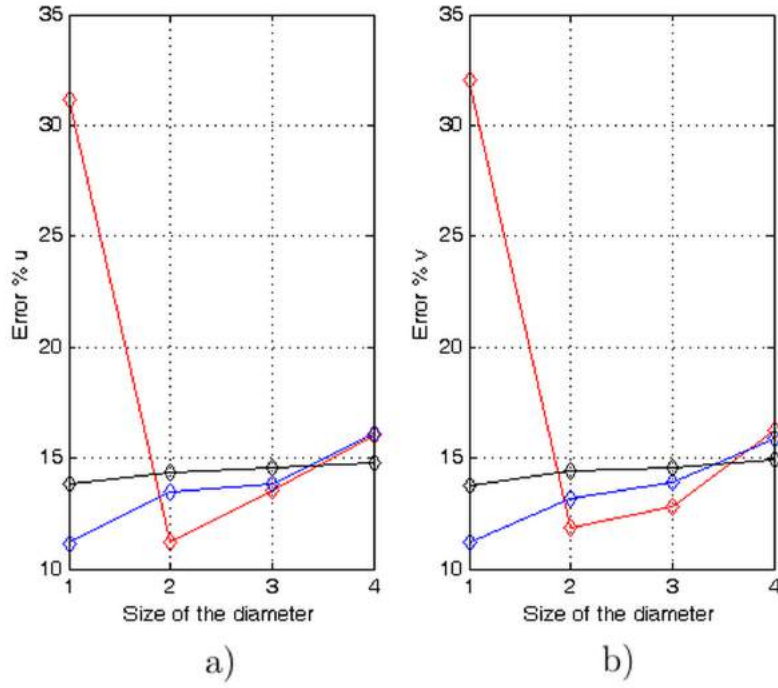


Figure 5.20: Diameter of particles. a) Horizontal component of displacement. b) Vertical component of displacement.

As before, the best value for the diameter of the particles is 2 pix. However, the calculated error is the largest one when considering the other types of displacement distributions.

## Amplitude of displacement

Table 5.16: Static parameters

# Particles	Window size	u	v	Radius
1000	16×16	$Ay$	$-Ax$	1

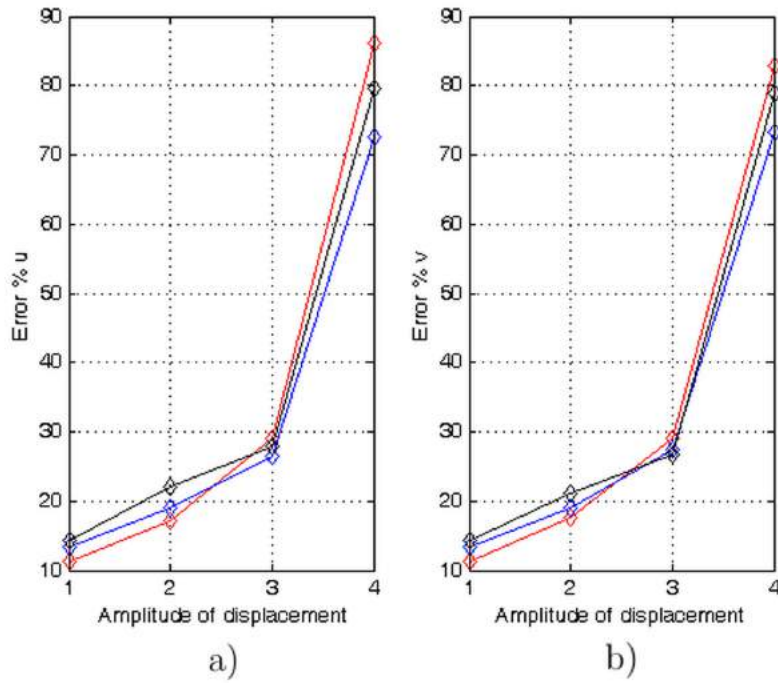


Figure 5.21: Amplitude of displacement. a) Horizontal component of displacement. b) Vertical component of displacement.

The errors for horizontal and vertical components of displacements are similar. Again, the largest errors are obtained for this type of displacement (as commented above, this is the most challenging type of displacement to be analyzed).

## Number of particles per image

Table 5.17: Static parameters

Radius	Window size	u	v	A
1	16×16	$Ay$	$-Ax$	$A/25$

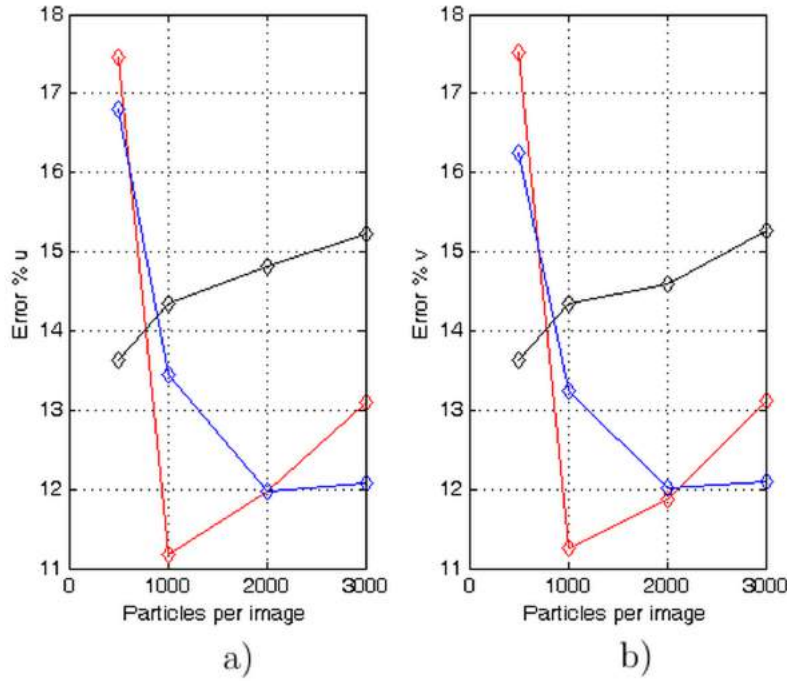


Figure 5.22: Number of particles. a) Horizontal component of displacement. b) Vertical component of displacement.

As for other types of displacements, the optimal value for the number of particles with vortex-like displacement is in the range from 1000 particles to 2000 particles.



## Nosie level

Table 5.18: Static parameters

# Particles	Window size	Radius	$u$	$v$	$A$
1000	$16 \times 16$	1	$Ay$	$-Ax$	$1/25$

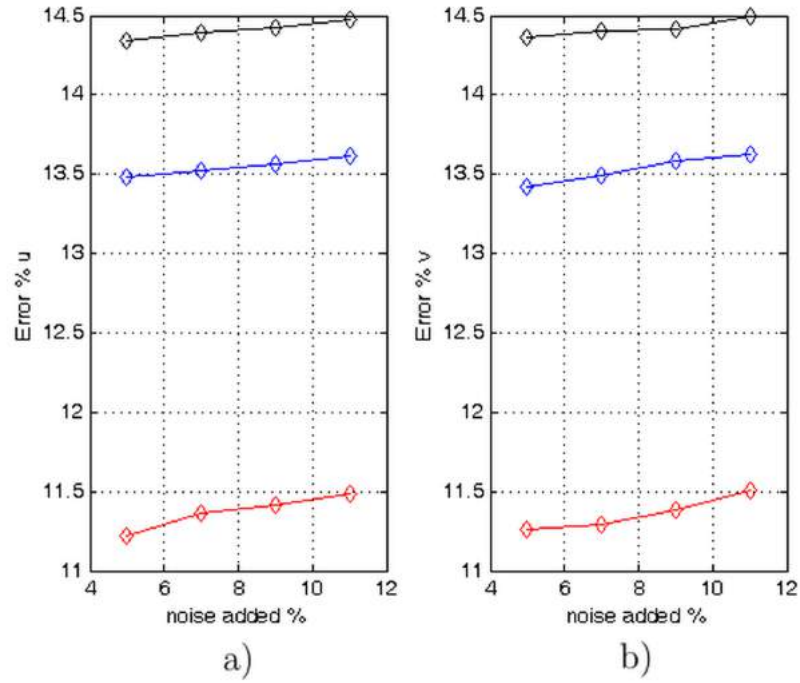


Figure 5.23: Noise level. a) Horizontal component of displacement. b) Vertical component of displacement.

