UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

GRADO EN INGENIERÍA AEROESPACIAL

Trabajo de Fin de Grado

NOMBRE DEL PROYECTO:

Desarrollo de una interfaz de seguimiento para un PIV. Análisis para uso docente.

Nombre del alumno: Carlos Guirado Navarro

Supervisado por: Jaime Riera Guasp y Miguel Ardid Ramírez

Departamento de Física Aplicada de la ETSID

Agradecimientos

Quisiera agradecer a los tutores Miguel Ardid Ramírez y Jaime Riera Guasp del Departamento de Física Aplicada de la Escuela Técnica Superior de Ingeniería del Diseño su paciencia y dedicación en cada momento que me ha sido necesario durante la realización de este Trabajo de Fin de Grado.

De esta manera, realizar este trabajo supone un buen final en mis estudios del Grado en Ingeniería Aeroespacial pues, a pesar de las dificultades en cuanto al tiempo y la obtención de los resultados deseados, la ayuda de los tutores y las instalaciones del laboratorio han hecho posible implementar y diseñar correcta y completamente todo lo necesario para cumplir los objetivos.

Resumen

Este trabajo se dedica al diseño e implementación de varias interfaces con MATLAB que simulan el software y el comportamiento de la técnica PIV (Velocimetría de Imágenes de Partículas). Asi pues, está dividido en tres interfaces distintas.

En la primera interfaz, [Cargarvideopiv.m], se carga el vídeo y se consiguen dos capturas con un incremento de tiempo. En la segunda, [Pruebaestudioimagenes.m], se selecciona la región que se desea estudiar y se aplica el algoritmo de correlación. Por último, en la tercera interfaz, [Resultados.m], se muestran los resultados obtenidos tanto en modo de campo de vectores como en modo de campo escalar (diferentes tonalidades de colores).

Conociendo previamente la teoría y ejemplos de cómo emplear esta técnica, el software permite obtener resultados susceptibles de ser entendidos e interpretados mediante conocimientos teóricos. Finalmente, se ha adaptado y utilizado una instalación piloto de un PIV con un fluido dopado con partículas para obtener grabaciones en las que aplicar el sofware desarrollado y demostrar así su funcionalidad en condiciones reales.

Abstract

This work is dedicated to the design and implementation of several interfaces with MATLAB that simulate the software and behavior of the PIV technique (Velocimetry of Images of Particles). Thus, it's divided into three distinct interfaces: Cargarvideopiv.m, Pruebaestudioimagenes.m and Resultados.m.

In the first interface the video is loaded and two captures are obtained with a increase in time. In the second one, the region to be studied is selected and the correlation algorithm is applied. Finally, in the third interface, the results obtained either as vector field or scalar field (different shades of colors).

Knowing beforehand the theory and examples of how to use this technique, the software allows to get hopeful and rather concise results to study them. Finally, a pilot installation of a PIV with particle-doped fluid has been adapted and used to obtain recordings for apply the developed software abd thus demostrate its functionality under real conditions.

Contenido del proyecto

Índice de figuras

Capítulo 1

Introducción

1.1. Antecedentes y conocimiento de la técnica

El deseo de una comprensión más completa del comportamiento del flujo de un determinado medio ha existido desde hace bastante tiempo. Leonardo da Vinci fue uno de los primeros que se dedicó en parte a la investigación del flujo de líquidos, como se observa en la Figura (1.1). Esta es una de las muchas imágenes que Da Vinci grabó en su cuaderno mientras estudiaba el flujo alrededor de varios tipos de obstrucciones en un río. La creación de modelos para el estudio del flujo de fluidos tiene aplicaciones reales e importantes en campos que abarcan la aerodinámica y dinámica de fluidos.

Figura 1.1: Dibujo de Leonardo Da Vinci de un flujo alrededor de objetos. Fuente [1].

Más recientemente, el estudio del flujo de materiales granulares [1] también ha adquirido gran importancia en la ingeniería. Así como en industrias que involucran fármacos y aplicaciones civiles. El modelado de los flujos granulares también tiene aplicaciones prácticas en fenómenos naturales, como el transporte de dunas de arena y la dispersión de sedimentos [1].

Sin embargo, la capacidad de modelar el flujo de varios medios de una forma simple y compacta para determinar su velocidad, se ha convertido recientemente en una realidad. Las técnicas clásicas como LDA (Anemometría Láser Doppler) utilizan elementos que se introducen dentro del líquido para medir la velocidad en un lugar escogido, durante un espacio de tiempo determinado. La Velocimetría de Imágenes de

[1] Conjunto de partículas sólidas lo suficientemente grandes para que la fuerza de interacción entre ellas sea la de fricción.

Partículas (PIV), técnica estudiada en este proyecto, por otro lado, utiliza el procesamiento de imágenes para determinar el campo de velocidades sin entrar el contacto con el fluido.

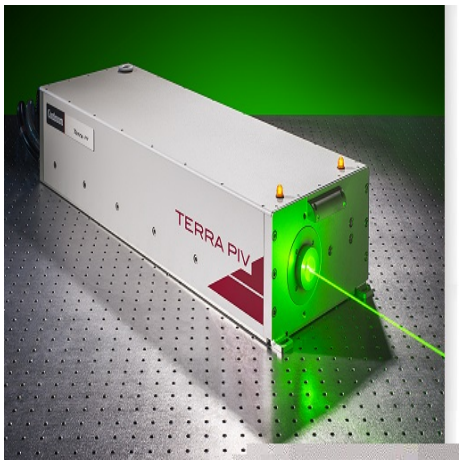
Los primeros casos de PIV, no surgieron hasta finales de los años setenta o principios de los ochenta, cuando se pudo procesar y analizar mediante computadoras las cantidades masivas de datos brutos asociados con esta técnica. La primera publicación que acuñó el término Velocimetría de Imagen de Partículas fue publicada en 1984 [2]. La potencia de cálculo necesaria y la complejidad del aparato experimental de los primeros experimentos de PIV presentaron una barrera para el avance de este tipo de ciencia.

A principios de los 80 el tiempo de preparación típico para un experimento PIV era de dos a tres días, además de los dos o más días que se usaban para la evaluación óptica [3].

Las formas actuales de PIV se caracterizan por la capacidad de tomar con precisión medidas cuantitativas de un gran número de puntos simultáneamente en un fluido. PIV es un técnica relativamente nueva que ha visto recientemente sus aplicaciones en expansión debido a la potencia de computación cada vez mayor que ahora existe en comparación a años anteriores.

Las primeras iteraciones de estos experimentos se realizaron con una cámara capaz de capturar un gran número de fotogramas por segundo y una luz estroboscópica [2] o láser, utilizado para iluminar las partículas sembradas dentro del fluido, véase la Figura (1.2). Estas imágenes podrían entonces ser analizadas para encontrar la velocidad instantánea de las partículas. Recientemente sólo se utiliza láser para la iluminación pues, como se justificará más adelante, también la longitud de onda y las áreas de iluminación pueden ser controladas.

(a) Estroboscopio. Fuente [Autor].



(b) Láser Nd:YAG, muy empleado actualmente en la técnica PIV. Fuente [*directindustry.com*]

Figura 1.2: Distintas fuentes de luz que se han empleado en PIV

Por lo tanto, PIV es un método no intrusivo de visualización de flujo que funciona a través de la iluminación láser de partículas suspendidas en el seno de un fluido transparente. Para que dichas partículas puedan verse y sigan el flujo de fluido de forma que puede utilizarse para estudiar la estructura del flujo (velocidad y dirección) mediante un campo vectorial.

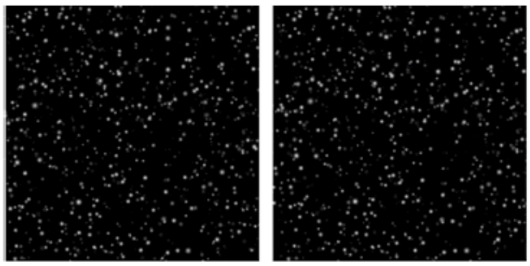
Actualmente, el PIV utiliza cámaras de vídeo digitales que tienen una alta densidad de píxeles mientras que mantienen una alta velocidad de grabación por segundo. El vídeo grabado se divide en imágenes fijas y posteriormente se introduce en un ordenador donde un software analiza imágenes sucesivas, consiguiendo así el campo de vectores que muestra el desplazamiento, ver el esquema de la Figura (1.4). De esta manera es capaz de analizar distintos tipos de flujos y debido a esto hay muchos métodos diferentes para el análisis

[2] Fuente luminosa que emite una serie de destellos muy breves en rápida sucesión y se usa para producir exposiciones múltiples de las fases de un movimiento.

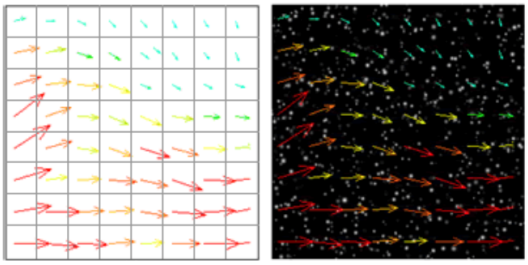
de imágenes. Todos estos métodos, sin embargo, se basan en la identificación de intensidades de píxel específicas dentro de una imagen para que una partícula pueda ser identificada y rastreada.

Cabe destacar que aunque no sea una técnica muy conocida hay bastantes artículos y fuentes que muestran mucho más a fondo la Velocimetría de Imágenes de Partículas como las referencias [2] y [3].

En la Figura (1.3), el panel (a) muestra dos capturas de un fluido dopado con partículas con un incremento de tiempo. El panel (b) se observa el campo de vectores obtenidos al aplicar el software, representando cada vector el desplazamiento de cada conjunto de partículas que se encuentran en cada área de interrogación.



(a)



(b)

Figura 1.3: (a) Muestra dos capturas consecutivas con partículas; (b) Campo vectorial correspondiente a la captura anterior. Fuente [3].

Cuando se consiguen las dos capturas, es probable que sea necesario mejorarlas para aumentar la precisión y facilitar el cálculo de los resultados, ya sea por presencia de ruido, elementos no deseados, etc.

