Software de *Particle Tracking*
Version 1.0

Martín Pastor
Laboratorio de Medios Granulares
Departamento de Física y Matemática Aplicada
Universidad de Navarra

Enero, 2007

Índice general

Capítulo 1

Introducción

Éste es el manual del programa hecho en MATLAB con el que se consigue obtener simultáneamente y de manera automática la trayectoria que sigue un conjunto de partículas que aparecen en una colección de imágenes (*Particle Tracking*). En realidad, el programa se divide en cuatro módulos independientes. Para conseguir detectar la trayectoria que sigue cada partícula se debe ejecutar un módulo tras otro en el siguiente orden:

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

Cada módulo es un *script* de MATLAB al que es necesario introducir una serie de parámetros. A continuación, se describirá cada módulo, explicando cual es la utilidad de sus correspondientes parámetros, y el resultado de su ejecución.

Capítulo 2

Elección de parámetros: param_detect_spot

Esta función sirve para elegir los parámetros adecuados para detectar los *spots* de luz que refleja cada partícula con el módulo **detect_spot** (ver capítulo 3) en cada imagen de la secuencia a analizar. Hay que hacer notar que idealmente cada partícula debería reflejar una única mancha brillante de la misma forma y nivel de gris. Esto, en general, no ocurre y cada partícula puede reflejar varios *spots* de distinta forma y nivel de gris (ver figura 2.1). Es por tanto necesario implementar una serie de criterios que nos permitan elegir de manera automática los puntos brillantes que caracterizan unívocamente cada partícula. Además, para poder seguir de manera fiable el *spot* que caracteriza cada partícula en una secuencia de imágenes, no puede desplazarse más de la mitad del tamaño de la partícula de una imagen a la siguiente.

Los criterios que se utilizan son los más básicos en la detección de manchas claras sobre fondo oscuro en imágenes de escala de grises.

1. El nivel de gris.
2. El tamaño del *spot*.
3. La excentricidad del spot.
4. La relación entre la anchura y altura del spot.

El *script* **param_detect_spot** implementa estos criterios. Para ello hay que introducir una serie de parámetros. Al ejecutar la función debemos teclear en el *prompt* de MATLAB:

>>**param_detect_spot(nfiles,s_raiz,umbral_1,umbral_2,area_1,area_2,caja,elipse)**

donde:

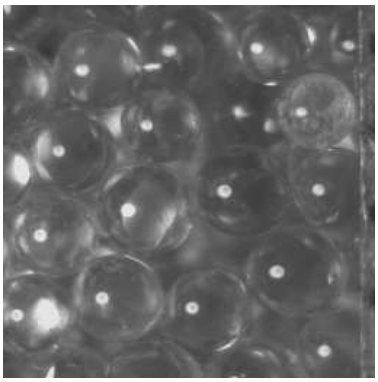

Figura 2.1: En esta imagen aparecen distintos puntos brillantes en cada partícula

nfiles Es el número de imágenes en las que se ensayan los parámetros elegidos.

s_raiz Parte común en el nombre de todos los archivos de imágenes. Esta es una variable de texto.

umbral_1 Mínimo nivel de gris de los *spots* a detectar.

umbral_2 Máximo nivel de gris de los *spots* a detectar.

area_1 Área mínima (en pixels) de los *spots* a detectar.

area_2 Área máxima (en pixels) de los *spots* a detectar. Si no se desea que se implemente este criterio, tanto **area_1** como **area_2** deben valer **cero**.

caja Cociente máximo entre la anchura y altura de los *spots*. Si no se desea que se implemente este criterio, **caja** debe valer elegir **cero**.

elipse Máxima excentricidad de los *spots*. Si no se desea que se implemente este criterio, **elipse** debe valer **uno**.