As it can be seen, the error arising from noise level is fairly constant and is on the order of the obtained for sinusoidal displacement. The Gaussian particles perform relatively better than the others.

## Size of the scan window, when it moves half of its size

Table 5.19: Static parameters

Radius	# Particles	u	v	A
1	1000	$Ay$	$-Ax$	$1/25$

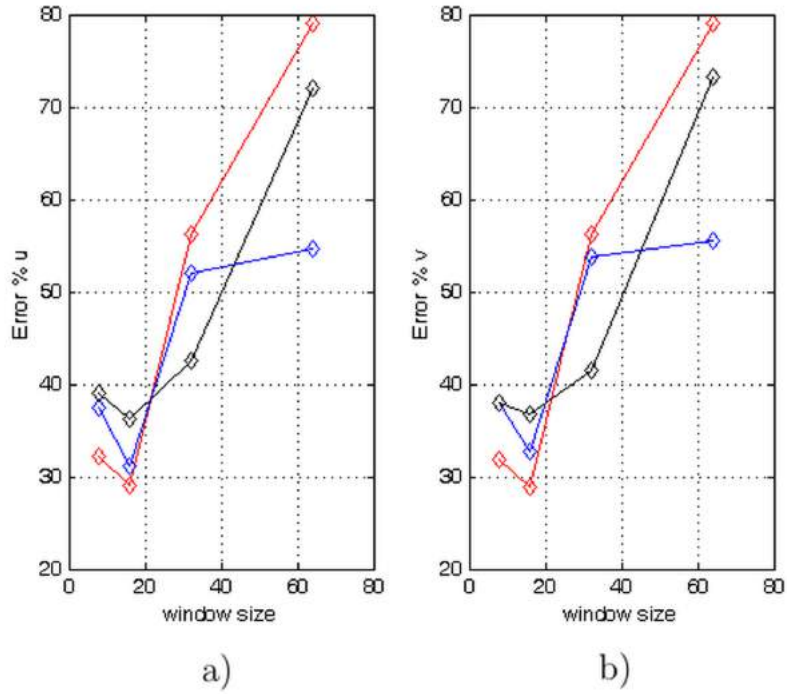


Figure 5.24: Scan window to half its size. a) Horizontal component of displacement. b) Vertical component of displacement.

Compared with the sinusoidal result, the error for this sort of displacement distribution is larger. The best results are obtained for a window size of  $16 \times 16$  pix as well. Additionally, the errors do not show any dependency on the type of particles.

## Contrast

Table 5.20: Static parameters

# Particles	Size of the window	Radius	$u$	$v$	$A$
1000	$16 \times 16$	1	$Ay$	$-Ax$	$1/25$

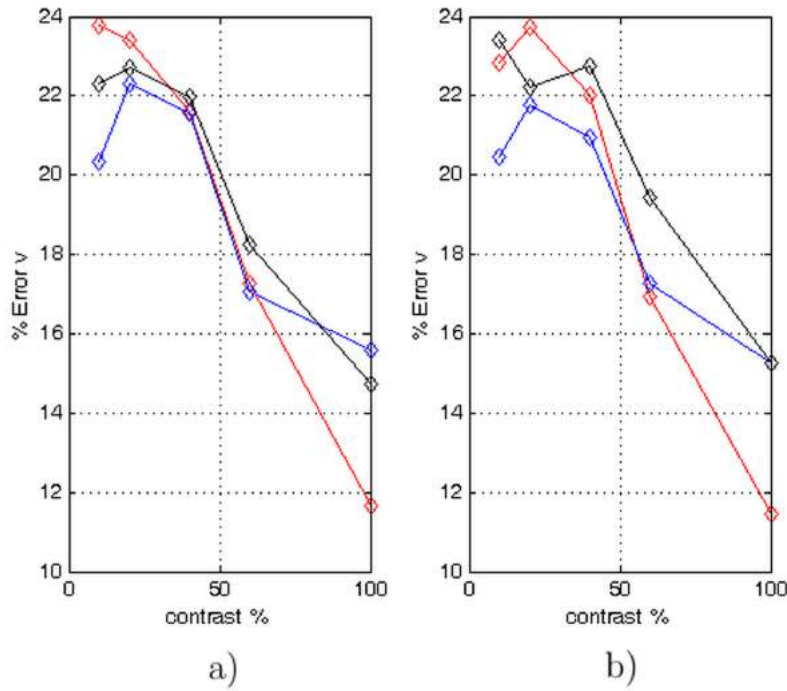


Figure 5.25: Contrast. a) Horizontal component of displacement. b) Vertical component of displacement.

In general, this parameter is the one with the largest effect on the error. Therefore, this variable should be carefully observed to attain large values in the experimental stage. As previously noted, window sizes larger than  $32 \times 32$  produce even larger errors, but this parameter can be selected straightforwardly as to produce low errors.

Furthermore, considering the other two types of displacement, on average, the vortex-type displacement yields slightly lower accuracy for the simulations.

## 5.4 Results of the Lucas-Kanade algorithm

In this Section the results obtained with the Lucas-Kanade algorithm are presented for all types of displacements. An analysis with the same changing variables is carried out.[4,5,6]

### 5.4.1 Constant displacement for Lucas-Kanade algorithm

Number of iterations

Table 5.21: Static parameters

# Particles	Size of the window	Radius	$u$	$v$
1000	$5 \times 5$	1	1.2	1.4

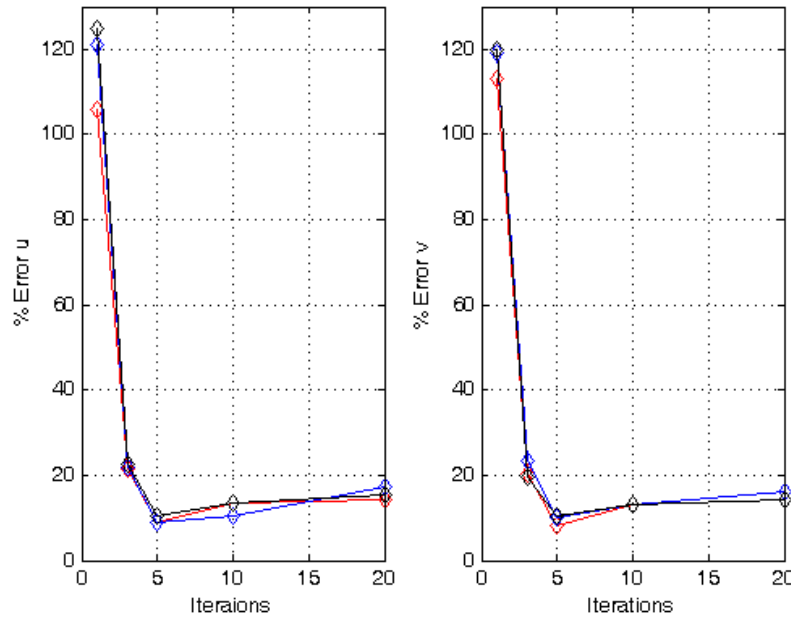


Figure 5.26: Varying the number of iterations. Percentage relative error. a) Horizontal component of displacement. b) Vertical component of displacement.

The error converges to the lowest value of percentage error in 5 iterations. This result is taken as a reference for the rest of the analysis.

## Size of the scan window

Table 5.22: Static parameters

# Particles	Radius	$u$	$v$
1000	1	1.2	1.4

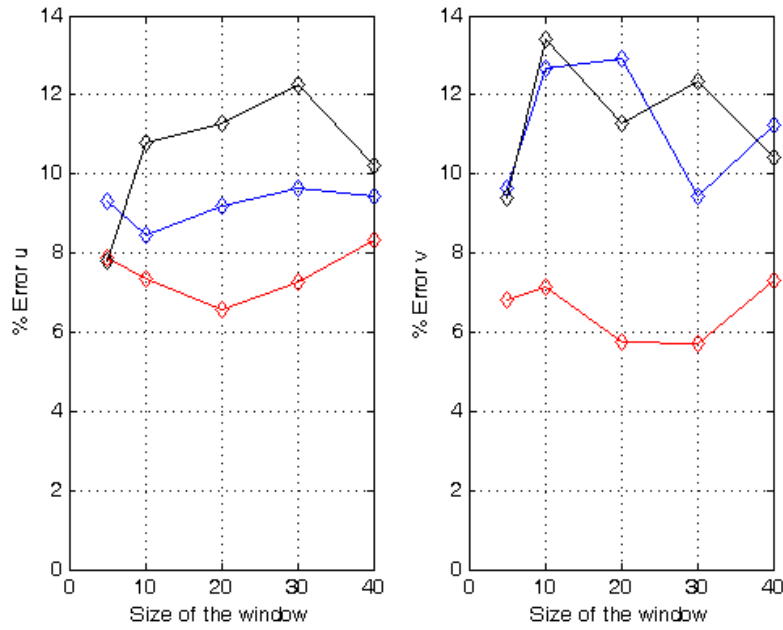


Figure 5.27: Window size. a) Horizontal component of displacement. b) Vertical component of displacement.