Posteriormente dichas capturas (imágenes) se procesan con un algoritmo de correlación. Para este procesamiento cada imagen seleccionada estará compuesta por pequeñas ventanas, las cuales se denominan "áreas de interrogación". Es importante conocer que se debe asumir que las partículas de cada ventana se muevan de manera homogénea y con cierta coherencia.

Para obtener el campo de velocidades (magnitud y dirección) y estimar sus valores es necesario conocer que lo que se obtiene es el valor medio de cada ventana. Para esto se utiliza el desplazamiento de las partículas y el tiempo que han tardado en realizar dicho desplazamiento mediante dos imágenes consecutivas. Por lo tanto, se emplea el algoritmo para repetir el proceso en todas las áreas de interrogación seleccionadas y de esta manera obtener el campo de vectores de velocidad.

De esta manera, el trabajo de este proyecto se basa en la edición y creación de una interfaz con el programa MATLAB que simule el software de la técnica PIV y ofrezca unos resultados creíbles y coherentes, todo esto desde un punto de vista académico.

Además, se tratará de obtener en el laboratorio de Física aplicada de la ETSID, un vídeo de un fluido dopado con partículas a través de una cámara de alta velocidad, para su posterior análisis.

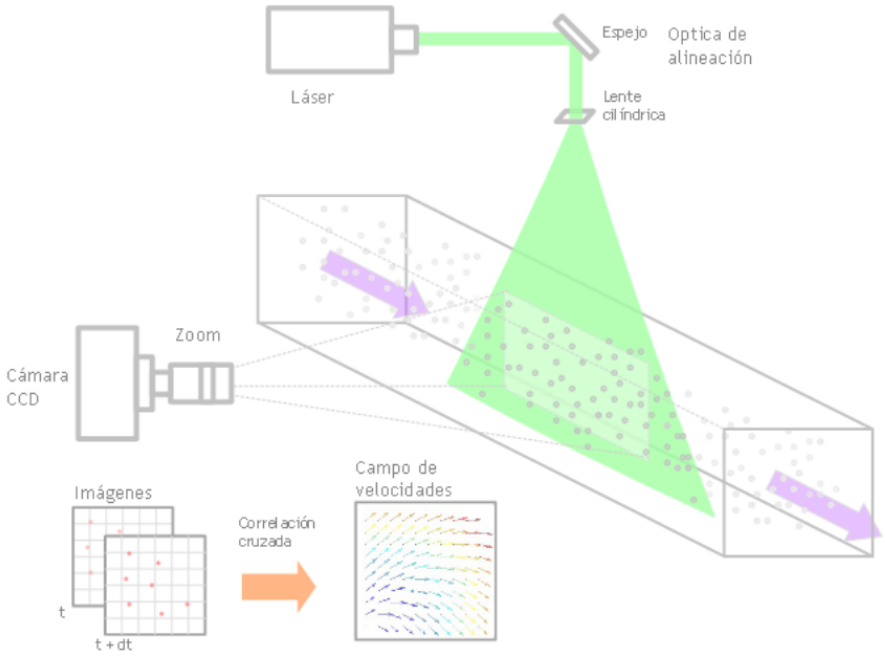


Figura 1.4: Instalación y procesado experimental de PIV. Fuente [*www.laseroptics.com.ar-aplicaciones.html*].

1.2. Aplicaciones

Las investigaciones sobre las mediciones de flujos aerodinámicos y sobre el desarrollo de la técnica PIV tiene aplicaciones en una amplia gama de ingeniería y campos de dinámica de fluidos [4], incluyendo:

- Ingeniería aeroespacial
- Ingeniería y desarrollo de automóviles
- Ingeniería de combustibles
- Sistema de supresión de incendios

En particular, se puede encontrar el uso de PIV en:

- Experimentos de velocidad en túneles de viento para probar la aerodinámica de automóviles, trenes, aviones, edificios además de otros objetos y estructuras.

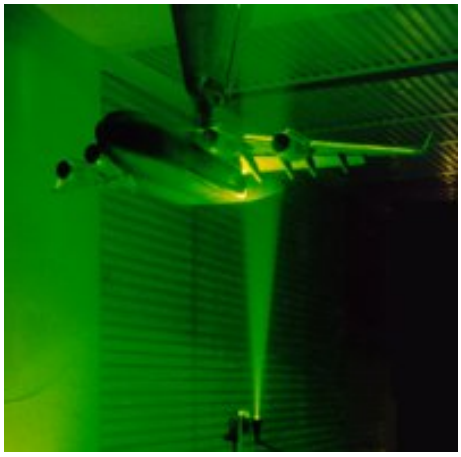


Figura 1.5: Estudio aerodinámico de una maqueta de avión mediante PIV. Fuente [*www.dantecdynamics.com-piv-application-examples*].

- Mediciones de velocidad en flujos de agua (investigación en hidrodinámica general, diseño de casco de barcos, maquinaria rotativa, caudales de tuberías, caudal de canales...).
- Investigación ambiental (investigación de la combustión, dinámica ondulatoria, ingeniería costera, modelización de mareas, hidrología fluvial...).
- Investigación biomédica.
- Investigación sobre la turbulencia.
- Verificación experimental de modelos CFD (Dinámica de fluidos computacional, basados en algoritmos y métodos numéricos).

Figura 1.6: Estudio del flujo para mejorar aerodinámica sobre la bicicleta. Fuente [*www.dantecdynamics.com-piv-application-examples*].

1.3. Objetivos del proyecto

El objetivo principal del proyeto es diseñar una interfaz gráfica mediante MATLAB que simule los cálculos y algoritmos empleados en la técnica "Velocimetría de Imágenes de Partículas"(PIV).

Por otro lado también se busca:

- Lograr la obtención de un vídeo grabado en el laboratorio de Física Aplicada de la ETSID que muestre el comportamiento de un fluido dopado con partículas para su posterior análisis con la interfaz creada.

- Conocer a fondo y de una manera sencilla la técnica PIV mediante dicha interfaz, además de implementar una serie de funciones que traten el vídeo y las imágenes que se quiere analizar.

- Simular el algoritmo que se emplea en esta técnica de manera que sea sencillo pero también eficaz, obteniendo unos resultados interpretables.

1.4. Partes del proyecto por capítulos

- *Capítulo 1:* Introducción. Describe la técnica PIV, así como sus aplicaciones y objetivos planteados en el proyecto.
- *Capítulo 2:* Características y descripción de los elementos que permiten el uso de PIV.
- *Capítulo 3:* Evaluación teórica de las imágenes (aplicación del algoritmo de correlación).
- *Capítulo 4:* Diseño e implementación de la interfaz. Se describen las partes de la interfaz MATLAB, que permite la simulación del procesamiento de datos PIV.
- *Capítulo 5:* Obtención de un vídeo que muestre un fluido dopado con partículas. Se describe la instalación doméstica elaborada en el laboratorio de Físcia Aplicada de la ETSID para la obtención del vídeo.
- *Capítulo 6:* Análisis de los resultados. Se interpretan los resultados obtenidos y se busca una aplicación que devuelva resultados con valores reales.
- *Capítulo 7:* Mejoras y usos más avanzados de la técnica PIV.
- *Capítulo 8:* Conclusiones. Cosas a extraer del proyecto y las futuras aplicaciones y desarrollo de mejora que puede llegar a ofrecer.

Capítulo 2

Descripción de la técnica y sus componentes.

En este capítulo se describen los elementos que componen la técnica PIV y el funcionamiento de los mismos, abarcando la parte material (láser, elementos ópticos, partículas, dispersión de la luz ...). Posteriormente en el Capítulo 3, se estudiará la parte analítica como la función de correlación, y el procesamiento de las imágenes mediante la correlación cruzada.

Como ya se ha comentado, los sistemas PIV miden la velocidad de las partículas determinando su desplazamiento durante un tiempo seleccionado de forma precisa usando una técnica láser de doble pulsación. Una lámina de luz láser ilumina un plano en el flujo, y las posiciones de partículas (naturalmente presentes o añadidas al flujo para tener un número suficiente de dispersores de la luz) en ese plano se registran por medio de una cámara digital CCD o CMOS. Un corto tiempo (micro o milisegundos) más tarde, un segundo impulso ilumina el mismo plano, creando un segundo conjunto de imágenes de partículas para su análisis.

No obstante, para la realización de este proyecto se ha editado este método mediante la grabación de un vídeo y seleccionando dos capturas consecutivas del mismo. Dicho vídeo corresponderá con el fluido dopado con partículas.

A partir de estos conjuntos de imágenes, los algoritmos de análisis PIV obtienen los desplazamientos de partículas correspondiente a cada región de las imágenes, para proporcionar la información de la velocidad de forma precisa.

Este procedimiento exige un sistema bastante elaborado para el procesamiento de los datos y la realización de las operaciones necesarias con ellos. Precisamente por eso se busca mejorar la eficiencia de la Velocimetría de Imágenes de Partículas mediante el uso de software y ordenadores más potentes, pudiendo captar imágenes de mejor resolución que tras ser analizadas devolverán resultados más precisos.

2.1. Partículas trazadoras

PIV como método de medición no mide la tasa de flujo directamente, pero sí la velocidad de las partículas contenidas que son arrastradas en el flujo de fluido. Dichas partículas son reconocidas como "partículas trazadoras". Por lo tanto se está asumiendo que las partículas van a la misma velocidad que el fluido [5].

2.1.1. Propiedades de las partículas

En principio las partículas deben ser pequeñas para seguir mejor el flujo, pero lo suficientemente grandes para detectarlas, y que sigan el mismo flujo de fluido sin retraso o descoordinación. El tamaño de todas las partículas debe ser el mismo debido a que dependiendo de la zona las partículas arrastradas pueden tener diferentes velocidades y eso no es apropiado para el empleo correcto de esta técnica. No obstante,

satisfacer o cumplir estas condiciones no es tan fácil, con lo cual en la práctica únicamente es exigible que sigan bastante bien el movimiento del fluido sin ser intrusivas [6].

Así pues, debido a que las partículas han de estar sumergidas y contenidas en el seno del fluido, tanto el fluido como las partículas tienen que tener la misma densidad o al menos muy semejante. Si esto no se cumple, es deducible que las partículas se hundirán en el fondo del fluido o se quedarán flotando en la superficie, y de esta manera no seguirán adecuadamente el comportamiento del flujo.

