Software for *Particle Tracking* Version 1.0

J. Martín Pastor
Granular Matter Laboratory
Physics and Applied Mathematics Department
University of Navarra

January, 2007

Índice general

Capítulo 1

Introduction

This is the reference manual for the particle tracking manual. Software is written in MATLAB. The software is able to to obtain simultaneously and automatically the path which follows a set of particles appearing in a collection of images (emph Particle Tracking). In fact, the program is divided into four separate modules. In order to achieve detecting the path followed by each particle must execute a module after another in the following order:

1. param_detect_spot
2. detect_spot
3. enlazar
4. ver_trayectorias

Each module is a MATLAB script which is neecesary to introduce a number of parameters. Next, it will be described each module, explaining what the corresponding utility parameters, and the result of its execution.

Capítulo 2

Parameter election: param_detect_spot

This function is used to select the appropriate parameters for detecting spots of light reflected by each particle with module **detect_spot** (see section 3) in each image of the sequence to be analyzed. It should be noted that each particle should ideally reflect an only single spot likewise bright and gray level. This, in general, no occurs and each particle can reflect several spots of different shape and gray level (see Figure 2.1). This is therefore necessary to implement a set of criteria that we allow to automatically choose the bright spots uniquely characterize each particle. Furthermore, in order to continue reliably the spot characterizing each particle in a sequence of images, can not be moved more than half the particle size of one image to the next.

The criteria used are the most basic in detecting light spots on a dark background in gray scale images.

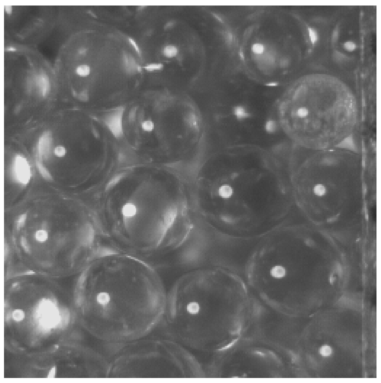

Figura 2.1: In this bright dots appear different each particle

1. Gray level.
2. Spot size
3. Spot eccentricity.
4. Ratio between spot width and height.

The script **param_detect_spot** implements these criteria. To this we must introduce a number of parameters. In order to execute the function, must type in the MATLAB prompt:

>>**param_detect_spot(nfiles,s_raiz,t_1,t_2,area_1,area_2,box,elipsis)**

where:

nfiles The number of images that are tested parameters chosen.

s_raiz Common part in the names of all files images. This is a text variable.

t_1 Minimum gray level of the spots to detect.

t_2 Maximum gray level of the spots to detect.

area_1 Minimum area (in pixels) of the spots to detect.

area_2 Maximum area (in pixels) of the spots to detect. Unless you want to implement this criterion, both **area_1** as **area_2** must assert **0**.

box Maximum ratio between the width and height of spots. If it is desired that this criterion is implemented, **box** must be worth choosing **0**.

elipsis Maximum eccentricity of the spots. Unless it is wanted to implement this criterion, **ellipsis** must assert **1**.

With the aim of learning how to use this module, we will adjust parameters with the image shown in Figure 2.1 automatically detecting spots. The most efficiently is to first adjust the gray levels, the area, the reason between height and width, and eccentricity. It's useful to some program to know the gray level of the pixels that make image (see chapter **??**).

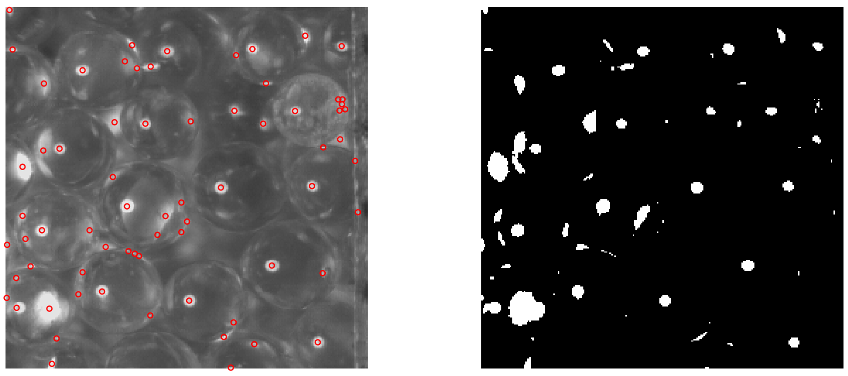

Figura 2.2: On the left, it is an image with the centroids of the found without applying any criterion that the gray level. On the right, it is the binarized image.