Para aprender como se utiliza este módulo vamos a ajustar los parámetros con la imagen que aparece en la figura 2.1 para detectar automáticamente los *spots*. La manera más eficiente es ajustar primero los niveles de gris, el área, la razón entre alto y ancho, y la excentricidad. Es muy útil utilizar algún programa para conocer el nivel de gris de los píxeles que componen la imagen (ver capítulo 6).

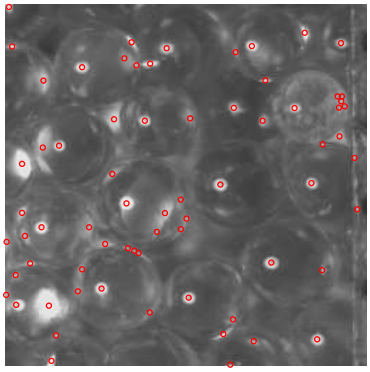
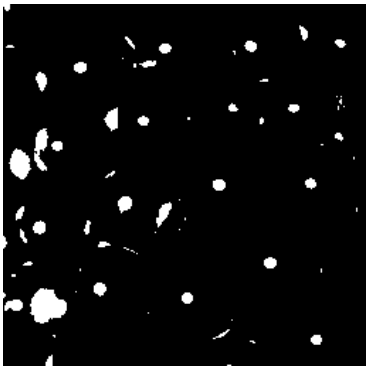

Figura 2.2: A la izquierda se observa la imagen con los centroides de los *spots* encontrados sin aplicar ningún criterio más que el nivel de gris. A la derecha aparece la imagen binarizada.

2.1. Elección de los umbrales de gris

Al elegir un umbral superior y otro inferior para los niveles de gris[1] estamos binarizando la imagen: a todos los puntos entre estas dos cotas se les asigna un uno y al resto cero. Tecleando en el prompt de MATLAB:
>>param_detect_spot(1,'imagen_prueba',100,255,0,0,0,1)
veremos la imagen original a la izquierda y a la derecha la imagen binarizada de acuerdo con los valores para $umbral\_1 = 100$ y $umbral\_2 = 255$ que hemos elegido (ver figura 2.2). Además sobre la imagen original aparecen marcados con punto rojos la posición de todos los centroides de los *spots* seleccionados. Al asignar al resto de parámetros los valores de 1 o 0, sólo estamos binarizando la imagen.

2.2. Elección del área máxima y mínima

Como se puede ver en la figura 2.2, algunos de los *spots* encontrados son reflejos no deseados y no representan a una partícula. De esta manera, una vez elegidos los umbrales de gris para binarizar la imagen, podemos seleccionar el tamaño de los *spots* asignando valores a los parámetros **area_1** y **area_2**. Tecleando en el prompt de MATLAB:
>>param_detect_spot(1,'imagen_prueba',100,255,30,100,0,1)
vemos que en la imagen binarizada (ver figura 2.3) han desaparecido todos los *spots* que cuyo área no está comprendida en el intervalo $[area\_1, area\_2] =$

[1] Es importante conocer si las imágenes son de 8 ó 16 bits

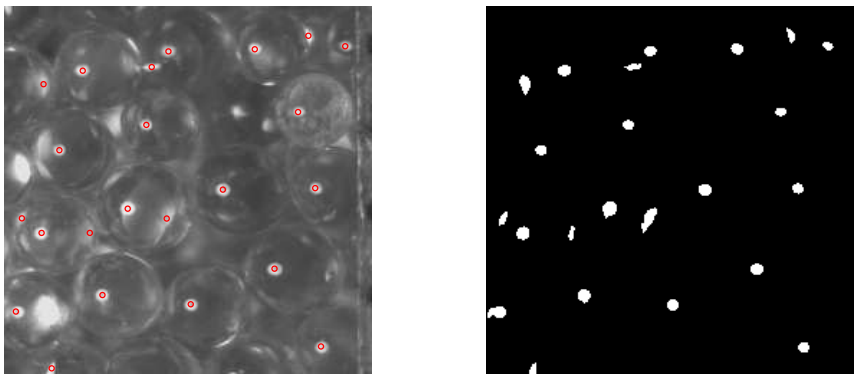

Figura 2.3: A la izquierda se observa la imagen con los centroides de los *spots* encontrados aplicando el criterio de las áreas. A la derecha aparece la imagen binarizada.