Es necesario que las partículas tengan buena dispersión de la luz para que puedan ser registradas de forma correcta por la cámara de alta velocidad. No obstante, una opción más barata y general debido a que la dificultad de obtención y el precio de las partículas es elevado, es utilizar un láser de más calidad y potencia. La dispersión de la luz depende bastante del sitio o el ángulo desde el que se observe, aunque generalmente este ángulo de observación es de 90^{o} con respecto a la luz que incide desde láser. Este ángulo es debido a que es una de las posiciones que tienen más dispersión de la luz.

La cantidad de partículas también es importante, pues demasiadas partículas afectarán mucho al fluido y si hay muy pocas, habrá escasez de información para realizar los cálculos necesarios. Dicho esto es necesario que esta cantidad se reparta de manera uniforme y equilibrada.

Por lo tanto, los principales requisitos son:

- Seguir de forma fiable el flujo de fluido.
- Buena dispersión de la luz.
- Buen precio.
- No afectar a las propiedades del fluido.

Todas estas propiedades van a ser expandidas en este capítulo.

2.1.2. Dispersión (SCATTERING) de la luz

Cuando se utilizan energías de pulso del láser altas, la distribución de esta energía sobre una lámina láser conduce a una densidad de energía que es baja en relación con otros instrumentos que trabajan por medio del láser, sea LDV (Laser Doppler Velocimetry) [1]. Por lo tanto, la eficiencia de dispersión de partículas es muy importante en el uso de esta técnica [6].

Una medida conveniente de la capacidad de dispersión de la luz (espacialmente integrada) es la sección transversal de dispersión C_s, definida como la relación de la potencia dispersa total P_s con la intensidad del láser I_0 incidente sobre la partícula:

$$C_s = \frac{P_s}{I_0} \tag{2.1}$$

La siguiente Tabla compara valores aproximados de la C_s para una molécula diatómica como nitrógeno u oxígeno y dos partículas más grandes.

Diámetro d_e	Sección transversal de dispersión C_s
Molécula	$\approx 10^{-33} m^2$
$1\mu m$	$C_s \approx (\frac{d_e}{\lambda})^4 \approx 10^{-12} m^2$
$10\mu m$	$C_s \approx (\frac{d_e}{\lambda})^2 \approx 10^{-9} m^2$

[1] La luz reflejada de las partículas sembradas en el flujo es recogida y procesada. Un único haz del láser es dividido en dos haces con la misma intensidad que se enfocan en un mismo punto formando un modelo de interferencia, definiendo el volumen de medición.

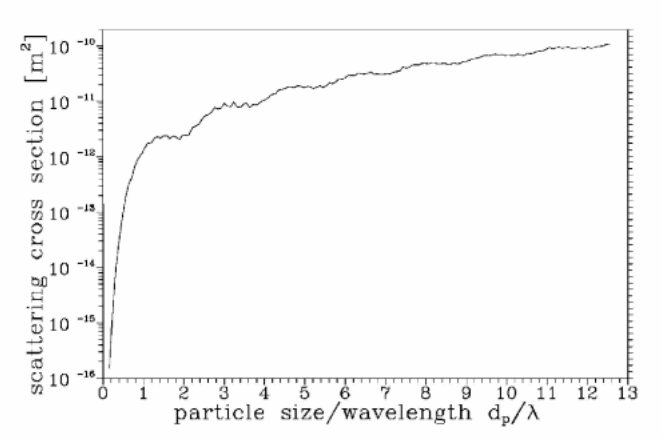


Figura 2.1: Variación de C_s en función de la relación entre el diámetro de partícula d_p y la longitud de onda λ del láser para partículas esféricas con un índice de refracción $m = 1{,}6$. Fuente [6].

Estos ejemplos indican claramente la enorme diferencia entre las secciones transversales de dispersión de la luz de las moléculas y las secciones transversales de las partículas que son utilizadas para los experimentos PIV [5].

Existen dos tipos de dispersión de la luz en función del tamaño de las partículas y la longitud de onda de la radiación incidente:

- La **dispersión de Rayleigh** se refiere principalmente a la dispersión elástica de la luz procedente de partículas atómicas y moleculares cuyo diámetro es inferior a aproximadamente una décima parte de la longitud de onda de la luz incidente.

- La **dispersión de Mie** se refiere principalmente a la dispersión elástica de la luz procedente de partículas atómicas y moleculares cuyo diámetro es mayor que aproximadamente la longitud de onda de la luz incidente.

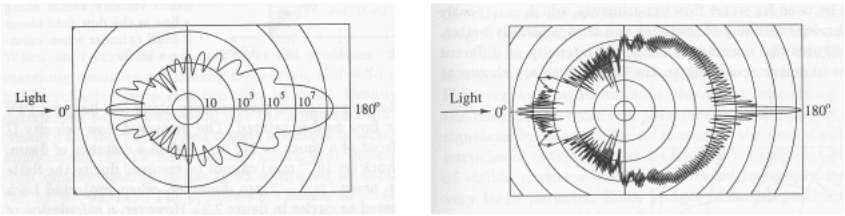
La dispersión de Rayleigh de moléculas es demasiado débil en las condiciones que se aplica PIV, incluso con la iluminación de un láser de potencia o energía de pulso máxima. Por tanto, en el uso de la técnica PIV se emplea una menor intensidad de luz incidente en comparación a *LDA (Anemometría Láser Doppler)*.

En cuanto a la dispersión Mie, se sabe que en experimentos PIV como ya se ha mencionado se observa normalmente la luz dispersada a 90^o de la hoja de luz incidente. Estudios de dispersión a partir de partículas esféricas han demostrado que la relación de intensidad luz dispersada a 90^o con luz difundida hacia delante Is_{90}/Is_0 es una función dependiente tanto del parámetro de relación de tamaño d_p/λ como del índice de refracción m, véase Figura (2.2). Además la distribución angular de la luz dispersa en los alrededores de 90^o es compleja, con varios nodos cuyas posiciones exactas son fuertemente dependientes del tamaño de la partícula.

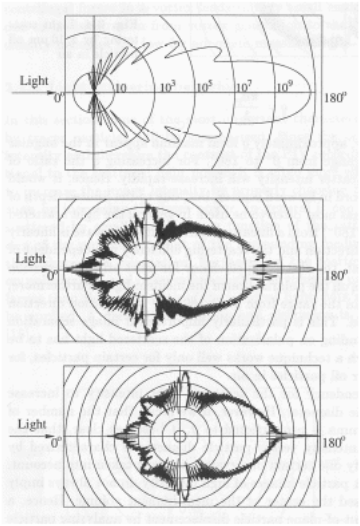
Es por esto, que PIV utiliza la teoría de dispersión de Mie, ya que la relación de tamaño de partícula d_p/λ, es bastante próxima a 1. Aunque se haya demostrado también que una gran parte de la luz que se dispersa lo hace hacia delante, contrariamente a la configuración que se ha impuesto en el sistema PIV (de 90^o), con lo cual se reduce mucho la intensidad de dispersión que se capta.

De esta manera hay que tener en cuenta que:

- La relación Is_{90}/Is_0 disminuye con el incremento del parámetro de tamaño d_p/λ, estando sus valores aproximadamente en el intervalo $[10^{-1}\ ,\ 10^{-3}]$ para una dispersión de partículas que sea útil en



(a) Dispersión de la luz por una partícula de aceite en el aire cuando el índice de refracción $m \sim 1{,}4$. Izquierda: diámetro de $1mm$. Derecha: diámetro de $10mm$.



(b) Dispersión de la luz por una partícula de vidrio de $1mm$, $10mm$ y $30mm$ en agua. Índice de refracción $m = 1{,}52$.

Figura 2.2: Ejemplos de dispersión Mie para distintos tipos de partículas. Fuente [6].

PIV.

- La intensidad resultante de la luz dispersada para una intensidad del plano de luz dada dependerá de la influencia de C_s y Is_{90}/Is_0, que muestran tendencias opuestas con el aumento del tamaño de partícula. En general, las partículas más grandes seguirán dando señales más fuertes.

- La relación Is_{90}/Is_0 crece con el aumento del índice de refracción m. Por lo tanto, las partículas en el aire dan una dispersión en 90^o más fuerte que en agua.

2.1.3. Propiedades fluido-dinámicas

La capacidad de seguir el fluido por parte de las partículas depende de:

- La forma de la partícula, que es asumida esférica y presenta un diámetro aerodinámicamente equivalente designado como d_p.
- La densidad de las partículas ρ_p.
- La velocidad de la partícula U_p, que definirá la aceleración $\frac{dU_p}{dt}$.
- La densidad del fluido ρ_f y la viscosidad dinámica μ del fluido o la viscosidad cinemática $\nu = \mu/\rho_f$.

Definiendo así la ley de Newton que regula el movimiento fluidodinámico de la partícula:

$$\rho_p \frac{\pi d_p^3}{6} \frac{dU_p}{dt} = \sum_i F_i \tag{2.2}$$

Donde $\sum_i F_i$ incluye todas las fuerzas experimentadas por la partícula:

$$\rho_p \frac{\pi d_p^3}{6} \frac{dU_p}{dt} = \ (1)\ -3\pi\mu d_p V + \ (2)\ \rho_f \frac{\pi d_p^3}{6} \frac{dU_f}{dt}\ (3)\ -\frac{1}{2}\rho_f \frac{\pi d_p^3}{6} \frac{dV}{dt}\ (4)\ -\frac{3}{2} d_p^2 (\pi\mu\rho_f)^{\frac{1}{2}}\ (5)\ + \int_0^t \frac{dV}{d\xi} \frac{d\xi}{\sqrt{t-\xi}} \tag{2.3}$$

Siendo cada término lo siguiente:

(1) Arrastre viscoso según la ley de Stokes.

(2) Fuerza de aceleración del fluido.

(3) Fuerza debido a un gradiente de presión en la proximidad de la partícula.

(4) Resistencia de un fluido no viscoso a la aceleración de la esfera ("masa añadida").

(5) Integral de Historia de Basset - resistencia causada por la inestabilidad del campo de flujo.