2.1. Gray scale thresholds election

When choosing an upper and lower threshold levels for gray [1]: all pixels between these two levels are assigned one and the other zero. Typing in the MATLAB prompt:

```
>>param_detect_spot(1,'imagen_prueba',100,255,0,0,0,1)
```

we will see the original image on the left and right binarized image according to the threshold values for $t\_1 = 100$ and $t\_2 = 255$ that we have chosen (see Figure 2.2). Also on the original image are marked with the red dot centroid position of all of the selected spots. When assigning parameters to other values of 1 or 0, we only binarize the image.

2.2. Minimum and maximum area election

As it is seen in Figure 2.2, some of the spots are found as unwanted reflections and do not represent a particle. Thus, once elected thresholds of gray for binarizing the image, you can select the size of the assigning values to the parameters **area_1** and **area_2**. Typing in the MATLAB prompt:

```
>>param_detect_spot(1,'imagen_prueba',100,255,30,100,0,1)
```

[1]It is important to know if the images is using 8 or 16 binarizando

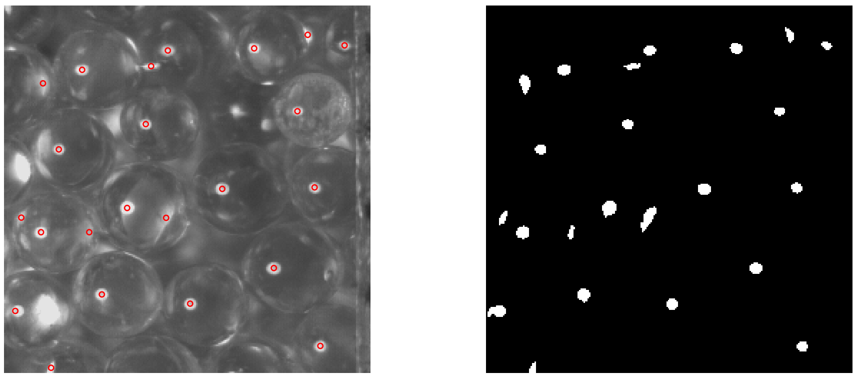

Figura 2.3: On the left, it is an image with the centroids of the spots are found applying the test areas. To the right is the binarized image.

we will see that in the binarized image (see Figure 2.3) have disappeared all spots which area is not in the range $[area\_1, area\_2] = [30, 100]$ pixels. Still, it can be seen from all selected areas, some particles represent and others are unwanted reflections.

2.3. Spot eccentricity election

Of all the spots selected after applying criteria areas, we can easily discriminate based its shape. So, we chose the value of **ellipsis** taking note that the eccentricity of a straight segment is one and a circumference is zero. Typing in the MATLAB prompt:

```
>>param_detect_spot(1,'imagen_prueba',100,255,30,100,0,0.9)
```

see that have disappeared from the binarized image most unwanted bright areas (see Figure 2.4) as the eccentricity of its shape should be less than 0,9.

2.4. Ratio between the width and height of spots election

As seen in Figure 2.5, after applying the criteria and eccentricity areas still appear in the original image bright areas selected by red dots are not representative of the particles and we should remove. To do this, we can choose only the areas shown on the binarized image such that the ratio between height and width of spot must be less than a certain amount. Typing