$[30, 100]$ pixels. Aún así, se puede observar de todas las áreas seleccionadas, algunas representan a las partículas y otras son reflejos no deseados.

2.3. Elección de la excentricidad del *spot*

De todas los *spots* seleccionados después de aplicar el criterio de las áreas, podemos discriminarlas fácilmente en función de su forma. Así, elegimos el valor de **elipse** teniendo en cuenta que la excentricidad de un segmento recto es uno y la de una circunferencia es cero. Tecleando en el prompt de MATLAB:
>>**param_detect_spot(1,'imagen_prueba',100,255,30,100,0,0.9)**
vemos que han desaparecido de la imagen binarizada la mayoría de las zonas brillantes no deseadas (ver figura 2.4) ya que la excentricidad de su forma debe ser menor que 0,9.

2.4. Elección del cociente entre altura y anchura del *spot*

Como se puede ver en la figura 2.5, tras aplicar el criterio de las áreas y la excentricidad, todavía aparecen en la imagen original áreas brillantes seleccionadas por puntos rojos que no son representativas de las partículas y que debemos eliminar. Para ello, podemos escoger únicamente las áreas que aparecen en la imagen binarizada tales que el cociente entre la altura y la

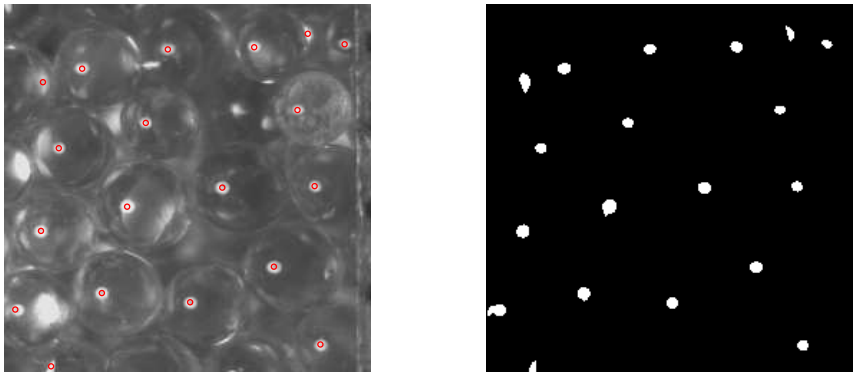

Figura 2.4: A la izquierda se observa la imagen con los centroides de los *spots* encontrados aplicando el criterio de las áreas y de la excentricidad. A la derecha aparece la imagen binarizada.

anchura del *spot* debe ser menor que cierta cantidad. Tecleando en el prompt de MATLAB:
>>**param_detect_spot(1,'imagen_prueba',100,255,30,100,1.7,0.9)**
vemos que han desaparecido de la imagen original los puntos rojos que corresponden a las áreas de la imagen binarizada que cuyo razon entre altura y anchura es mayor que 1,7.

Se advierte que no es necesario para aplicar todos los criterios para analizar una imagen. En imágenes de la suficiente calidad, eligiendo adecuadamente los umbrales, puede ser suficiente para asignar a cada partícula un *spot*.

Por último, el programa **param_detect_spot** tiene también algún parámetro interno que no es necesario modificar frecuentemente:

dir_0 Directorio donde se encuentra el programa.

dir_1 Directorio donde se encuentran las imágenes a analizar[2].

formato Formato de los archivos de imagen.

n_number Número de dígitos que diferencian un archivo de otro.

num Parámetro para la detección de cluster a primeros ($num = 4$) o segundos vecinos ($num = 8$).