The Gaussian particles present the best performance. Besides, the best condition for the window size is at  $20 \times 20$  pixels. This is an expected result since the larger of input data points, the larger the level of information for the method. As in the correlation method an average error of 4% was obtained, the Lucas-Kanade algorithm produces worst results.

## Size of diameter of the particles

Table 5.23: Static parameters

# Particles	Size of the window	$u$	$v$
1000	$5 \times 5$	1.2	1.4

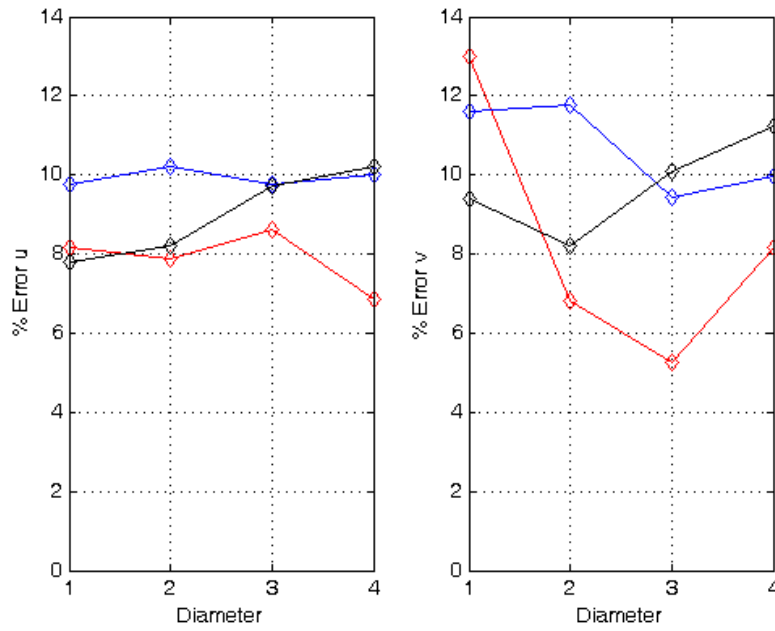


Figure 5.28: Diameter. a) Horizontal component of displacement. b) Vertical component of displacement.

It is noticed that a diameter of particle of around 3 pix gives the best overall results in combination with the use of Gaussian particles. The obtained performance is lower than for the correlation method.

## Size of the displacement

Table 5.24: Static parameters

# Particles	Size of the window	Radius
1000	$5 \times 5$	1

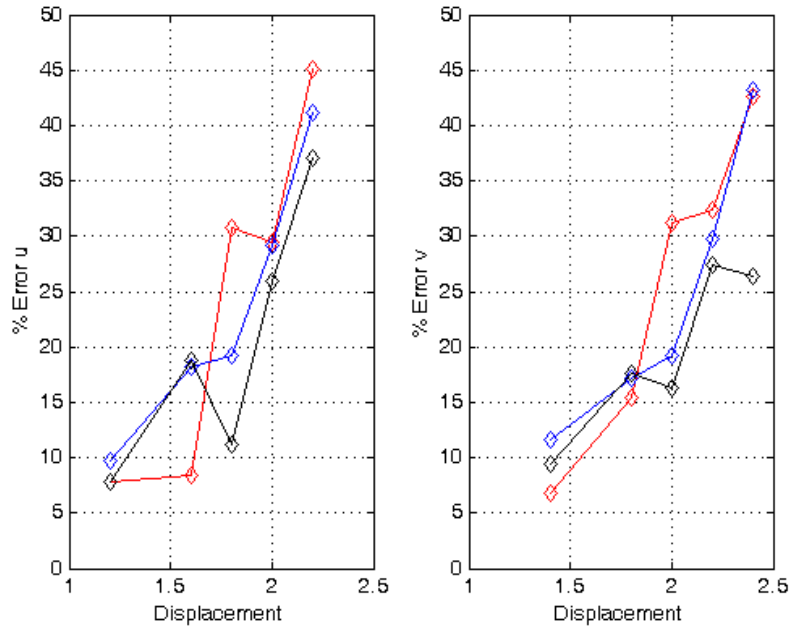


Figure 5.29: Displacement. a) Horizontal component of displacement. b) Vertical component of displacement.

The observed behavior is as the found for the correlation method (qualitatively and quantitatively): when the displacement is beyond a threshold value, the similarity of the images to be compared decreases.

## Number of particles per image

Table 5.25: Static parameters

Size of the window	Radius	$u$	$v$
$5 \times 5$	1	1.2	1.4

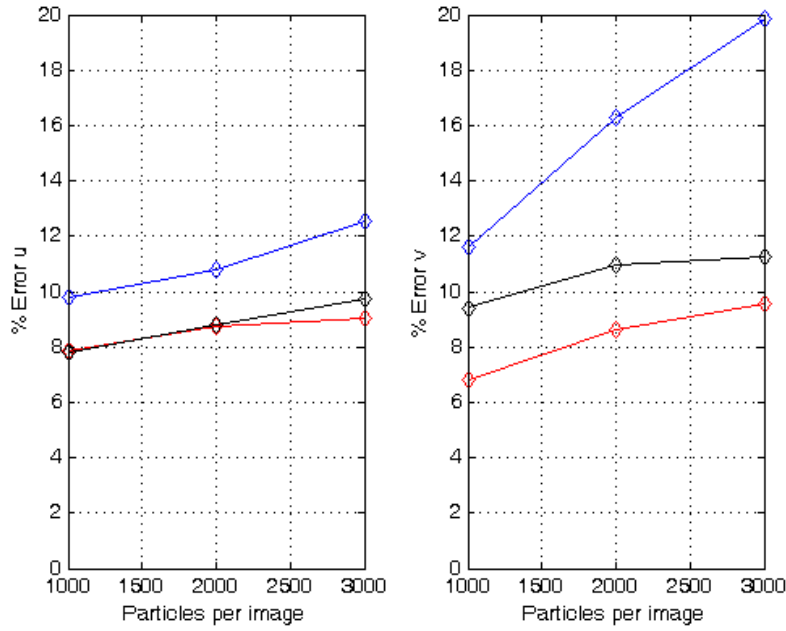


Figure 5.30: Number of particles. a) Horizontal component of displacement. b) Vertical component of displacement.

In this test it can be concluded that the Lucas-Kanade works better for 1000 particles in the region of  $128 \times 128$  pix. When the number of particles is lower than 1000, the error increases significantly because the image spatial information gets lower. When compared with the correlation algorithm, the accuracy is lower.



Nosie level

Table 5.26: Static parameters

# Particles	Size of the window	Radius	$u$	$v$
1000	$5 \times 5$	1	1.2	1.4

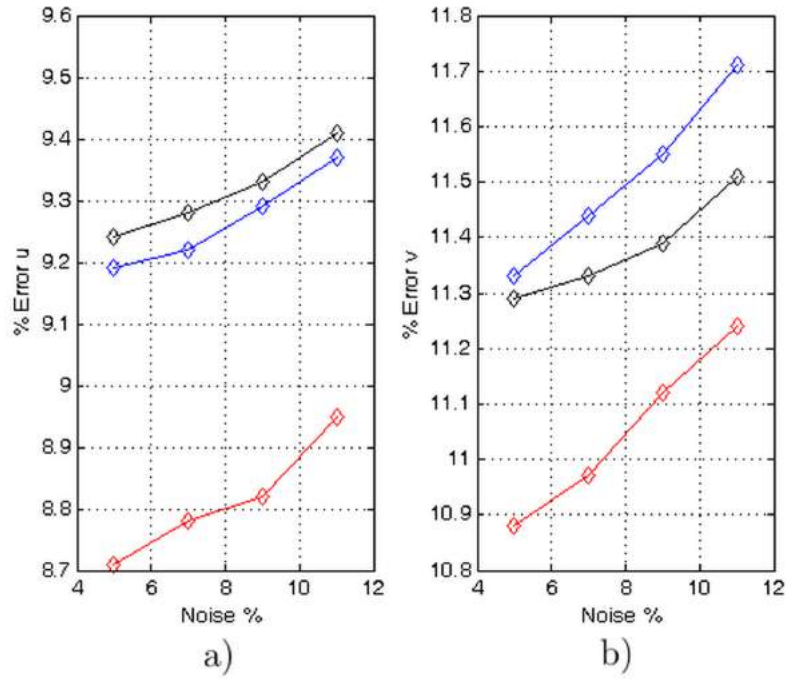


Figure 5.31: Noise. a) Horizontal component of displacement. b) Vertical component of displacement.