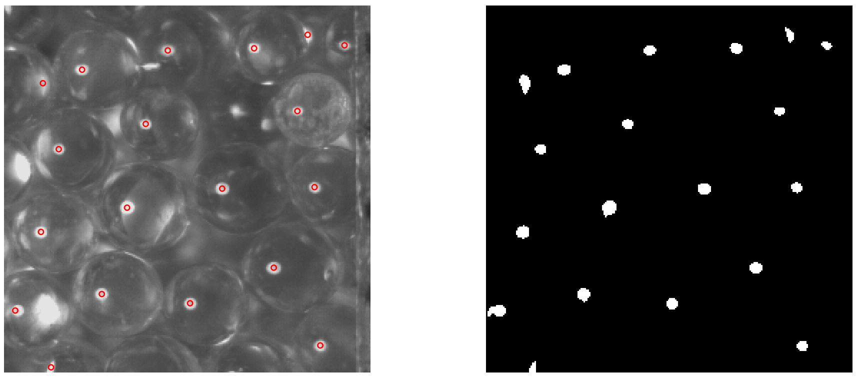

Figura 2.4: On the left, it is an image with the centroids of the spots are found applying the test area and eccentricity. On the right, it is the binarized image.

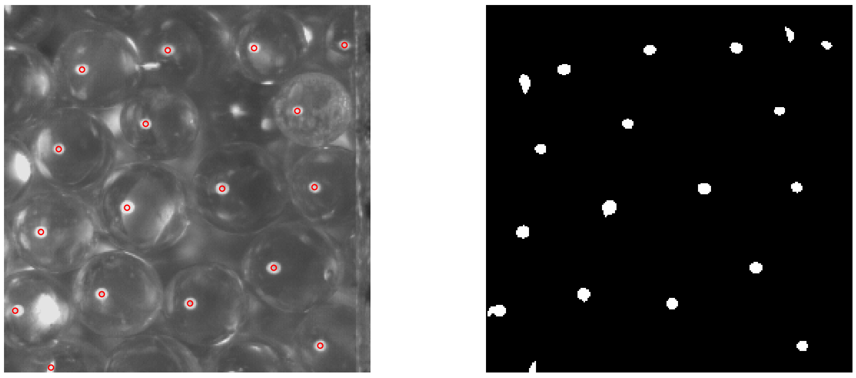

Figura 2.5: On the left, it is an image with the centroids of the spots are found applying the test area, eccentricity and ratio of height and width. On the right, it is the binarized image.

in the MATLAB prompt:

```
>>param_detect_spot(1,'imagen_prueba',100,255,30,100,1.7,0.9)
```

we can see that are missing from the original image red dots corresponding to the binarized image areas that whose ratio between height and width is greater than 1,7.

Be warned that it is not necessary to apply all criteria to analyze an image. In images of sufficient quality, suitably choosing the thresholds may be sufficient to assign each particle a spot.

Finally, the program **param_detect_spot** has also some internal parameter should not be frequently changed.

dir_0 Folder in which software is placed.

dir_1 Folder in which image is placed[2].

format Image files format.

n_number Number of digits to differentiate one from another file.

num Parameter for the early detection of a cluster ($num = 4$) or second neighbors ($num = 8$).

[2]None of these directories are created when you run the function. Unless they exist, the function will fail

Capítulo 3

Spot detection: detect_spot

The function **detect_spot** analyzes the sequence of images in which appearing in the bright spots produced by the reflection of light by a set of particles.

To run this function, type in the MATLAB prompt:

>>**detect_spot(s_raiz,t_1,t_2,area_1,area_2,box,elipsis)**

where:

s_raiz Common part in the names of all files images. This is a text variable.

t_1 Minimum gray level of the spots to detect.

t_2 Maximum gray level of the spots to detect.

area_1 Minimum area (in pixels) of the spots to detect.

area_2 Maximum area (in pixels) of the spots to detect. Unless you want to implement this criterion, both **area_1** as **area_2** must assert **0**.

box Maximum ratio between the width and height of spots. If it is desired that this criterion is implemented, **box** must be worth choosing **0**.

elipsis Maximum eccentricity of the spots. Unless it is wanted to implement this criterion, **ellipsis** must assert **1**.

These parameters are selected in advance using the function **param_detect_spot** (see section 2).