[2]Ninguno de estos directorios se crea al ejecutar la función. Si no existen la función dará un error

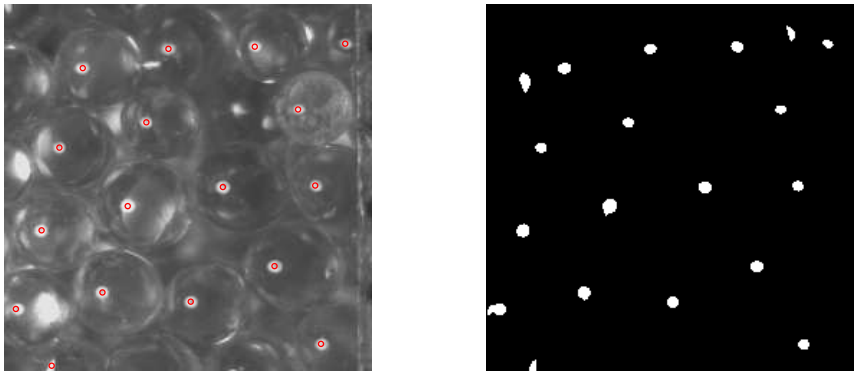

Figura 2.5: A la izquierda se observa la imagen con los centroides de los *spots* encontrados aplicando el criterio de las áreas, de la excentricidad y de la razón de altura y anchura. A la derecha aparece la imagen binarizada.

Capítulo 3

Detección de los *spots*: detect_spot

La función **detect_spot** analiza una secuencia de imágenes en la que aparecen *spots* brillantes producidos por la reflexión de luz de un conjunto de partículas.

Para ejecutar la función hay que teclear en el prompt de MATLAB:
>>**detect_spot(s_raiz,umbral_1,umbral_2,area_1,area_2,caja,elipse)**
donde:

s_raiz Parte común en el nombre de todos los archivos de imágenes. Esta es una variable de texto.

umbral_1 Mínimo nivel de gris de los *spots* a detectar.

umbral_2 Máximo nivel de gris de los *spots* a detectar.

area_1 Área mínima (en pixels) de los *spots* a detectar.

area_2 Área máxima (en pixels) de los *spots* a detectar. Si no se desea que se implemente este criterio, tanto **area_1** como **area_2** debe valer **cero**.

caja Cociente máximo entre la anchura y altura de los *spots*, y viceversa. Si no se desea que se implemente este criterio, **caja** debe valer **cero**.

elipse Máxima excentricidad de los *spots*. Si no se desea que se implemente este criterio, **elipse** debe valer **uno**.

Estos parámetros se eligen previamente utilizando la función **param_detect_spot** (ver capítulo 2).

Al ejecutar este programa obtendremos dos archivos, uno con nombre "s_raiz_param.dat" que contiene los parámetros que hemos utilizado en **param_detect_spot**, además del tamaño de las imágenes en pixels, y el otro "s_raiz_fxyA.dat" que contiene una tabla de datos ordenada en nueve columnas y un número de filas igual al número de *spots* detectados en todas y cada una de las imágenes (este número puede ser muy grande).

1ª	**2ª**	**3ª**	**4ª**	**5ª**	**6ª**	**7ª**	**8ª**	**9ª**
imagen	x	y	A	x_v	y_v	*ancho*	*alto*	*excentricidad*

donde

imagen Número de la imagen en la que aparece el *spot*. Si se obtiene una secuencia de imágenes a partir de un archivo de vídeo, en función del software que se utilice para este fin, el número que acompaña a "s_raiz" puede ser distinto. Esto es, el programa "*VirtualDub*" comienza a asignar números desde cero y el "*Photron FASTCAM Viewer*" desde uno. Por defecto, **detect_spot** comienza en uno.

x Coordenada horizontal del centroide en pixels.

y Coordenada vertical del centroide en pixels. Debemos tener en cuenta que el origen de coordenadas para el caso de imágenes es el vértice superior izquierdo.

area Área del *spot*.

x_v Coordenada horizontal del vértice superior izquierdo del menor de los rectángulos que contiene el *spot*.

y_v Coordenada vertical del vértice superior izquierdo del menor de los rectángulos que contiene el *spot*.

ancho Anchura del menor de los rectángulos que contiene el *spot*.

alto Altura del menor de los rectángulos que contiene el *spot*.

excentricidad Excentricidad del *spot*.