The computed percentage relative errors are around twice those obtained by the correlation method even for the Gaussian particle that has the best performance.

## Contrast

Table 5.27: Static parameters

# Particles	Size of the window	Radius	$u$	$v$
1000	$5 \times 5$	1	1.2	1.4

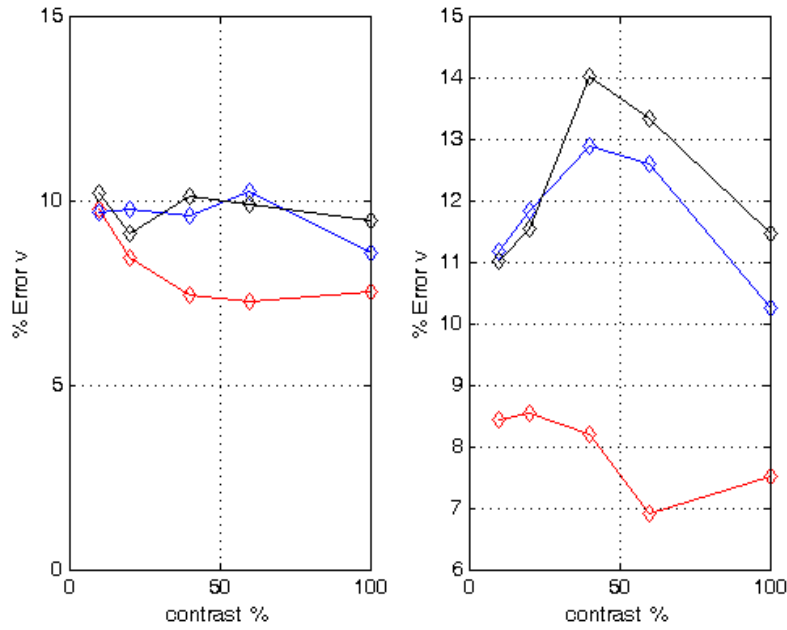


Figure 5.32: Varying the contrast of the images in the Lucas-Kanade algorithm. a) Horizontal component of displacement. b) Vertical component of displacement.

The contrast of the images does not affect significantly the performance since the variation of error is just 3%. When compared with the correlation result, the accuracy of this result is on average greater.

In summary, for this type of displacement distribution, the performance of the Lucas-Kanade is better than the computed by the correlation method, about 50%.

## 5.4.2 Sinusoidal displacement

### Size of the scan window

Table 5.28: Static parameters

# Particles	Radius	u	v	A
1000	1	$A \sin(x)$	0	1

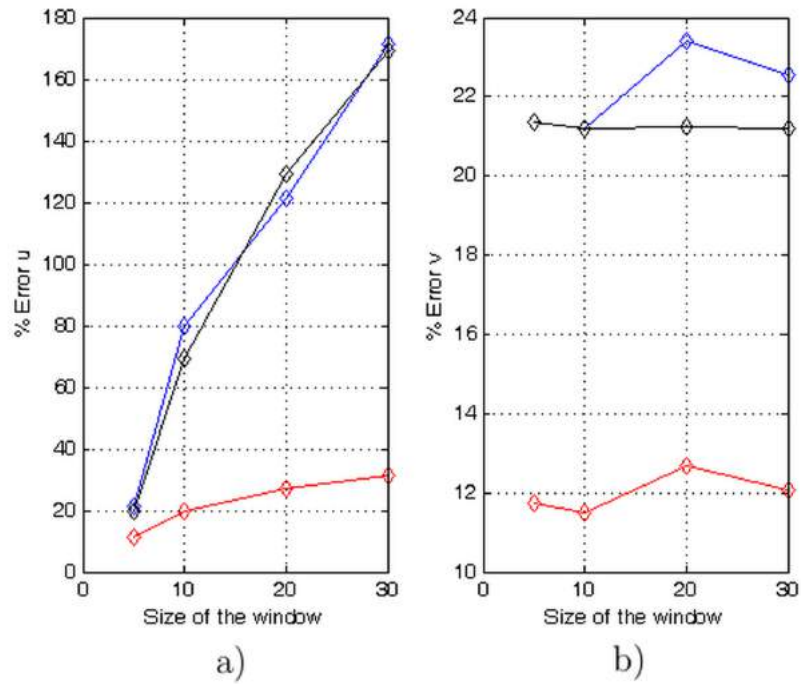


Figure 5.33: Varying the size of the scan window of the Lucas-Kanade method with sine displacement. a) Horizontal component of displacement. b) Vertical component of displacement.

The maximum displacement is less than a pixel; small windows are the best option, because a big window gets an averaged result. When compared with the correlation method, the accuracy is lower.

## Size of the diameter of the particles

Table 5.29: Static parameters

# Particles	Window size	u	v	A
1000	$5 \times 5$	$A \sin(x)$	0	1

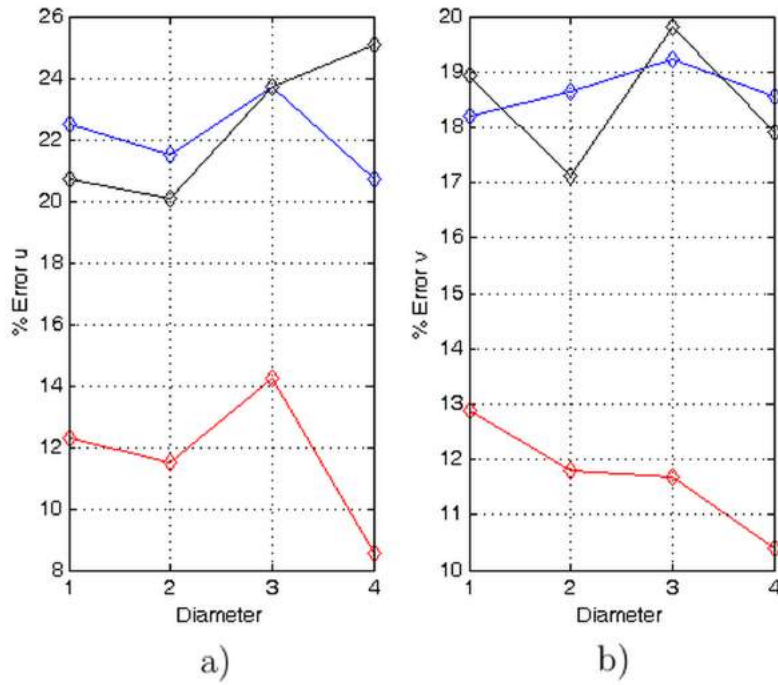


Figure 5.34: Diameter versus error in the Lucas-Kanade method with sine displacement. a) Horizontal component of displacement. b) Vertical component of displacement.

In this case, the Lucas-Kanade method outperforms the correlation method, where errors of approximately 13% were computed.