Ley de arrastre de Stokes

Se considera que la ley de arrastre de Stokes se aplica cuando el número de Reynolds de la partícula Re_p es menor que la unidad, donde Re_p se define como:

$$Re_p = \frac{\rho_f V d_p}{\mu} = \frac{V d_p}{\nu} \tag{2.4}$$

En un experimento típico de PIV con partículas de $10\mu m$ y velocidad media de $20cm/s$, dependiendo del fluido en el que se trabaje el Re_p será lo siguiente:

$$Re_p = 10 \times 10^-6 \times 0{,}2/1{,}46 \times 10^-5 = 0{,}13 \quad (aire)$$

$$Re_p = 10 \times 10^-6 \times 0{,}2/1{,}0 \times 10^-6 = 2 \quad (agua)$$

Donde aparentemente, en el caso del agua se desvía bastante del arrastre de Stokes lineal, sin embargo, la mayoría de las simulaciones asumen este valor. Por tanto, la teoría de arrastre de Stokes proporciona una estimación conservadora de la capacidad de seguimiento de la partícula, ya que el arrastre real tiende a ser mayor.

Parámetros de las partículas

- **Tiempo de respuesta de las partículas** t_p

En la mayoría de los experimentos realizados con PIV, muchos términos que aparecen en la ecuación de arrastre de Stokes puede ser despreciados ya que normalmente el número de Reynolds de la partícula que aparece en estos casos es del orden de la unidad. Si no hay fuerza externa ejercida sobre la partícula, se puede expresar el desfase de velocidad (diferencia entre las velocidades de partícula y de fluido donde se encuentra inmersa) de una partícula en un fluido que se acelera continuamente como:

$$V = U_p - U_f = d_p^2 \frac{(\rho_p - \rho_f)}{18\mu} \frac{dU_f}{dt} \tag{2.5}$$

La velocidad de la partícula correspondiente a la velocidad del fluido en el caso de que las partículas pesadas ($\rho_p >> \rho_f$) estén en un flujo de aceleración continua es:

$$U_p(t) = U_f[1 - exp(-\frac{t}{\tau_p})] \tag{2.6}$$

De esta manera el tiempo de respuesta de las partículas será:

$$\tau_p = d_p^2 \frac{\rho_p}{18\mu} \tag{2.7}$$

τ_p es una medida conveniente de la tendencia que tienen las partículas para seguir el flujo, incluso si la aceleración del fluido no es constante o si la ley de arrastre de Stokes no se aplica. Además es el tiempo de respuesta de las partículas, que corresponde con una medida de la inercia de la partícula.

- **Número de Stokes St**

El tiempo de respuesta de las partículas τ_p se debe evaluar frente a la escala de tiempo de turbulencia, es decir, el tiempo de revolución de la estructura turbulenta más relevante. En un flujo turbulento completamente desarrollado, se usa a menudo la escala de tiempo de Kolmogorov, denominada τ_k, que es el tiempo de revolución del remolino más pequeño.

Para este propósito se define número de Stokes St como la relación del tiempo de respuesta de la partícula con la escala de tiempo de Kolmogorov:

$$St = \frac{\tau_p}{\tau_k} \tag{2.8}$$

En función del valor de St el grado de acoplamiento entre la fase de la partícula y el fluido puede variar de la siguiente manera:

1 Si $St \to 0$, las partículas se comportan como trazadores.

2 Si $St \to \infty$, las partículas son completamente insensibles al flujo de fluido.

- **Frecuencia característica C**

En el caso de flujo de gas donde $\rho_p >> \rho_f$, la respuesta de la partícula con la frecuencia de la turbulencia debe ser considerada, definiendo así C:

$$C = \frac{18\mu}{\rho_p d_p^2} \tag{2.9}$$

Si se define $\omega_c = 2\pi f$ como la frecuencia de turbulencia más alta de interés, C es una frecuencia característica del movimiento de la partícula, se puede usar la siguiente ecuación para determinar la capacidad de rastreo de la partícula:

$$\frac{\overline{u_p}^2}{\overline{u_f}^2} = \frac{1}{(1+\omega_c/C)} \tag{2.10}$$

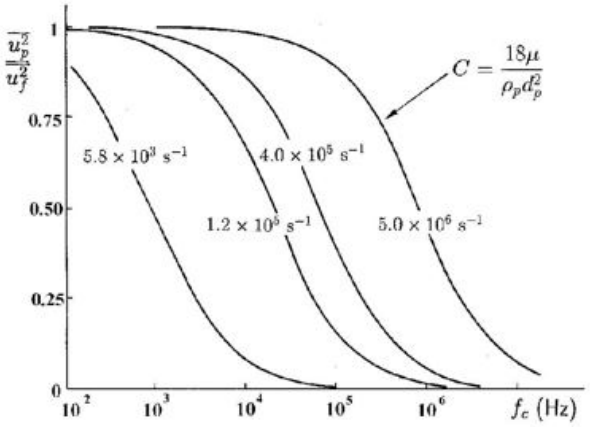


Figura 2.3: La respuesta de las partículas en flujo con turbulencia. Fuente [21].

La Figura (2.3) muestra la relación entre las intensidades de fluctuación de las partículas y el movimiento del fluido en un flujo turbulento sumado sobre todas las frecuencias de turbulencia por debajo de f_c.

La curva para $C = 1{,}2 \times 10^5\ s^{-1}$ corresponde aproximadamente al movimiento de una gota de agua de $1\mu m$ en el aire. Los valores de C más altos conducen a un mejor comportamiento de seguimiento de flujo por parte de la partícula.

Consideraciones adicionales

- **Tamaño de partícula vs. escala de turbulencia**

Las partículas sembradas deben ser más pequeñas que la escala de turbulencia más pequeña si se quiere identificar todas las estructuras en las proximidades del flujo. La escala de longitud de fluido más pequeña se llama escala de longitud de Kolmogorov, y está relacionada con el tamaño del remolino más pequeño.

- **Uniformidad de siembra de partículas**

Para esto, será necesario saber el comportamiento a diferentes números de Stokes.

$$St << 1; St \sim 1; St >> 1$$

Además, para asegurar un detalle espacial lo suficientemente importante en el campo de flujo, se necesita generalmente una mayor concentración de partículas con PIV que con LDV, con lo que es posible [2] esperar la llegada indefinida de la dispersión de una partícula en el volumen estudiado.

Un tamaño de partícula uniforme es deseable para evitar la excesiva intensidad de las partículas más grandes y el ruido de fondo, disminuyendo la precisión, a partir de partículas pequeñas. Un sembrado uniforme es también importante para el cálculo de la correlación, donde las partículas grandes producirían un sesgo del resultado de la correlación hacia los trazadores grandes, comprometiendo así la exactitud de la

[2] Dentro de las limitaciones impuestas por la inestabilidad del flujo, muestreo sesgado de las estadísticas de velocidad o costes de funcionamiento.

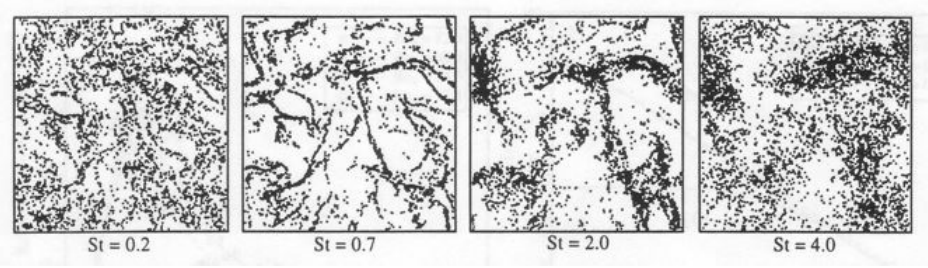


Figura 2.4: Posiciones de partículas instantáneas en un plano 2D después de la rotación de remolino para partículas con distintos números de Stokes. Fuente [6].

medida.

Las partículas que existen naturalmente en el flujo (como la impureza en el agua) rara vez cumplen los requisitos anteriores. Por lo tanto, en las aplicaciones PIV, a menudo es necesario sembrar el flujo con partículas trazadoras elegidas. De esta manera las partículas se premezclan agitándose con el fluido completo o liberándolas in situ mediante una fuente de sembrado.

Desde el punto de vista de la uniformidad de datos, elegir partículas trazadores con un pequeño número de Stokes, ver Figura (2.4).

2.1.4. Tipos de partículas

Para la mayoría de los experimentos es deseable que las partículas sembradas sean no tóxicas, no corrosivas, no abrasivas, no volátiles y químicamente inertes. Teniendo en cuenta estos requisitos, una gran variedad de partículas trazadoras están disponibles para los experimentos PIV.

Tipo	**Material**	**Diámetro promedio en micras**
Sólido	Poliestireno	0.5 – 10
	Aluminio	2 – 7
	Magnesio	2 – 5
	Micro-globos de vidrio	30 – 100
	Gránulos con cubiertas sintéticas	10 – 50
	Ftalato de dioctilo	1 – 10
Líquido	Diferentes aceites	0.5 – 10
Humo		< 1

Figura 2.5: Partículas empleadas en gases. Fuente [Autor].

Las Figuras (2.6) y (2.5) constituyen una guía general para seleccionar partículas apropiadas para PIV en función del medio del fluido, ya sea líquido o gas.

Algunos fabricantes aconsejan que:

- Las partículas deben tener un tamaño de entre 3 y 5 píxeles de la imagen, es decir, cuando se tome la foto, la partícula ocupará entre 3 y 5 píxeles.

Tipo	Material	Diámetro promedio en micras
Sólido	Poliestireno	5 – 100
	Aluminio	2 – 7
	Esferas de vidrio	10 – 100
	Gránulos con cubiertas sintéticas	10 – 500
Líquido	Diferentes aceites	50 – 500
Gaseoso	Burbujas de Oxígeno	50 – 1000

Figura 2.6: Partículas empleadas en líquidos. Fuente [Autor].

- Por área de interrogación debe haber entre 5 y 15 partículas.
- Los tamaños generales utilizados para PIV están comprendidos entre 0.5 y 10 μm.
- El desplazamiento máximo del patrón buscado en el segundo fotograma ha de ser como máximo del 25 % del tamaño de la región de interrogación.

2.2. Fuente de luz

Un factor crucial en la técnica PIV es la generación de un haz de luz plano con unas características significativas de espesor, color, superficie e intensidad. La única fuente que cumple estas características es el láser, que mediante el uso de elementos ópticos (lentes) son capaces de formar un plano de luz. Dicho plano obtenido se introduce en el fluido para iluminar la región deseada, es aconsejable que el plano láser vaya en la dirección del flujo laminar.