Se debe tener en cuenta que los *spots* se detectan en la imagen binarizada de acuerdo con los parámetros que le hemos asignado a **detect_spot**. Por tanto, las coordenadas de los centroides se calculan sin tener en cuenta los niveles de gris de la imagen.

El programa **detect_spot** también tiene algún parámetro interno que no es necesario modificar frecuentemente:

dir_0 Directorio donde se encuentra el programa.

dir_1 Directorio donde se encuentran las imágenes a analizar.

dir_2 Directorio donde se guardarán los datos correspondientes a la imágenes analizadas[1].

formato Formato de los archivos de imagen.

n_number Número de dígitos que diferencian un archivo de otro.

num Parámetro para la detección de cluster a primeros o segundos vecinos.

refresco Número de imágenes que analiza antes de pausar para enfriar el ordenador.

Pausar Tiempo que el ordenador permanece en pausa para enfriar el ordenador.

[1]Ninguno de estos directorios se crea al ejecutar la función. Si no existen la función dará un error

Capítulo 4

Obtención de la trayectorias individuales: enlazar

La función **enlazar** sigue la trayectoria de una partícula que aparece en una secuencia de imágenes que ha sido analizada previamente con la función **detect_spot**. Para ejecutar esta función debemos teclear en el prompt de MATLAB:
>>**enlazar(s_raiz,L,epsilon,salto)**
donde:

s_raiz Parte común en el nombre de todos los archivos de imágenes. Esta es una variable de texto.

L Número mínimo de puntos que debe contener cada trayectoria.

epsilon Máximo desplazamiento (en pixels) de una partícula de una imagen a la siguiente

salto Número de imágenes que una partícula puede desaparecer en una secuencia. Se recomienda cero.

Para comprender mejor estos parámetros se va a esbozar cual es el método para seguir una partícula y construir su trayectoria.

- Partimos del supuesto que los datos están ordenados según se explico en el capítulo 3, es decir, una matriz donde cada fila corresponde a los datos de un *spot*, y las tres primeras columnas son respectivamente el número de imagen en la que aparece el *spot*, la coordenada horizontal del centroide y la coordenada vertical.

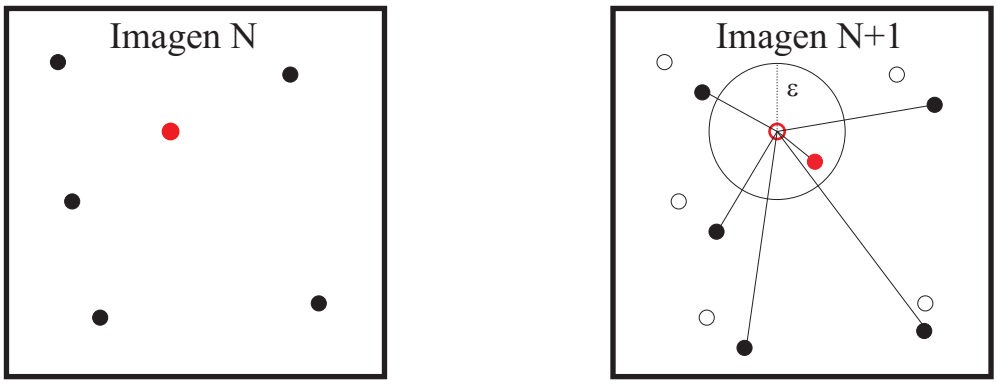

Figura 4.1: En la figura de la izquierda aparece representado un esquema de la posición de los *spots* correspondientes a la imagen N. El que está marcado en rojo corresponde a la partícula que se está siguiendo. En la figura de la derecha aparecen los *spots* de la imagen $N+1$ solidos y los de la imagen N huecos. El *spot* hueco que está marcado en rojo es la posición de la partícula en la imagen N. La posición de la partícula en la imagen $N+1$ es el *spot* sólido marcado en rojo dentro del circulo de radio ϵ.