Running this program will get two files, one named "s_raiz_param.dat" containing the parameters we used in **param_detect_spot**, besides the size of the images in pixels, and other "s_raiz_fxyA.dat" containing an ordered data table of nine columns and a number of rows equal the number of spots detected in all and each of the images (this number may be very large).

1ª	**2ª**	**3ª**	**4ª**	**5ª**	**6ª**	**7ª**	**8ª**	**9ª**
image	x	y	A	x_v	y_v	*width*	*height*	*eccentricity*

where

image Image number which appears the spot. If a sequence of images obtained from a video file, depending on the software used for this purpose, the number accompanying "s_raiz" may be different. That is, the program "*VirtualDub*" assign numbers start from zero and "*Photron FASTCAM Viewer*" from one. By default, **detect_spot** begins in one.

x Horizontal coordinate of the centroid in pixels.

y Vertical coordinate of the centroid in pixels. We must that the origin of coordinates for images is the upper left corner.

area Spot area.

x_v Horizontal coordinate of the upper left corner of the smallest of rectangles that contain the spot.

y_v Vertical coordinate of the upper left corner of the smallest of rectangles that contain the spot.

width Width of the smallest of rectangles that contain the spot.

height Lower of the smallest of rectangles that contain the spot.

eccentricity Spot eccentricity.

It should be noted that the spots are detected in the binarized image according to the parameters that we have assigned to **detect_spot**. Therefore, the coordinates of centroids are calculated without taking into account the levels of gray image.

Finally, the program **detect_spot** has also some internal parameter should not be frequently changed.

dir_0 Folder in which software is placed.

dir_1 Folder in which image are placed.

dir_2 Folder in which data are saved[1]..

format Image files format.

n_number Number of digits to differentiate one from another file.

num Parameter for the early detection of a cluster ($num = 4$) or second neighbors ($num = 8$).

refresh Number of images analyzed before stopping to cool the computer.

Pause While the computer remains paused to cool the computer.

[1]None of these directories are created when you run the function. Unless they exist, the function will fail

Capítulo 4

Obtaining individual trajectories: link

The function **link** follows the trajectory of a particle that appears in a sequence of images that have been previously analyzed with the function **detect_spot**. To run this function must type in the MATLAB prompt:

>>link(s_raiz,L,epsilon,jump)

where:

s_raiz Common part in the names of all files images. This is a text variable.

L Minimum number of points to be included in each trajectory.

epsilon Maximum displacement (in pixels) of a particle from one image to the next.

jump Number of images that a particle can disappear in a sequence. Zero is recommended.

To better understand these parameters are going to outline which method to follow a particle and build its trajectory.

- We assume that the data are ordered as explained in chapter 3, i.e. a matrix where each row corresponds to data from only one spot, and the first three columns are respectively the number of image that appears spot, the horizontal coordinate of the centroid and the vertical coordinate.

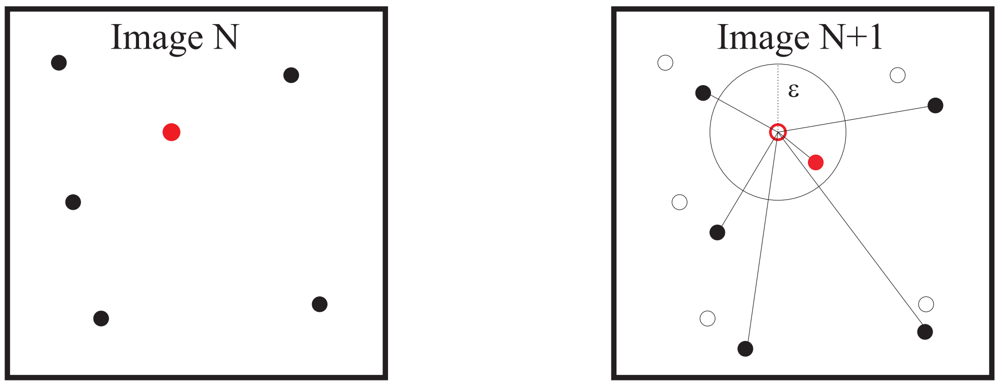

Figura 4.1: On the left figure is depicted a schematic of the position of the *spots corresponding to the image* N. Which is marked in red corresponds to the particle being tracked. On right figure is shown the spots of $N+1$ as solid dots and the image of N as empty dots. The empty dot marked in red is the position of the particle in the image N. The position of the particle in the image $N+1$ is the solid dot in red within the circle of radius ϵ.