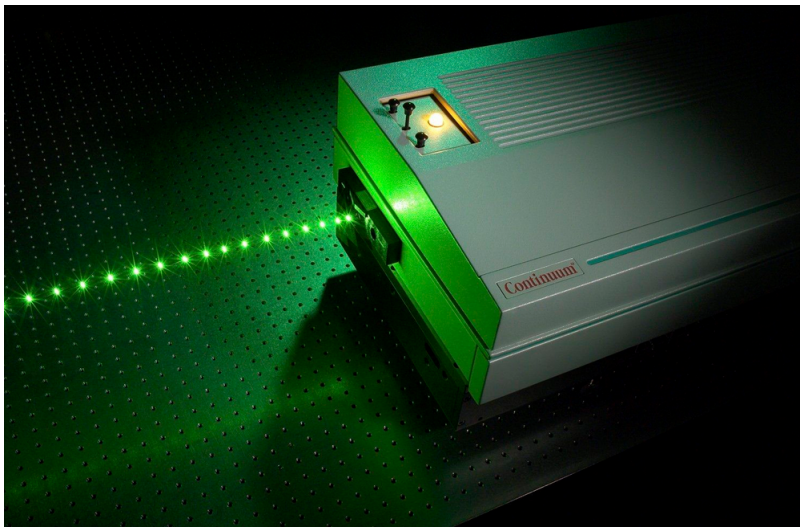


Figura 2.7: Láser pulsado Nd:YAG. Fuente [directindustry.com].

Por otro lado, en los laboratorios especializados en esta técnica se emplean láseres de doble pulso, otra razón por la cual se utiliza el láser como fuente de luz. Estos láseres generan los pulsos a distintas frecuencias y periodos de duración.

Estos láseres que se emiten mediante pulsos que se sincronizan mediante un dispositivo hardware con la cámara digital. Así el obturador de la cámara permanecerá abierto el tiempo que se ilumine la región seleccionada en el fluido, con lo cual se capturará la posición de las partículas en el instante deseado y de esta manera se controla el movimiento de las partículas estudiadas evitando el desenfoque.

En resumen, normalmente su utiliza un láser pulsado, con un tiempo de exposición lo suficientemente corto que 'congele' el flujo, y una cámara CCD (Charge Coupled Device) con alta resolución espacial y sincronizable con el láser. La cámara tiene que ser de muy alta velocidad para permitir tomar fotos muy próximas y captar los dos pulsos del láser (uno por cada foto).

Hay varios tipos de láser que sirven para realizar experimentos con la técnica PIV. La selección del láser a utilizar depende de varios parámetros, como la potencia de luz, el tamaño de muestra que se quiere estudiar en el fluido o la velocidad del fluido. Teniendo en cuenta estas variables, el láser más empleado por la técnica PIV es el Nd:YAG (Neodimio YAG), que es un láser de estado sólido y a su vez tiene la capacidad de seguir movimientos de fluido rápidos, pues produce una gran intesidad de luz en pulsos cortos y a una frecuencia lo suficientemente elevada.

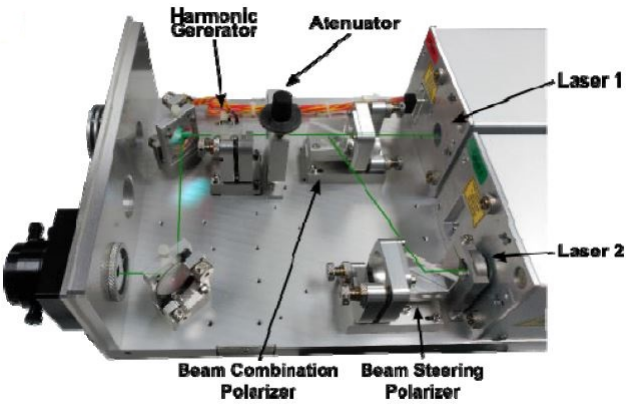


Figura 2.8: Combinación de dos láseres para formar un Nd:YAG. Fuente [Autor].

Por tanto, el Nd:YAG es un láser con una longitud de onda de 532 nm y pulsante, posee una gran intensidad de pulso y la capacidad de emitir pulsos cortos, más cortos conforme más rápido sea el movimiento del fluido, existen prácticas donde se emplean dos láseres diferentes que van a ser unidos y sincronizados, ver Figura (2.8), se lanzan los dos láseres y uno de ellos se desvía mediante un espejo, haciendo que se junte con el otro. Así se consiguen dos haces con una separación espacial pequeña, cumpliendo las capacidades de este tipo de láseres ya mencionadas.

Añadiendo características de este láser bastante utilizado mucho en la práctica PIV, tiene:

- Frecuencias de repetición bajas (15 Hz), se puede hacer funcionar por debajo de 15 Hz, pero por encima de este valor no se puede, por tanto esto es una limitación, ya que no siempre es regulable.
- La energía del haz es de 135 mj por pulso, esto es más que suficiente para el tamaño de campos de fluido que se suelen medir con esta técnica.
- El diámetro del haz depende de la barra de neodimio, en el caso del YAG es de 5 mm.
- Estas son las características de la fuente de luz más común en esta técnica, en cuanto al sistema PIV creado en el laboratorio de física de la ETSID, utiliza un láser monopulso (LED continuo), que mediante la ayuda de unas lentes (sistema óptico) formarán un plano láser que se sumergirá en el fludio analizado.

2.3. Captura de las imágenes

La técnica PIV necesita un sistema de captura y almacenamiento de las imágenes que se quieren del fluido dopado con partículas. A pesar de que la tecnología CMOS ha evolucionado mucho en la última década, el medio más común para la captación de estas imágenes son las cámaras CCD (Charge-Coupled Device).

Las cámaras digitales han aumentado su presencia y uso a medida que los precios han bajado. Uno de los precursores que ha provocado la caída de los precios ha sido la introducción de sensores de imagen CMOS. Los sensores CMOS son mucho menos costosos de fabricar que los sensores CCD.

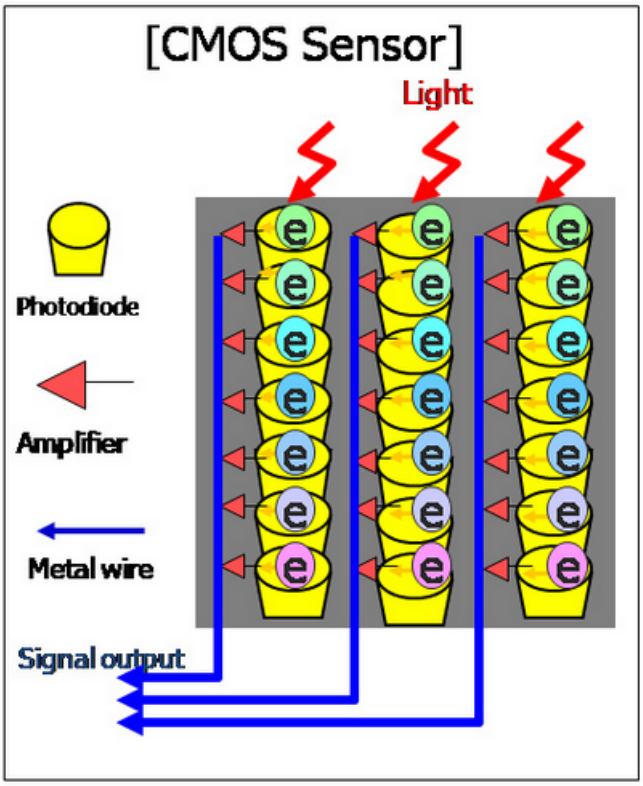


Figura 2.9: Funcionamiento del registro en sensores CMOS. Fuente [*www.parentesis.com-tutoriales-SensorCCDoCMOS*].

Ambos sensores de imagen CCD (dispositivo de carga acoplada) y CMOS (semiconductor de óxido de metal complementario) se emplean de la misma manera, pues transforman la luz en electrones, ofreciendo un comportamiento similar a las células solares.

Una forma simplificada de entender como funciona el sensor utilizado en una cámara digital (o videocámara) es pensar que tiene una matriz bidimensional de miles o millones de pequeñas células solares, cada una de las cuales transforma la luz de una pequeña parte de la imagen en electrones. Ambos dispositivos CCD y CMOS realizan esta tarea utilizando una variedad de tecnologías [9].

El siguiente paso es procesar el valor (carga acumulada) de cada celda de la imagen. En un dispositivo CCD, la carga se transporta realmente a través del chip y se lee en una esquina de la matriz. Un convertidor analógico-digital convierte el valor de cada píxel en un valor digital. En la mayoría de los dispositivos CMOS, hay varios transistores en cada píxel que amplifican y mueven la carga usando cables más tradicionales. El enfoque CMOS es más flexible porque cada píxel se puede leer individualmente, ver Figuras (2.11) y (2.9).

Los CCD utilizan un proceso de fabricación especial para crear la capacidad de transportar la carga a través del chip sin distorsión. Este proceso conduce a sensores de muy alta calidad en términos de fidelidad y sensibilidad a la luz. Los chips CMOS, por otro lado, usan procesos de fabricación tradicionales para crear el chip, los mismos procesos usados para fabricar la mayoría de los microprocesadores.

Debido a las diferencias de fabricación, existen varias diferencias notables entre los sensores CCD y CMOS:

- Los sensores CCD, como se mencionó anteriormente, crean imágenes de alta calidad y bajo ruido. Sensores CMOS, tradicionalmente, son más susceptibles al ruido.
- Debido a que cada píxel en un sensor CMOS tiene varios transistores situados a su lado, la sensibilidad a la luz de un chip CMOS tiende a ser menor.
- CMOS tradicionalmente consume poca potencia. La implementación de un sensor en CMOS produce un sensor de baja potencia.
- Los CCD utilizan un proceso que consume mucha energía. Los CCD consumen hasta 100 veces más energía que un sensor CMOS equivalente.
- Los chips CMOS pueden fabricarse prácticamente en cualquier línea de producción con silicio estándar, por lo que tienden a ser extremadamente baratos en comparación con los sensores CCD.
- Los sensores CCD han sido producidos en masa por un período de tiempo más largo, por lo que están más desarrolados y tienden a tener mayor calidad y más píxeles.