- El primer paso es conocer cuales de las filas de toda la matriz de datos corresponden a cada imagen. Por ejemplo, de la fila 1 a la 16 son los datos de la imagen número 1; de la fila 17 a la 33 son los de la segunda imagen; y así sucesivamente.

- Se eligen los datos correspondientes una primera fila de la matriz, a estos datos les llamaremos *spot* de la primera imagen. A continuación, comparando con las filas correspondientes a la siguiente imagen, se busca cual de todos los *spots* es el más cercano, siendo éste la partícula que estamos siguiendo en esta segunda imagen. Además la distancia entre el spot de la primera y la segunda imagen deben ser menor que "epsilon" (ver figura 4.1). Si se cumple esta ultima condición de proximidad, el *spot* de la segunda imagen pasa a ser de la primera imagen e iteramos el proceso.

- Si "salto" es igual a cero, la trayectoria que sigue una partícula se deja de construir cuando la separación entre los *spots* más próximos de dos imágenes consecutivas es mayor que ϵ. Si "salto" es distinto de cero se permite que no se encuentre el *spot* en un número de imágenes igual a "salto".

- Todos los datos que pertenecen a cualquier trayectoria se marcan para no repetirlos en la búsqueda de nuevas trayectorias con el objetivo de

acelerar el proceso.

- Cuando se termina de construir una trayectoria se elige un nuevo punto inicial y se repite todo el proceso. No se consideran trayectorias aquellas que no tienen un número de puntos mayor o igual que "L".

Como resultado de la ejecución de esta función se obtienen los siguientes archivos.

Archivo de "cortes": s_raiz_cut.dat

Este archivo contiene tres columnas. La primera es el número de imagen de la secuencia. La segunda es el la fila inicial para la imagen dentro de la matriz de datos. La tercera es la fila final ara la imagen dentro de la matriz de datos.

Imagen	**Inicial**	**final**
1	1	16
2	17	33
3	34	45
⋮	⋮	⋮

Archivos de trayectorias: s_raiz_t*.dat

Para cada partícula que ha reconocido la función **enlazar** existe un archivo que contiene datos suficientes como para reconstruir la trayectoria que ha seguido la partícula en función del tiempo. La estructura de datos es la siguiente.

1ª	**2ª**	**3ª**	**4ª**	**5ª**	**6ª**	**7ª**	**8ª**	**9ª**
imagen	x	y	A	x_v	y_v	*ancho*	*alto*	*excentricidad*

donde

imagen Número de la imagen en la que aparece el *spot* de la partícula. Se debe conocer la tasa de adquisición de imágenes para conocer el instante de tiempo que le corresponde.

x Coordenada horizontal del centroide en pixels.

y Coordenada vertical del centroide en pixels. Se debe tener en cuenta que el origen de coordenadas para el caso de imágenes es el vértice superior izquierdo.

area Área del *spot*.

x_v Coordenada horizontal del vértice superior izquierdo del menor de los rectángulos que contiene el *spot*.

y_v Coordenada vertical del vértice superior izquierdo del menor de los rectángulos que contiene el *spot*.

ancho Anchura del menor de los rectángulos que contiene el *spot*.

alto Altura del menor de los rectángulos que contiene el *spot*.

excentricidad Excentricidad del *spot*.