## Horizontal displacement

Table 5.30: Static parameters

# Particles	Window size	u	v	Radius
1000	5×5	$Asin(x)$	0	1

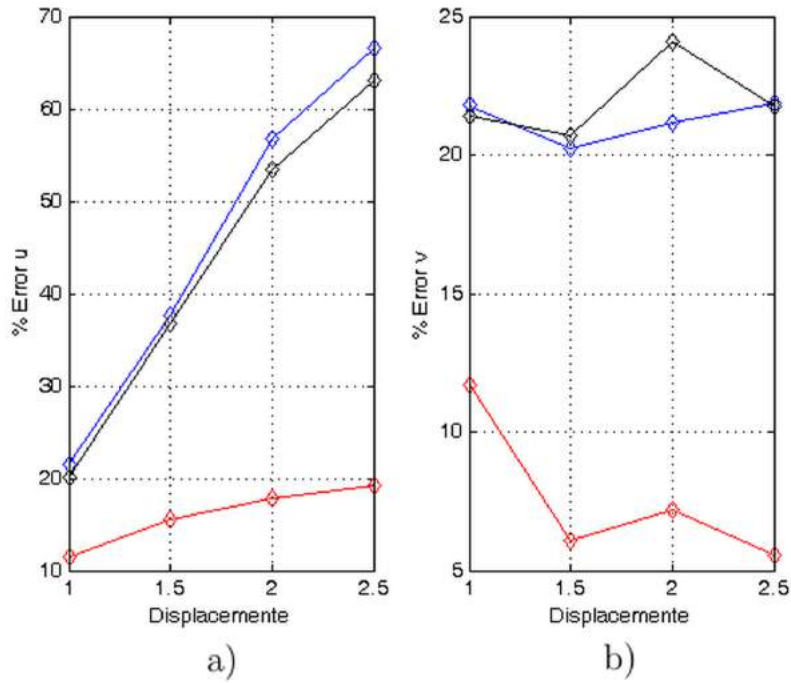


Figure 5.35: Displacement. a) Horizontal component of displacement. b) Vertical component of displacement.

In this test the triangular and the binary particles produce worst results. In the correlation method an average value of 8% for the error was calculated.

## Number of particles per image

Table 5.31: Static parameters

Size of the window	Radius	u	v	A
$5 \times 5$	1	$A \sin(x)$	0	1

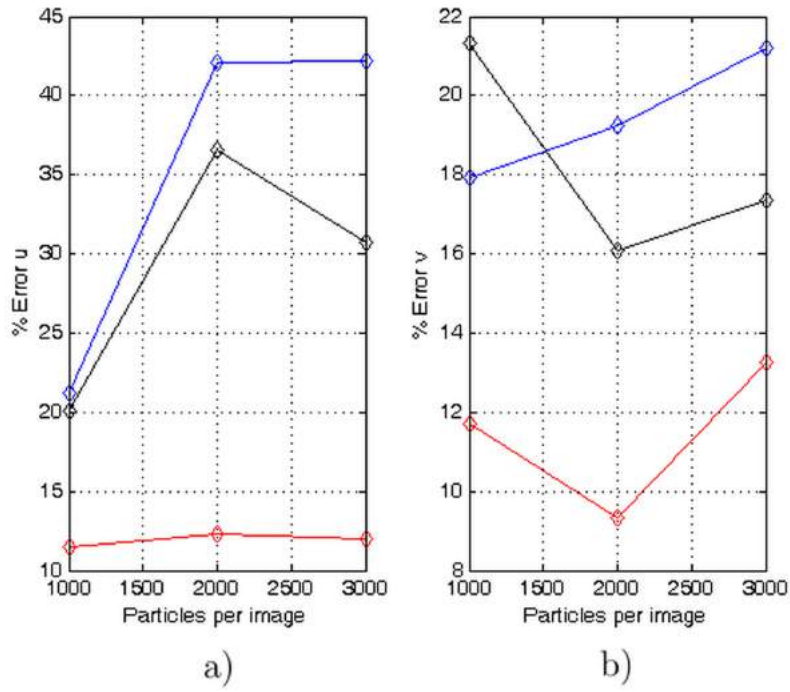


Figure 5.36: Number of the particles. a) Horizontal component of displacement. b) Vertical component of displacement.

An accuracy lower than the obtained with correlation is observed.

## Noise level

Table 5.32: Static parameters

# Particles	Window size	Radius	$u$	$v$	$A$
1000	$5 \times 5$	1	$A \sin(x)$	0	1

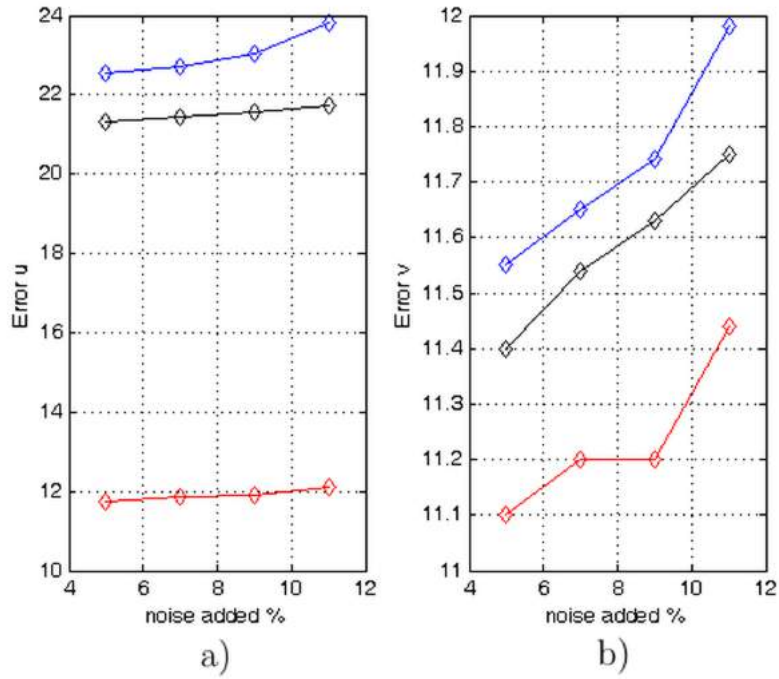


Figure 5.37: Adding noise to the displacement of the particles in the Lucas-Kanade algorithm with sine displacement. a) Horizontal component of displacement. b) Vertical component of displacement.

The error for the  $u$  component shows the same error as in the case of correlation. Unlike this, for the  $v$  component, the accuracy decreases.

## Contrast

Table 5.33: Static parameters

# Particles	Window size	Radius	$u$	$v$	$A$
1000	$5 \times 5$	1	$A \sin(x)$	0	1

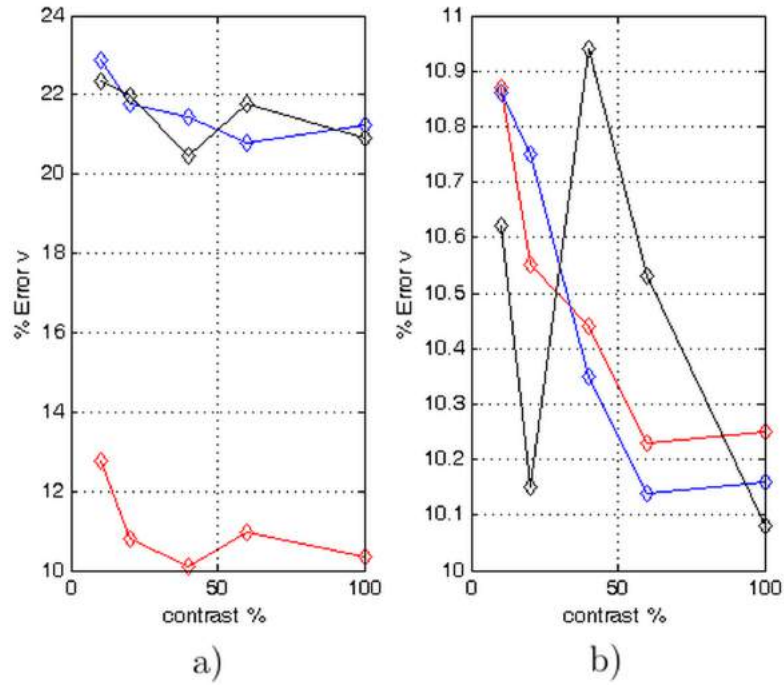


Figure 5.38: Contrast. a) Horizontal component of displacement. b) Vertical component of displacement.

In general, for this type of displacement the Lucas-Kanade algorithm outperforms slightly the correlation method by around 10%.



### 5.4.3 Vortex-like displacement

The method yields larger errors for displacements greater than 2 pix. Therefore, these simulation results consider maximum displacement values of one third of those used in the correlation algorithm (around 1 pix).