Basándose en estas diferencias,se puede ver que los CCD tienden a utilizarse en cámaras que se centran en imágenes de alta calidad con gran cantidad de píxeles y excelente sensibilidad a la luz, justificando su uso en PIV.

Se pueden lograr cortos tiempos de exposición intermedia con una cámara de alta velocidad que grabe continuamente imágenes a una velocidad de varios kHz o utilizando una cámara con una arquitectura de exploración progresiva. Con una cámara de alta velocidad el tiempo entre las imágenes es lo más corto posible, alrededor de $10\mu s$ y esto se consigue al dejar que el primer impulso del láser llegue hasta el final de la exposición de la primera imagen y el segundo impulso llegue hasta el inicio de la exposición de la segunda imagen, ver Figura (2.10).

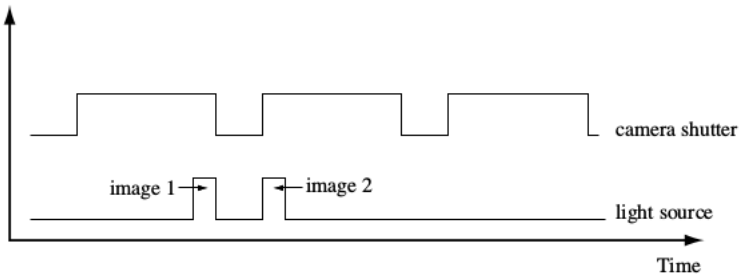


Figura 2.10: Esquema del procedimiento seguido para acortar el tiempo entre imágenes cuando se utilizan cámaras de alta velocidad continuas. Fuente [9].

Los sensores CMOS tradicionalmente tienen menor calidad, menor resolución y menor sensibilidad, aunque ahora están mejorando hasta el punto de que alcanzan una paridad cercana con los dispositivos CCD en algunas aplicaciones siendo menos caras y teniendo una gran duración de la batería.

Por lo tanto, mediante una cámara CCD se puede tomar una foto y cuando se decida, lo que se hace es llevar la información al buffer (almacenamiento), dejando libres los sensores para tomar una segunda captura. La información en el buffer tienen que ser leída para vaciarla y una vez hecho esto se pasa a la

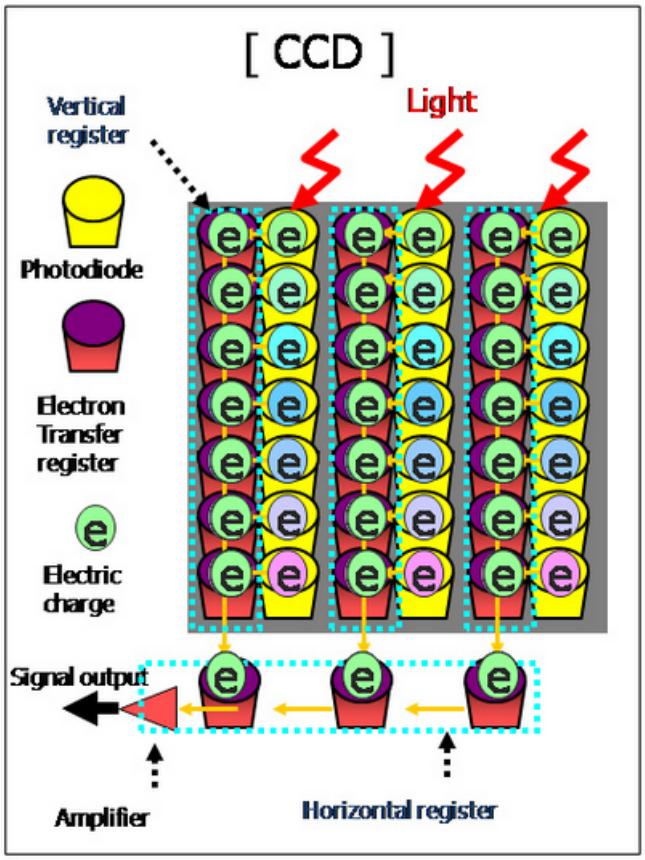


Figura 2.11: Funcionamiento del registro en cámaras CCD. Fuente [*www.parentesis.com-tutoriales-SensorCCDoCMOS*].

RAM.

Esa lectura se hace por columnas, por tanto, el tiempo para vaciar el buffer es bastante grande, haciendo que las cámaras sean lentas, ya que necesitan tiempo para lectura de información.

De esta manera se puede controlar el tiempo de exposición, pero cuando se quiera dejar de capturar información, necesariamente hay que volcar la información a la RAM, con lo cual es imprescindible que el buffer esté vacío. Si no hay espacio para volcar la información (buffer no vacío), no se puede dejar de exponer, por tanto, el tiempo mínimo de exposición (unos 80 ms) vendrá impuesto por cómo de rápido se pueda escribir.

Es importante conocer las distintas formas de registrar las imágenes de la técnica PIV, ya sea la obtención de una imagen con múltiples exposiciones o varias imágenes con una exposición.

En la Figura (2.12) se muestra la imagen de una partícula con distintas maneras de exposición. En la primera se observa un ejemplo en el que el tiempo de exposición del obturador es elevado, por tanto el movimiento de la partícula se registra en una línea. En la segunda se observa un ejemplo en el que el tiempo de exposición del obturador es más corto y repetido varias veces, por tanto se registra el movimiento de la partícula en cada una de esas exposiciones, es decir, la posición puntual en cada una [10].

Sin embargo, en este proyecto únicamente se considera la segunda opción, pues los cálculos se realizan mediante múltiples (dos) imágenes, y para cada una de ellas se genera una simple exposición, por lo tanto en la Figura (2.12) se muestra la posición de una partícula en cada una de las exposiciones, sin haber en este caso imágenes de partículas superpuestas.

(a) Una exposición.

(b) Doble exposición.

Figura 2.12: Distintas maneras de exposición de imágenes. Fuente [4].

2.4. Óptica

Un tema importante para la correcta puesta en funcionamiento de la técnica PIV, es la obtención del plano láser. Éste se obtiene mediante óptica configurada por lentes, ver Figura (2.13) pues en función de como se estructure se podrá obtener la configuración deseada.

Figura 2.13: Lentes cilíndricas, muy empleadas en PIV. Fuente [*www.directindustry.es*].

El procedimiento y la finalidad que se pretende, es dirigir el láser en la posición correcta y además conseguir la formación de un plano (2D) en dicha dirección, de esta manera el haz de láser estará en la dirección correcta. Es importante tener cuidado con la configuración del sistema, pues puede provocar la disminución de la intensidad del haz.

A continuación, se describen tres configuraciones de lentes diferentes, que se han utilizado en varios experimentos de esta técnica y que por tanto son las más comunes, cabe destacar que las ecuaciones para el cálculo de la distribución de intensidad del haz no resultan importantes. La razón de esto es que las ecuaciones de la óptica geométrica ya son suficientes para un diseño general si la configuración de la lente elegida no requiere un descripción especial [7]. Por otro lado, cálculos más sofisticados basados en la óptica gaussiana suelen requerir hipótesis ya dadas, que son válidas sólo para casos excepcionales. Los programas informáticos pueden utilizarse para predecir otros parámetros tales como el grosor del plano de luz en el centro del haz donde el espesor teórico (geométrico) es cero, pero su descripción es mucho más precisa de lo que se requiere para la realización del PIV.

El elemento esencial para la generación de plano de luz es una lente cilíndrica. Cuando se utilizan láseres con un diámetro de haz suficientemente pequeño y divergente, como láser de iones de argón, una lente cilíndrica puede ser suficiente para generar una lámina de luz de forma apropiada. Para otras fuentes de luz; como láseres Nd: YAG, generalmente se requiere una combinación de diferentes lentes para generar planos de luz de alta intensidad. Al menos una lente adicional es necesaria para enfocar la luz y que tenga un espesor apropiado.

Tal configuración se muestra en la Figura (2.14), donde también se ha añadido una tercera lente cilíndrica con el fin de generar una lámina de luz de constante altura.

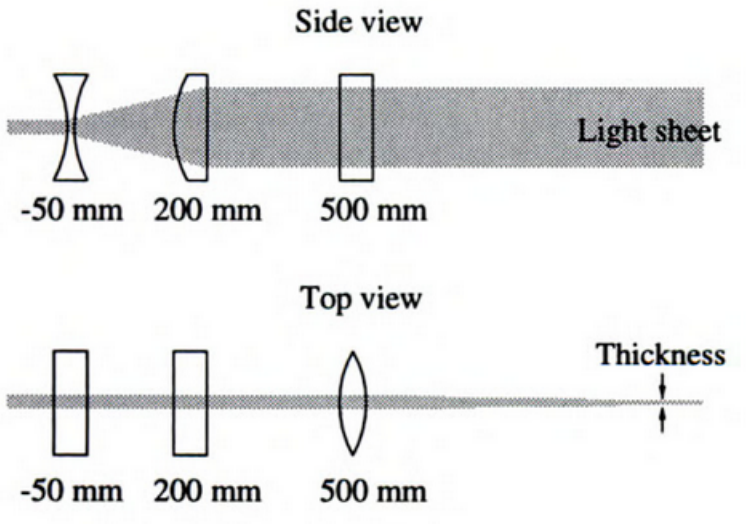


Figura 2.14: Configuración óptica con tres lentes cilíndricas (una de ellas con distancia focal negativa divergente). Fuente [3].

La razón por la que una lente divergente se ha utilizado en primer lugar es que las líneas focales deben evitarse. En los láseres de pulso de alta potencia hay que evitar puntos focales, de lo contrario el aire próximo al punto focal será ionizado. Las líneas focales generalmente no ionizan el aire, pero las partículas de polvo pueden quemarse si el área cercana a la línea no está cubierta o evacuada. En estos casos ocurrirá una radiación acústica y las propiedades del haz cambiarán significativamente.

En la lámina de luz que se muestra en la Figura (2.14), la posición de espesor mínimo viene dada por la divergencia del haz de la fuente de luz y la longitud focal de la lente cilíndrica del lado derecho a una distancia de 500 mm de la última lente.