Cuando "salto" es distinto de cero, puede que alguno de los puntos que ha aparecido en el archivo de trayectorias no sea real. Esto se debe a que si se ha perdido una partícula un número de imágenes menor o igual que "salto", los puntos de la trayectoria que corresponden a estas imágenes perdidas se han calculado a partir de la interpolación lineal con los punto más próximos a estos. Para distinguir estos puntos, el resto de datos distintos a las coordenadas del centroide son cero.

Archivos de indices: s_raiz_ind.dat

Este archivo es una columna de números enteros. Cada uno de ellos representa la fila del archivo "s_raiz_fxyA.dat" que pertenece a cualquiera de todas las posibles trayectorias analizadas.

Archivo de parámetros: s_raiz_Les.dat

Este archivo almacena los parámetros con los que se han construido las trayectorias, esto es:"L", "epsilon" y "salto".

Otros parámetros

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

dir_0 Directorio donde se encuentra el programa.

dir_1 Directorio donde se encuentran los datos a analizar.

dir_2 Directorio donde se guardarán de las trayectorias, etc.

dir_3 Directorio donde se encuentran las imágenes analizadas[1].

[1]Ninguno de estos directorios se crea al ejecutar la función. Si no existen la función dará un error

n_number Número de dígitos que diferencian un archivo de otro.

formato Formato de los archivos de imagen.

coletilla Extensión del archivo que contiene los datos provenientes de la detección de los *spots*.

Pausar Tiempo que el ordenador permanece en pausa entre la reconstrucción de dos trayectorias.

Advertencias importantes

Es importante destacar que al cambiar los parámetros de la función **enlazar**, el número de trayectorias encontradas para la misma secuencia de imágenes puede variar y aparecer problemas de duplicación. Para evitarlos, cada vez que se ejecute esta función, se borrarán automáticamente los archivos: s_raiz_cut.dat, s_raiz_t*.dat, s_raiz_ind.dat y s_raiz_Les.dat.

También debemos advertir, que por construcción, esta función puede reconstruir las trayectorias que han seguido un conjunto de partículas que aparecen en una secuencia de imágenes independientemente del método que se haya utilizado para detectar las posiciones de éstas, siempre que los datos se hayan almacenado de manera similar a como se ha hecho con el archivo "s_raiz_fxyA.dat" (ver capítulo 3)

Capítulo 5

Visualización de las trayectorias: ver_trayectorias

Este programa sirve para visualizar cada trayectoria construidas con el programa **enlazar** (ver capítulo 4). Para ejecutarlo basta con teclear en el prompt de MATLAB:
>>**enlazar(s_raiz,ind)**
donde:

s_raiz Parte común en el nombre de todos los archivos de imágenes. Esta es una variable de texto.

ind Imagen de la secuencia que queremos representar (se recomienda uno).

Al ejecutar la función aparece representadas las coordenadas del centroide en función del tiempo, y la partícula que ha seguido esa trayectoria marcada con un círculo rojo para cada trayectoria que se ha construido (ver figura 5.1).

Tras haber representado cada trayectoria, se representan juntas todas las trayectorias que se han encontrado en diferentes colores. En esta imagen las trayectorias se pueden cruzar porque no se representan en función del tiempo (ver figura 5.2).

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

dir_0 Directorio donde se encuentra el programa.

dir_1 Directorio donde se encuentran los datos.