#### Size of the scan window

Table 5.34: Static parameters

# Particles	Radius	u	v	A
1000	1	$Ay$	$-Ax$	$1/75$

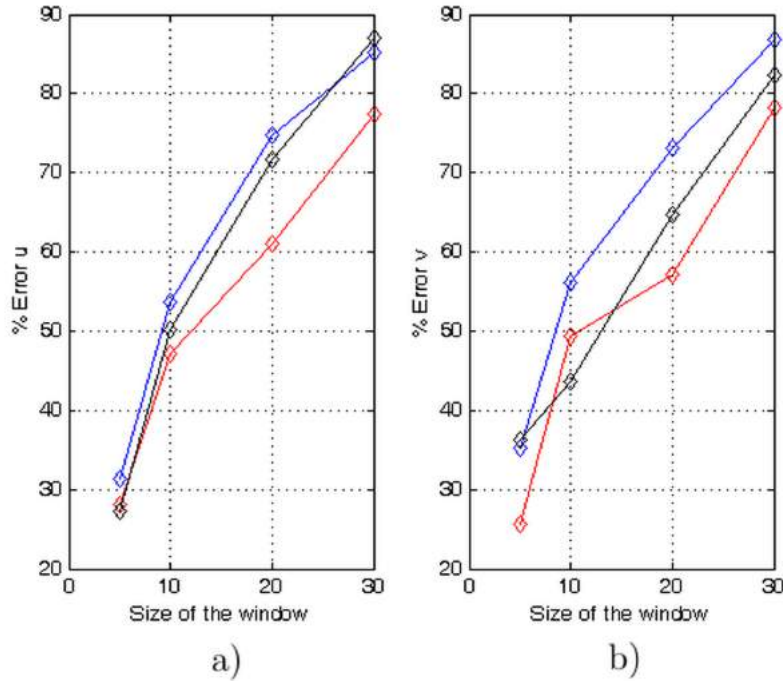


Figure 5.39: Size of the scan window. a) Horizontal component of displacement. b) Vertical component of displacement.

When compared with the correlation method, the errors are significantly larger. As mentioned in the correlation section, this type of displacement is one of the most challenging for evaluation purposes.

## Size of the diameter of the particles

Table 5.35: Static parameters

# Particles	Size of the scan window	u	v	A
1000	$5 \times 5$	$Ay$	$-Ax$	$1/75$

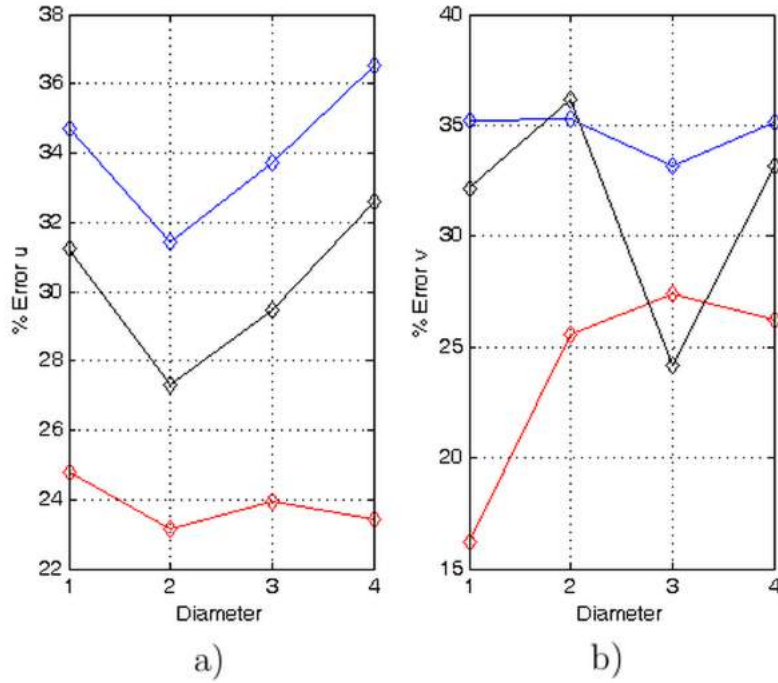


Figure 5.40: Diameter versus error in the Lucas Kanade algorithm with vortex-like displacement. a) Horizontal component of displacement. b) Vertical component of displacement.

The optimal value of the diameter is not clear, but is greater than for the previous types of displacement.

## Amplitude of displacement

Table 5.36: Static parameters

# Particles	Size of the scan window	u	v	Radius
1000	5×5	$Ay$	$-Ax$	1

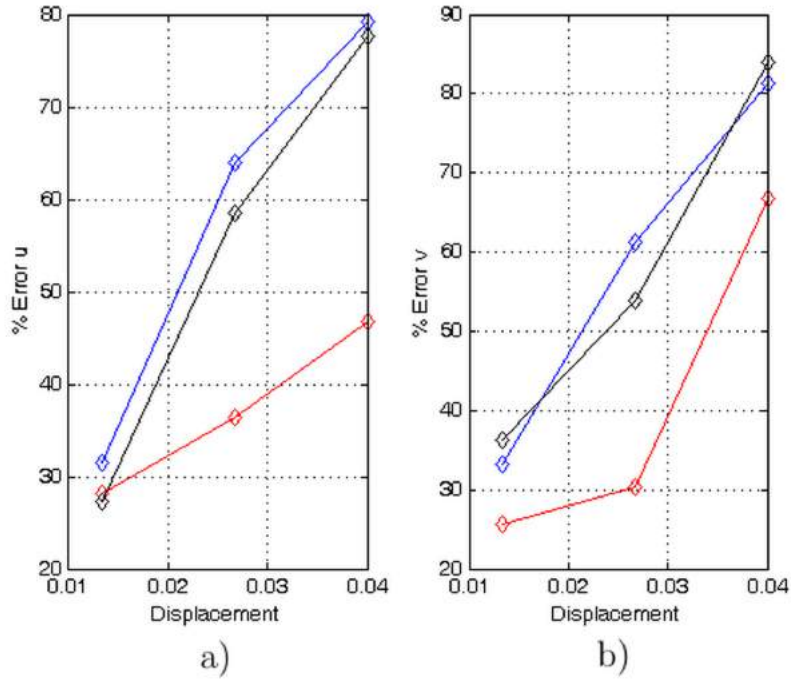


Figure 5.41: Varying the displacement of the particles of the Lucas-Kanade method. a) Horizontal component of displacement. b) Vertical component of displacement.

When the displacement has a vortex-like behavior the constant  $A$  must have a small value because this type of displacement increases a lot in the edge of the image and almost all the particles escape from the window and the scan window can not detect them. For comparing purposes, the correlation average error was 15%.

## Number of particles per image

Table 5.37: Static parameters

Radius	Size of the scan window	u	v	A
1	$5 \times 5$	$Ay$	$-Ax$	$1/75$

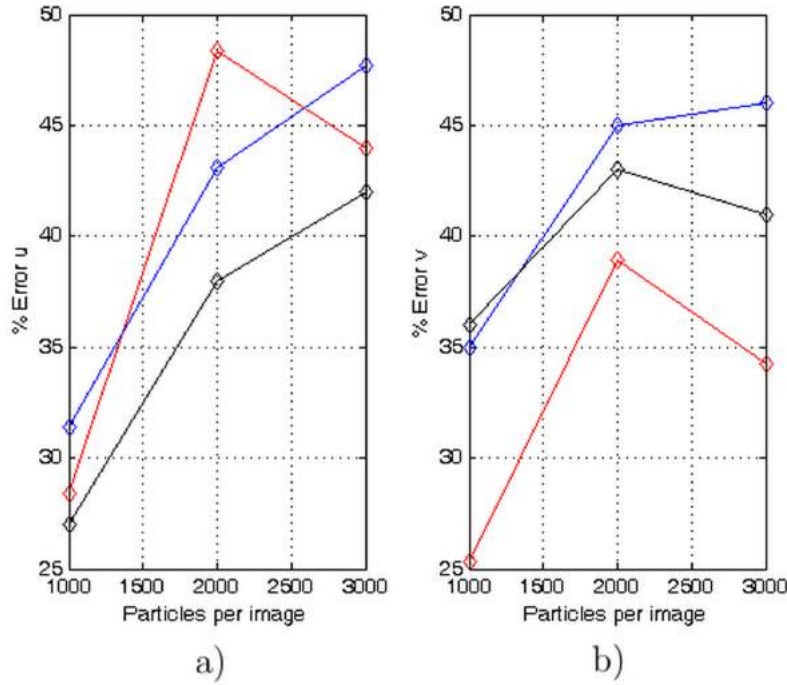


Figure 5.42: Number of the particles. a) Horizontal component of displacement. b) Vertical component of displacement.