- The first step is to know what all the rows of the data matrix corresponding to each image. For example, in row 1 through 16 are the data of the image number 1 of row 17 to 33 are those of the second image, and so on.

- Data are selected for a first row of the matrix, these data call them spot of the first image. Then, comparing with the corresponding rows to the next image, which seeks the closest spots , and this is the particle that we are following in this second image. Also the distance between the spot of the first and second image should be less than “epsilon” (see Figure 4.1). Following this last condition of proximity, the emph spot of the second image becomes the first picture and iterate the process.

- If “jump” is zero, the trajectory a particle build stops when the separation between the spots closest two consecutive images is greater than ϵ. If “jump” is nonzero is allowed that is not the spot on a number of images equal to “jump”.

- All data belonging to any path are marked to avoid repeating them in the search for new paths in order to expedite the process. item When finished constructing a trajectory is chosen a new starting point and repeat the whole process. No paths are considered those that do not have a number of points greater than or equal to “L”.

As a result of the execution of this function yields the following.

"Cutting" files: s_raiz_cut.dat

This file contains three columns. The first is the number of image in the sequence. The second is the initial row within the image data array. The third is the final row ara the image within the data matrix.

Image	**Initial**	**Final**
1	1	16
2	17	33
3	34	45
⋮	⋮	⋮

Trajectory files: s_raiz_t*.dat

For each particle that has recognized the function **link**, there is a file that contains enough data to reconstruct the path followed by the particle as a function of time. The data structure is as follows.

1ª	**2ª**	**3ª**	**4ª**	**5ª**	**6ª**	**7ª**	**8ª**	**9ª**
imagen	x	y	A	x_v	y_v	*width*	*height*	*eccentricity*

where

image Number of the image that appears in the spot of the particle. You must know the rate imaging to determine the point in time where it belongs.

x Horizontal coordinate of the centroid in pixels.

y Vertical coordinate of the centroid in pixels. We must that the origin of coordinates for images is the upper left corner.

area Spot area.

x_v Horizontal coordinate of the upper left corner of the smallest of rectangles that contain the spot.

y_v Vertical coordinate of the upper left corner of the smallest of rectangles that contain the spot.

width Width of the smallest of rectangles that contain the spot.

height Lower of the smallest of rectangles that contain the spot.

eccentricity Spot eccentricity.

When “jump” is nonzero, any of the points that has appeared in the file path could not be real. This is because of a particle has lost a number of images smaller or equal to “jump”, the trajectory points corresponding to these lost images were calculated from linear interpolation with the nearest point these. To distinguish these points, the remaining data other than the centroid coordinates are zero.

Index files: s_raiz_ind.dat

This file is a column of integers. Each row represents the file “s_raiz_fxyA.dat” belonging to any of all possible trajectories analyzed.

Parameters file: s_raiz_Les.dat

This file stores the parameters that have been constructed trajectories, i.e. “L”, “epsilon” and “jump”.

Other parameters

El programa **enlazar** utiliza otros parámetros internos que no es necesario variarlos a menudo. Son los siguientes:

dir_0 Folder in which software is placed.

dir_1 Folder in which datar resulting from “linking” is save.

dir_2 Folder in which data is saved.

dir_3 Folder in which image is placed[1].

n_number Number of digits to differentiate one from another file.

format Image files format.

coletilla File extension that containing the data from the detection of the spots.

Pausar While the computer remains in pause between the reconstruction of two paths.

[1]None of these directories are created when you run the function. Unless they exist, the function will fail

IMPORTANT

Importantly, by changing the parameters of the function **link**, the number of paths found for the same sequence of images may vary and duplication problems appear. To avoid them, every time you execute this function, the following files will be automatically deleted: s_raiz_cut.dat, s_raiz_t *.dat, s_raiz_ind.dat and s_raiz_Les.dat.