dir_2 Directorio donde se guardan de las archivos de trayectorias

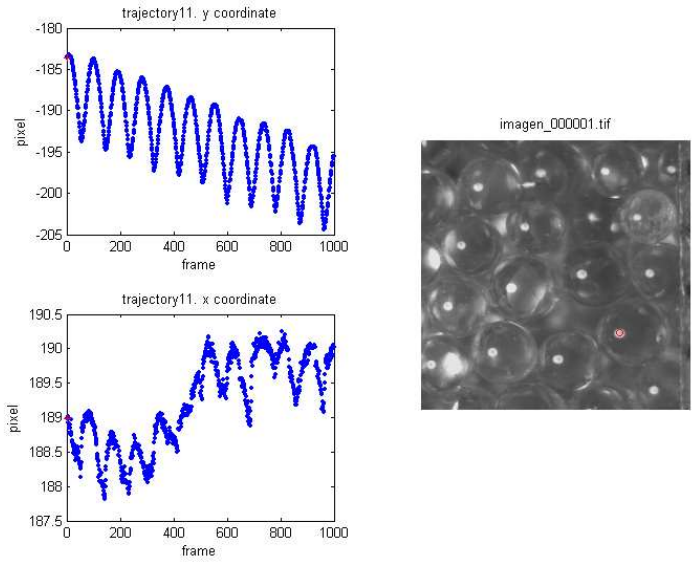

Figura 5.1: Izquierda arriba, coordenada vertical del centroide en función del tiempo. Izquierda abajo, coordenada horizontal del centroide en función del tiempo. En ambas figuras, la cruz roja marca el punto "ind" de toda la secuencia de imágenes. Derecha, imagen "ind" de la secuencia en la que aparece la partícula marcada con un círculo rojo.

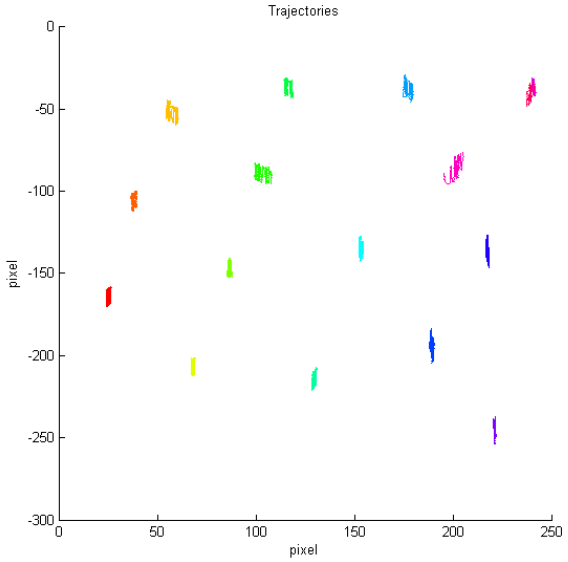

Figura 5.2: Todas las trayectorias encontradas

dir_3 Directorio donde se encuentran las imágenes analizadas[1].

n_number Número de dígitos que diferencian un archivo de otro.

formato Formato de los archivos de imagen.

[1]Ninguno de estos directorios se crea al ejecutar la función. Si no existen la función dará un error

Capítulo 6

Niveles de gris.

Si no se dispone de un software con el conocer cual es el nivel de gris de los pixels de una imagen, se puede utilizar el script **gray_scale**. Al teclear en el prompt de MATLAB:
>>**gray_scale(s_raiz)**
nos da a elegir la imagen que deseemos de la secuencia que tiene como parte común en el nombre "s_raiz". Cada vez que se haga click con el botón izquierdo del ratón sobre cualquier punto, en el prompt de MATLAB aparecerá el nivel de gris del punto y su posición en pixels. Para terminar, se debe hacer click con el botón derecho. Este script puede ser muy útil para ajustar los parámetros en la función **param_detect_spot**.

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

dir_0 Directorio donde se encuentra el programa.

dir_1 Directorio donde se encuentran las imágenes analizadas[1].

n_number Número de dígitos que diferencian un archivo de otro.

formato Formato de los archivos de imagen.

Por último, también se dispone de una mejora del programa **detect_spot**, es **detect_spot_gray**. Este último calcula los centroides de los *spot* teniendo en cuenta los niveles de gris de las imágenes. Los parámetros que se deben utilizar son los mismos que en **detect_spot** y los datos que devuelve están ordenados de la misma manera (ver capítulo 3)

[1]Ninguno de estos directorios se crea al ejecutar la función. Si no existen la función dará un error