The optimized value of particles coincides with that obtained in the correlation method, but error values are larger. The correlation method has better performance than the Lucas-Kanade algorithm.

## Noise level

Table 5.38: Static parameters

# Particles	Window size	Radius	$u$	$v$	$A$
1000	$5 \times 5$	1	$Ay$	$-Ax$	$1/75$

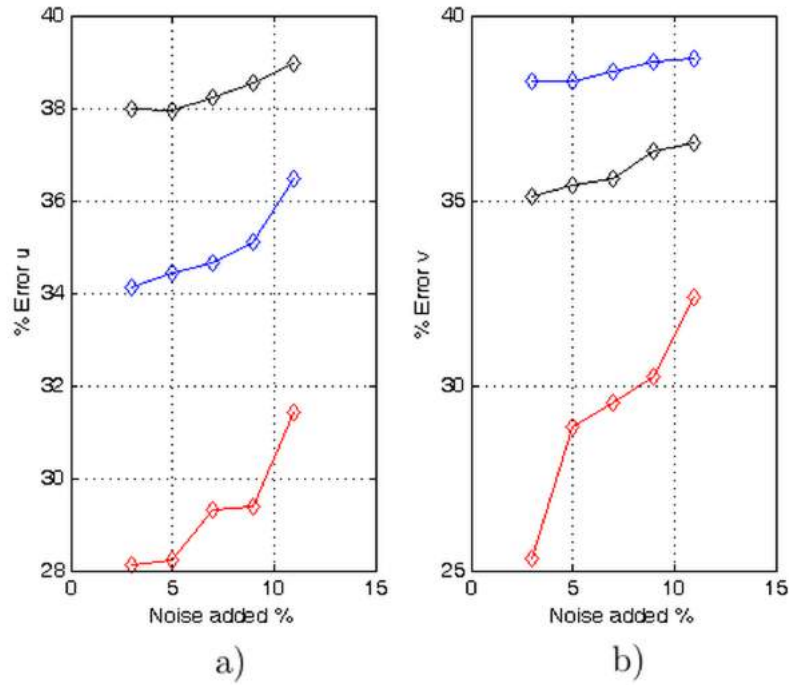


Figure 5.43: Noise level. a) Horizontal component of displacement. b) Vertical component of displacement.

For this variable and the next, the tendency of the error is similar to the observed for correlation, but again the accuracy is smaller.

Considering this type of displacement distribution, the results show that the accuracy of the Lucas-Kanade method is lower than the obtained by the correlation algorithm (about 60%).

## Contrast

Table 5.39: Static parameters

# Particles	Window size	Radius	$u$	$v$	$A$
1000	$5 \times 5$	1	$Ay$	$-Ax$	$1/75$

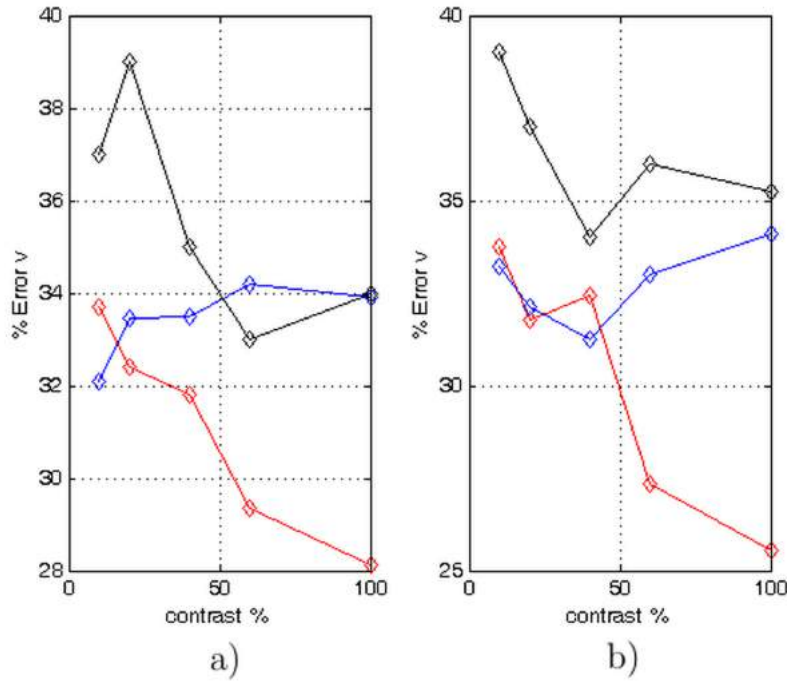


Figure 5.44: Changing the contrast of the image of Lucas-Kanade algorithm with vortex-like displacement. a) Horizontal component of displacement. b) Vertical component of displacement.

Small contrast affects less drastically the performance of the Lucas-Kanade algorithm. In this test the Gaussian particles have the best performance of all the three types of particles.

## 5.5 Results of hybrid Lucas-Kanade

The performance of the hybrid method is, in the best cases, about 10% better than the Lucas-Kanade algorithm. The qualitative form of the obtained results is practically the same as the form of the results obtained by the Lucas-Kanade method. Therefore, the plots and descriptions are omitted. The details of the numerical simulations are as follows. First, we apply 5 iterations of the Lucas-Kanade algorithm. Then, 15 iterations are executed by the swarm algorithm. In this case, we use a population of 25 birds and a scan window of  $4 \times 4$  pix for the swarm code.

## 5.6 Experimental results

The experimental setup has the objective to test the performance of the three algorithms with real images. The setup is similar to the setup described in the introduction of this work: a laser sheet of light illuminates the object under test and a camera registers a certain number of images.

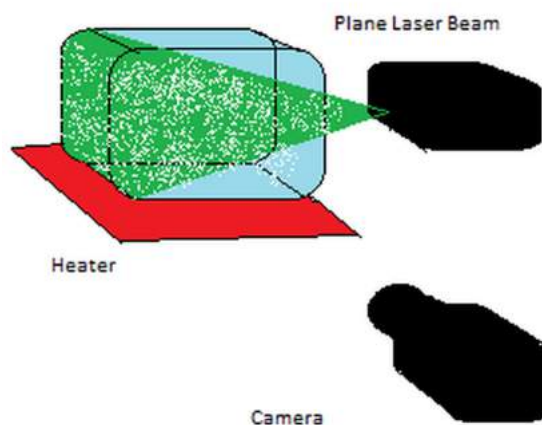


Figure 5.45: Diagram of the experimental setup.

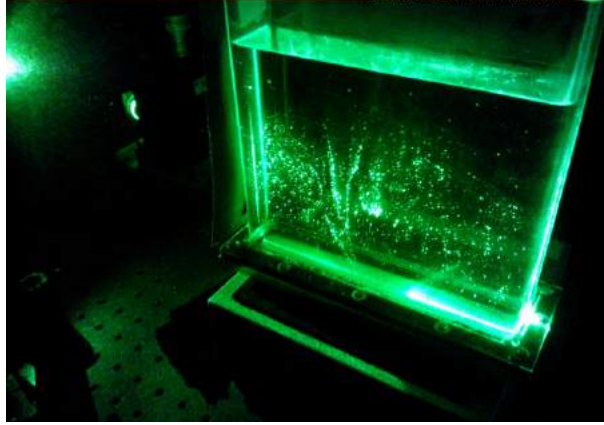


Figure 5.46: Photograph of the experimental setup.

This setup consists of a box-like recipient with water that is heated by an electronic heater. For the seeding of the water we use talcum powder. Because of the heat, a convection water flow is established. To capture the movement of the water flow, a laser sheet is used in such a way that only one plane of the object is studied. The light sheet is obtained by using a spherical lens and a cylindrical lens.

Following, the main parameters of the system are listed.

- Temperature at the lower part,  $60^{\circ}\text{C}$
- Temperature at the top part,  $25^{\circ}\text{C}$
- The laser is a 400-mW Nd:YAG laser emitting at 532 nm
- The camera is from Lumenera, model lm165m
- Frames per second, 15
- Exposition time, 67 ms
- Size of the recipient, 20x10x30 cm<sup>3</sup>
- Size of the images, 1040X1392 pix
- The range of values of displacements go from -4 pix to 4.5 pix
- The range of values of velocity are within -7.2 to 9.8 mm/s
- The diameter of the particles goes from 2 pix to 6 pix
- The focal length of the camera is 35mm
- The distance between the camera and the recipient is 1.07 m
- The distance between the recipient and the laser exit is 34.5 cm
- The spherical lens has a focal length of 500 mm
- The cylindrical lens has a focal length of -12.5 mm

Next, a pair of typical PIV images is shown in Figure 5.47. As it is noted, the



contrast of the images at the upper part is lower than at the bottom. Besides, the diameter of the seeded particles is not uniform.