La combinación de una lente cilíndrica junto con dos lentes de telescopio hace que el sistema sea más versátil. Esto se muestra en la Figura (2.15) donde se han utilizado lentes esféricas. La altura del plano de luz que se muestra viene dada principalmente por la distancia focal de la lente cilíndrica en el centro. Sin embargo, también podría utilizarse una lente focal negativa divergente, ya que la línea focal tiene una extensión relativamente grande, por tanto, esta configuración se puede utilizar también para láseres pulsados. La adaptación de la altura del plano de luz debe hacerse cambiando la altura de la lente cilíndrica. El ajuste del espesor puede realizarse fácilmente moviendo las lentes esféricas una con respecto a la otra.

El uso de lentes esféricas en general no permite que la altura y el espesor del plano de luz se cambien independientemente. Esto puede hacerse mediante la configuración mostrada en la Figura (2.16). Además, esta configuración permite la generación de planos de luz que son más delgados que el diámetro del haz en cada lugar. Por lo tanto, permite la generación de láminas de luz que ya son delgadas poco después de la última lente. Con esta disposición el espesor puede ser pequeño constantemente. Sin embargo, la energía por área de estas configuraciones es alta cuando se usan láseres de pulso, la región crítica debe cubrir la línea focal para evitar polvo o aparición de partículas que perturban la generación de un plano de luz definido. Usando una lente cilíndrica divergente se resolverían esos problemas, pero la combinación mostrada en la Figura (2.16) tiene la ventaja de obtener imágenes del haz desde una cierta posición adelantada de la lente con respecto al área de estudio mientras que mantiene sus propiedades constantes.

Pueden usarse consideraciones geométricas simples de estas combinaciones de lentes para determinar desde qué posición se ha representado el haz de láser, esta información se puede utilizar para optimizar la distribución de intensidad del plano de luz. Para láseres con un perfil del haz crítico, esto puede mejorar el

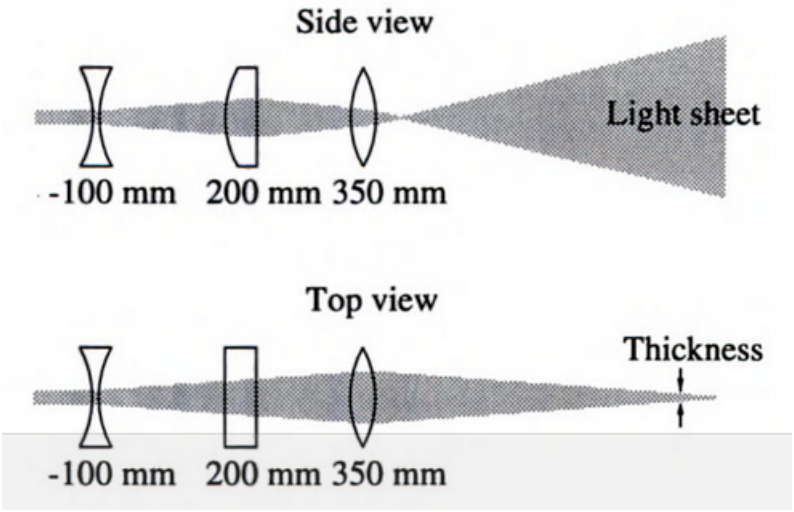


Figura 2.15: Óptica clara utilizando dos lentes esféricas (una de ellas con distancia focal negativa) y una lente cilíndrica. Fuente [3].

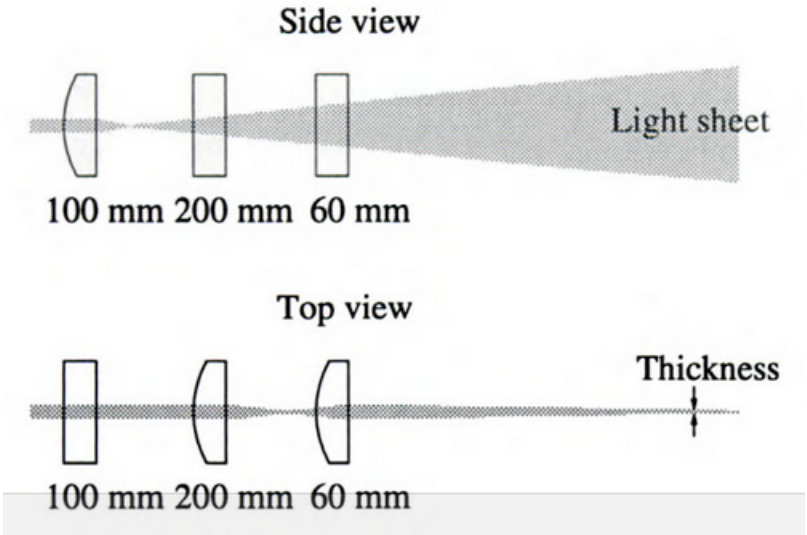


Figura 2.16: Óptica utilizando tres lentes cilíndricas. Fuente [3].

rendimiento de obtención de datos válidos, porque la distribución de la intensidad del plano de luz, especialmente en la dirección fuera del plano, es esencial para la calidad de la medida [8].

2.5. Sistema de registro

PIV se basa en el estudio de imágenes en secuencia para determinar el movimiento de las partículas. Por tanto, se utiliza un método denominado deformación de ventana para rastrear partículas entre la primera imagen y la segunda, con el fin de generar un marco. Para iniciar el proceso de análisis, primero divide la imagen en las llamadas áreas de interrogación. Éstas son subsecciones de las imágenes y son de un tamaño específico medido en píxeles. El tamaño de la ventana suele ser seleccionado por el usuario en función tipo de material que se está rastreando. Para que PIV pueda rastrear las partículas de un área de interrogación a la siguiente es necesario que el programa sepa la intensidad de varios píxeles. PIV controla y sigue la intensidad de los píxeles. Entonces, se utiliza un tipo de función de correlación cruzada [11].

$$Max = \int_A f_1(x) f_2(x+a) dx^2 \tag{2.11}$$

Esto permite que el programa PIV comparar intensidades de las partículas imagen a imagen de modo que pueda generar el campo de la velocidades. En la correlación cruzada la función "a" es el vector de desplazamiento o el desplazamiento entre el área de interrogación y "A" que es el dominio de correlación.

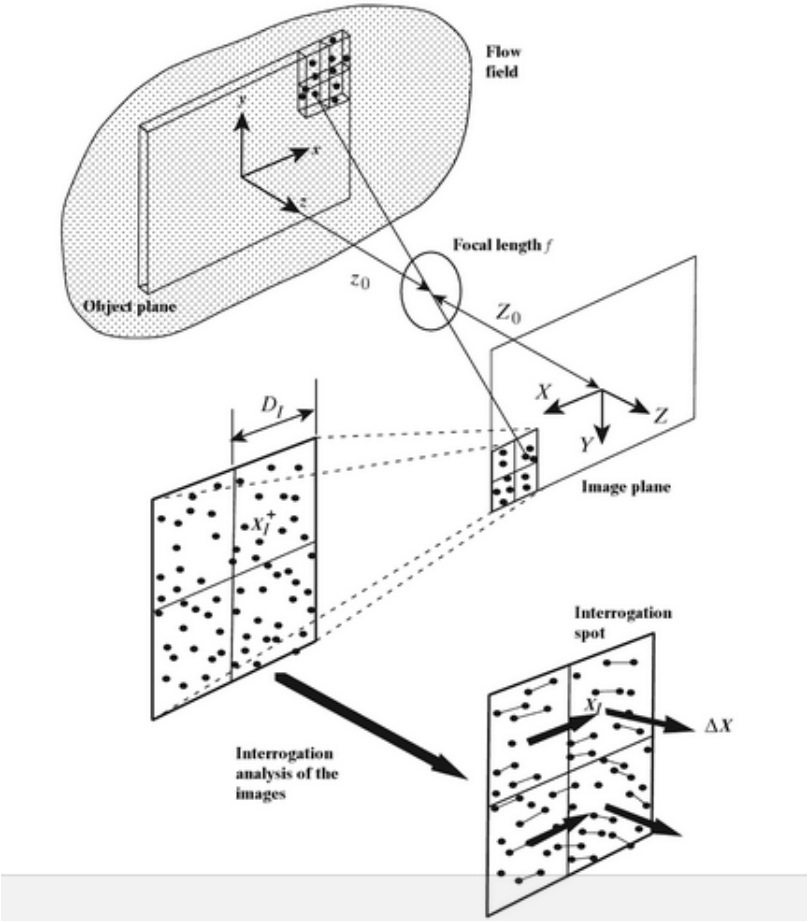


Figura 2.17: Muestra dos capturas con áreas de interrogación y el resultado de la correlación (campo vectorial). Fuente [2].

La función de intensidad $f(x)$ es conocida entre los tiempos t_1 y t_2 y se expresa en $f_1(x)$ y $f_2(x)$. Aquí, $f_1(x)$ y $f_2(x)$ también se puede considerar como áreas de interrogación uno y dos. Esto constituye la base para definir el registro en PIV.