We should also note that by construction, this function can reconstruct the trajectories followed a collection of particles contained in an image sequence, whichever method was used to detect the positions thereof, provided that the data has been stored similar manner as was done with the file "s_raiz_fxyA.dat" (see chapter 3).

Capítulo 5

Viewing paths: ver_trayectorias

This program is used to display each path built with the script **link** (see section 4). To run it simply type in the MATLAB prompt:

>>**ver_trayectorias(s_raiz,ind)**

were:

s_raiz Common part in the names of all files images. This is a text variable.

ind Image sequence that we want to represent (recommended one).

When running the function, the centroid coordinates in function of time is represented , and the particle that has followed that path marked with a red circle for each path that has been built (see Figure 5.1).

After each path represented, all paths that have been found are shown together in different colors. In this image can cross paths because there are not represented in function of time (see figure 5.2).

The program **ver_trayectorias** uses other internal parameters do not need to change them often. They are:

dir_0 Folder in which software is placed.

dir_1 Folder in which data are saved.

dir_2 Folder in which path files are saved.

dir_3 Folder in which image are placed[1].

[1]None of these directories are created when you run the function. Unless they exist, the function will fail

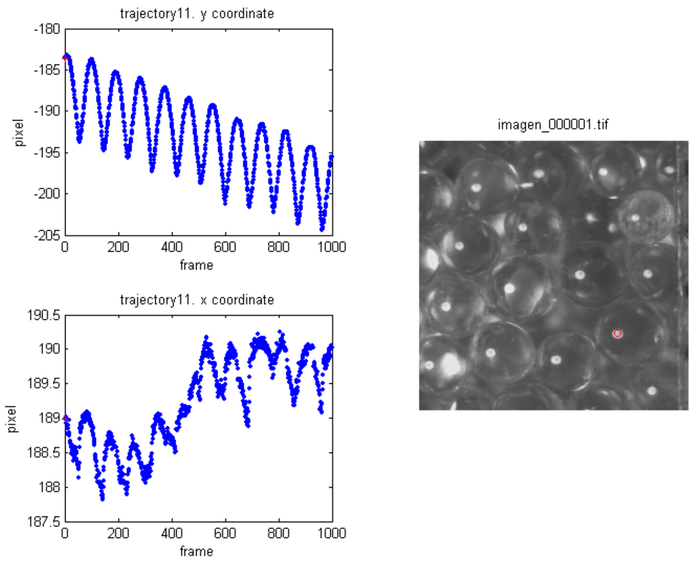

Figura 5.1: Top left, vertical coordinate of the centroid in function of time. Bottom left, horizontal coordinate of the centroid in function of time. In both figures, the red cross marks the spot "ind" in the whole sequence of images. Right, "ind" image sequence in which the particle appears marked with a red circle.

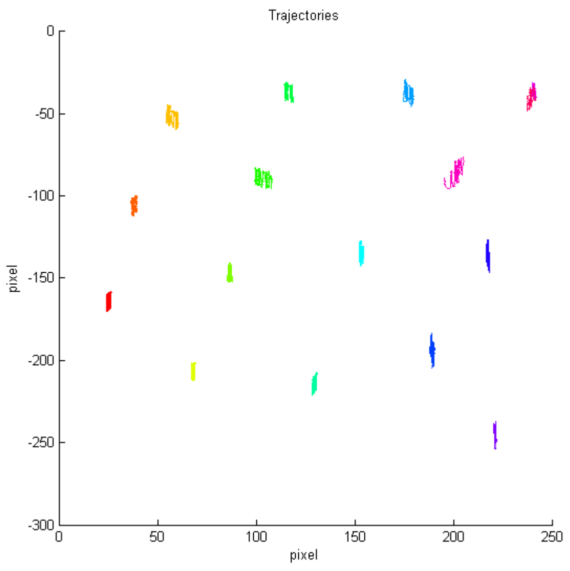

Figura 5.2: The full set of the trajectories

n_number Number of digits to differentiate one from another file.

format Image files format.