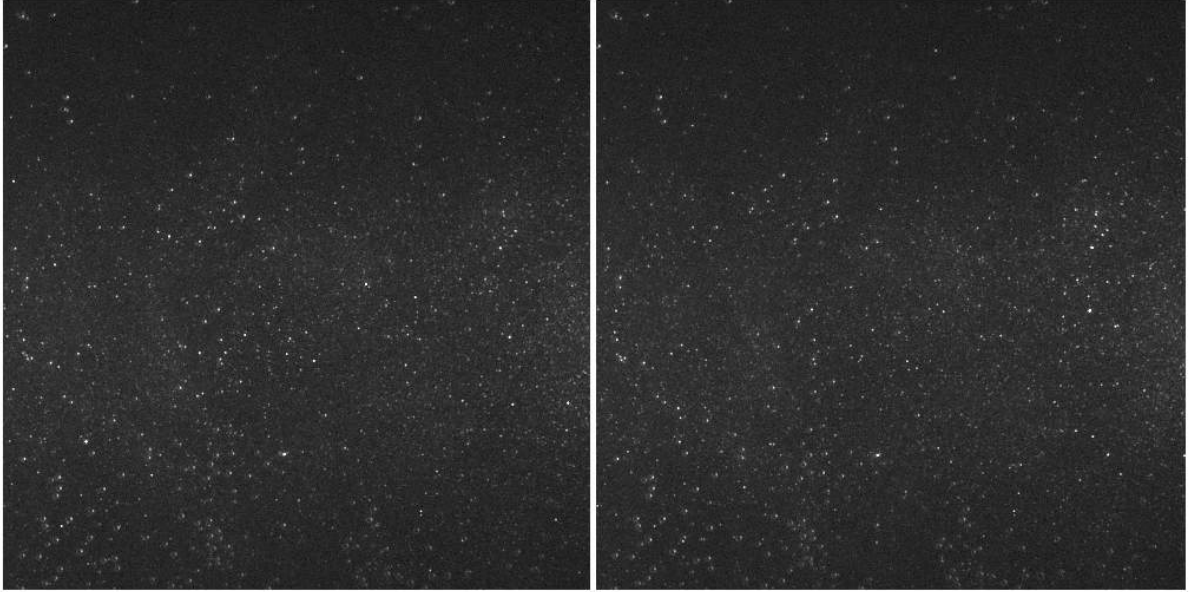


Figure 5.47: Shows two typical images obtained in the experiment.

By applying the three algorithms to the previous images, the following Figure shows the obtained results. The a) part is the result obtained with the correlation technique, b) and c) correspond to the results produced by the Lucas-Kanade algorithm and by the hybrid Lucas-Kanade algorithm, respectively.

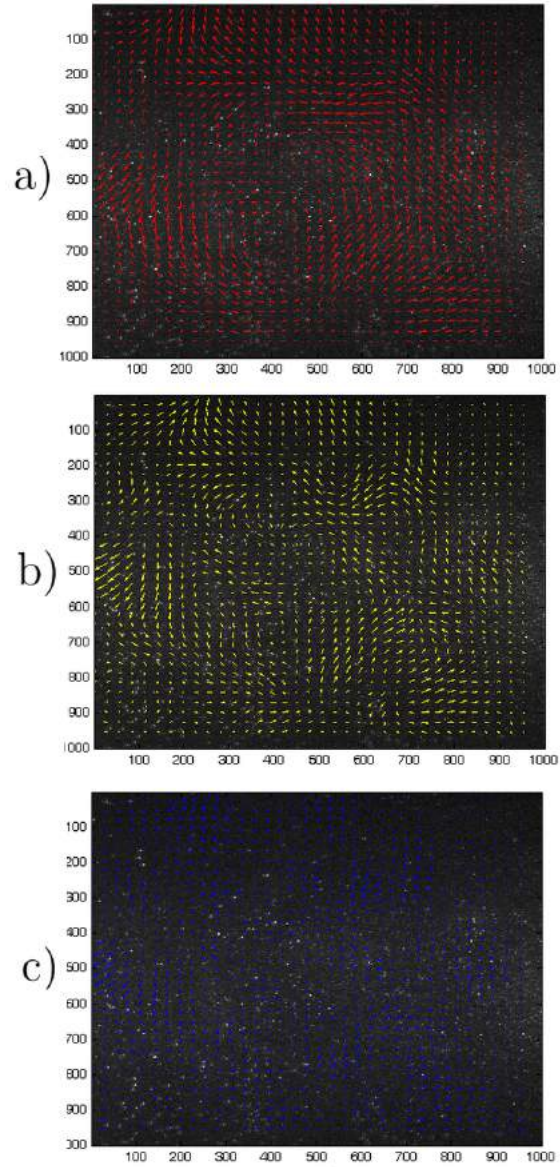


Figure 5.48: Typical experimental result, by a) the correlation technique, b) the Lucas-Kanade algorithm and c) the hybrid Lucas-Kanade algorithm.

As noticed, the three methods give similar results. The discrepancy is less notorious between the Lucas-Kanade and the hybrid methods, since the latter works with the results given by the Lucas-Kanade routine.

## 5.7 Conclusions

In this Section, the main conclusions of this work are listed.

- The variable that shows the most influencing behavior on the accuracy of the methods is the level of spatial details of the images. This parameter is most influenced in turn by the number of particles and the contrast.
- The Gaussian particles are the easiest to implement digitally and in most of the cases are the ones that have best performance.
- The binary and the triangular particles have almost the same performance for all the tests. Their implementation is complicated for sub-pixel displacements.
- Considering the type of displacement, the more challenging is the vortex-like displacement, then the sinusoidal type and at last the constant type (yielding accuracies 50% higher than the other two). For the vortex-like distribution, the error decreases around 10% considering the sinusoidal distribution.
- For all methods, the size of the displacement can't be larger than a quarter of the size of the scan window.
- In general, the correlation method outperforms both the Lucas-Kanade method and its hybrid version, which includes artificial intelligence, by about 50%. The exception particular case to this is for the sinusoidal displacement distribution, where the Lucas-Kanade presents an improvement of 10%.
- The Lucas-Kanade algorithm has better performance than the correlation method in the contrast test. This has to do with the derivatives that the Lucas-Kanade code has implicitly.
- The hybrid version of the Lucas-Kanade method shows an overall improvement of approximately 10% with respect to the standard Lucas-Kanade method.
- The Lucas-Kanade algorithm and the correlation method need a minimum of particles in the image, otherwise the codes collapse.
- A disadvantage of the correlation method is that it produces displacement vectors only for each subwindow. Unlike this, the Lucas-Kanade and its hybrid version yield displacement vectors at each pixel of the image.
- Experimentally, the most difficult variable to control and optimize is the contrast. For example, for relatively large fluid flow velocities, it is complicated to carry out the seeding of particles. Further, when the laser does not have the necessary power, the intensity and the contrast are reduced and the level of noise is amplified.

## 5.8 Future work

In this Section the main points for a future work are listed.

- The next step of this work is to apply a genetic algorithm instead of the swarm particle optimization algorithm.
- Apply artificial intelligence to the correlation algorithm.
- Make a library with this code for a commercial software.

# Bibliography

- [1] M. Raffel, C. Willert and J. Kompenhans, *Particle Image Veocimetry A Practical Guide*. (Springer, 1998).
- [2] R. Adrian and J. Westerweel, *Particle Image Velocimetry*. ( Cambridge University Press, 2011).
- [3] J. Westerweel, *Digital Particle Image Velocimetry Theory and Application*. (Delft University Press, 1993).
- [4] B. Lucas and T. Kanade (1981), “*An iterative image registration technique with an application to stereo vision.*” Workshop, Canada (Proceedings of Imaging Understanding, 1981).
- [5] S. Birchfield , “*Derivation of Kanade-Lucas-Tomasi tracking equation*”. (Private communication, 1997).
- [6] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. (Addison-Wesley,1989).