Para un sistema de imágenes, tal como se muestra esquemáticamente en la Figura(2.18) , la distancia focal f viene dada por las distancias del objeto a la lente y de la imagen a la lente, $1/f = l/d_o + l/d_i$. Se supone que las partículas están situadas en el plano de la imagen dentro de la profundidad del campo estudiado, dz, de la lente de la cámara, ver Figura (2.17)

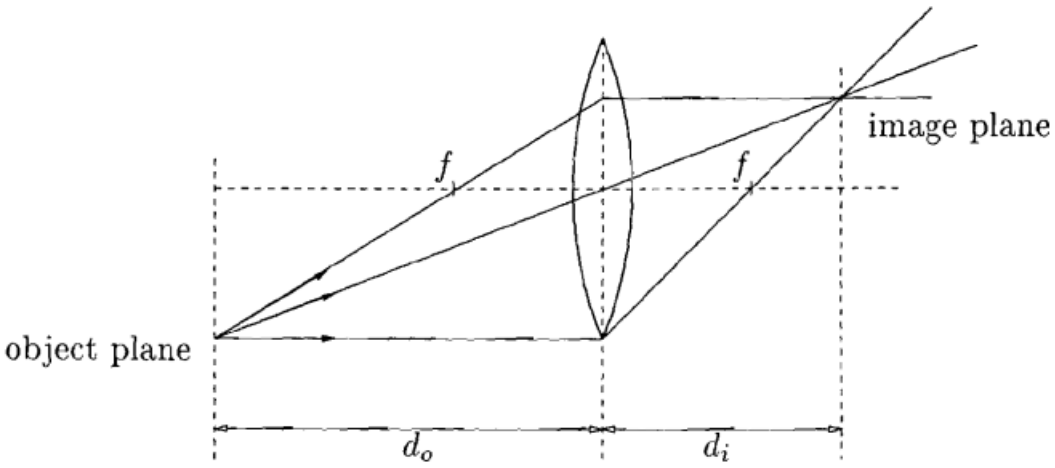


Figura 2.18: Construcción de imágenes. Fuente [12].

$$dz = 4(1 + M^{-}1)^2 f\#^2 \lambda \tag{2.12}$$

Donde $M = d_i/d_o$ es el aumento lateral, ver Figura (2.19), λ es la longitud de onda de la luz y $f\#$ es la abertura de la cámara, que es la relación entre la distancia focal f y el diámetro de abertura D_a.

$$M = \frac{y'}{y} \tag{2.13}$$

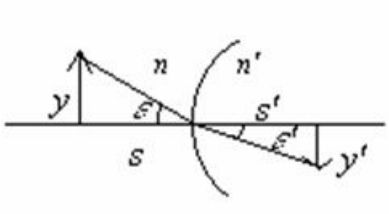


Figura 2.19: El aumento lateral es el cociente entre el tamaño (altura) de la imagen y del objeto. Fuente [12].

En estos casos, la imagen de una partícula de diámetro finito es la convolución de la respuesta puntual con la imagen geométrica de la partícula. De esta manera si la lente está limitada por difracción, la función de respuesta puntual se determina mediante la difacción de Fraunhofer, dando como resultado una función de Airy (patrón de difracción).

La ecuación de difracción de Fraunhofer se utiliza para modelar la difracción de ondas cuando el patrón de difracción es visto a una larga distancia del objeto de difracción, y también cuando se ve en el plano focal de una lente de formación de imágenes. Por lo que sobre el objeto y la pantalla incidirán ondas planas.

Patrón de difracción de Airy

Debido al tamaño pequeño de las partículas a utilizar en PIV, comienza a tener efecto la interacción entre la partícula y la longitud de onda por sus tamaños comparables, y por tanto se empieza a tener problemas de difracción.

Según Airy, el tamaño de la partícula, comparado con la longitud de onda de la luz, tiene que ser lo suficientemente grande para que sirva la utilización de las leyes de la óptica, de esta manera el patrón de Airy es un patrón de difracción marcado por lo siguiente:

- Este patrón de difracción consta de anillos concéntricos, los cuáles marcan el paso por cero, es decir, que en los anillos la intensidad de luz es cero, ver Figura (2.20)

- Cuándo se ve un punto emisor de luz muy pequeño, en ese momento se observa el patrón de difracción.

- Si se quiere ver dos puntos lejanos lo suficientemente separados, a menos que se disponga de un telescopio muy grande para distinguirlo, esto mismo se aplica a los patrones de difracción, pues se necesita un sistema óptimo muy potente que permita distinguir los patrones y no verlos superpuestos.

- Como ya se ha mencionado, la difracción de un láser resulta de la interacción de la luz con las partículas, y puede ser descrita matemáticamente. Para una partícula esférica sencilla, el patrón de difracción muestra una estructura típica de anillo.

- Si se tiene una partículas pequeña, su imagen va a ser un disco de Airy que va a tener un tamaño 'r_o', relacionado con el tamaño de la partícula pero no es el tamaño que tendría como consecuencia del aumento del sistema óptimo, pues es el tamaño asociado a la difracción. Si la partícula es más grande, ofrece un disco de Airy con un núcleo sensiblemente más pequeño.

- Hay todo un rango de tamaños en el que el tamaño del objeto que difracta es comparable a la longitud de onda, por tanto lo que manda es la difracción. Si lo que manda es la difracción, cuanto más pequeña sea la partícula, más grande es la difracción. Con lo que cuanto más grande es la partícula, menos importante es la difracción.

Por todo esto, si se quiere hacer un haz de láser que se proyecte en un punto más pequeño, lo que se necesita no es que el láser sea finito, al contrario, se necesita un láser más grueso para poder focalizar correctamente en un punto pequeño. Gracias a éste principio funciona el PIV, ya que si de una partícula de 10 μm forma su imagen para que impresione un trocito de píxel, la cantidad de luz que tendría sería ridícula, pues haría falta mucha amplificación y todas las partículas sería iguales (todas ocuparían un píxel), porque todas serían más pequeñas que el tamaño real del píxel.

No obstante, lo que ocurre con PIV es diferente, pues una partícula pequeña va a iluminar varios píxeles. De esta manera, el resultado de la respuesta puntual, que viene dado por una función de Airy, se calcula mediante un diámetro (del disco de difracción):

$$d_{diff} = 2{,}44(M+1)f\#\lambda \tag{2.14}$$

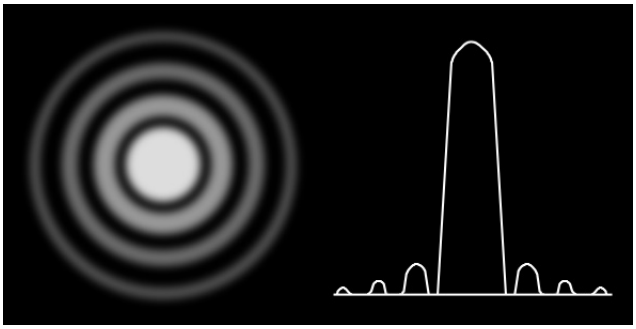


Figura 2.20: Representación de un disco de Airy. Fuente [*www.captandoelcosmos.com/la-optica-de-las-estrellas-los-anillos-de-difraccion/*].

Cuando se realiza el cálculo de la convolución del disco de Airy, con la partícula se obtiene un diámetro de imagen que va a tener una componente que representará el tamaño real de la partícula y otra que va a ser el tamaño del disco de difracción:

$$d_e = \sqrt{(Md_p)^2 + (d_{diff})^2} \tag{2.15}$$

Aquí d_e se define como el diámetro en el que la intensidad de las partículas de imagen ha disminuido al 2,5 % de su valor máximo.

En función del tamaño de la partícula los valores de Md_p y d_{diff} cambiarán:

- Si la partícula es muy pequeña $\rightarrow (Md_p)$ es despreciable $\rightarrow$ por lo tanto todo será difracción.
- Si la partícula es grande $\rightarrow (d_{diff})$ es despreciable $\rightarrow$ por lo tanto la difracción será despreciable.

En la práctica, como las lentes siempre van a estar en el límite de difracción, el diámetro efectivo de la partícula es grande es independiente del tamaño para $d_p < 10\mu m$, es decir, se consideran partículas pequeñas en las que manda la difracción cuando $d_p < 10\mu m$. Por otro lado, en partículas con $d_p > 50\mu m$ la difracción es irrelevante.

Características comunes de los patrones de difracción

- La separación angular de las características en los patrones de difracción es inversamente proporcional a las dimensiones del objeto causante de la difracción, con lo que cuanto menor es el objeto difractado más ancho es el patrón resultante, y viceversa.
- Los ángulos de difracción dependen simplemente del ratio entre la longitud de onda y la magnitud del objeto difractante.
- Cuando el objeto difractante tiene una estructura periódica, las características son más marcadas.
- La capacidad de resolver con detalle un sistema de imágenes está limitada por la difracción.

El tamaño real de las imágenes que se forman depende del tamaño de las partículas, las características del sistema de formación de imágenes y la longitud de onda de la fuente de luz.

De esta manera, la intensidad de dispersión está determinada por las propiedades ópticas, el tamaño de las partículas y la longitud de onda de la fuente de luz. En el caso de las partículas con tamaños más pequeños que la longitud de onda del haz entrante, $d_p << \lambda$, la dispersión está régimen de Rayleigh cuya energía media aumenta como $(d_p/\lambda)^4$. Para las partículas grandes,$d_p >> \lambda$, se produce dispersión geométrica, proporcional a $(d_p/\lambda)^2$. Las partículas con diámetros del orden de magnitud de la longitud de onda de la luz a la que están expuestos, $d_p \backsim \mathcal{O}(\lambda)$, están en el régimen de dispersión de Mie. En este caso, la energía promediada sobre un área de imagen de partículas viene dada por:

$$\overline{\varepsilon} \backsim \frac{I_0 D_a^4}{d_0^2 d_i^2 \Delta y_0 \Delta z_0} (\frac{d_p}{\lambda})^3, \tag{2.16}$$

Con I_0 la energía por pulso, Δy_0 y Δz_0 la altura y el espesor del plano del láser respectivamente.

Capítulo 3

Evaluación de las imágenes (Cross-Correlation)

3.1. Correlación cruzada de imágenes

El objetivo de la correlación cruzada es averiguar la distancia que el patrón de partículas se ha movido entre una imagen y otra en un intervalo de tiempo y traducir esto en una medida de la velocidad [12].

La relación entre las velocidades $\vec{u}$ y los desplazamientos de partículas $\vec{d}$ es:

$$\vec{u} = \frac{\vec{d}}{M\Delta t} \tag{3.1}$$

Donde M es el aumento lateral y Δt es el tiempo entre imágenes.

La función de correlación cruzada no se calcula sobre todas las imágenes, sino en unas partes más pequeñas que ya se han definido como áreas de interrogación (**AI**), ver la Figura (3.1).

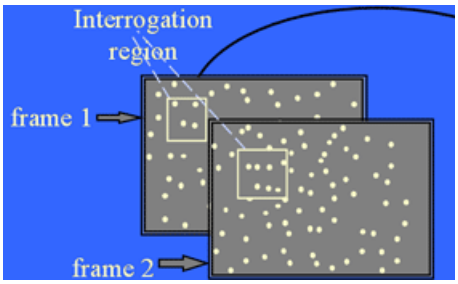


Figura 3.1: Selección de dos áreas de interrogación. Fuente [13].

La correlación cruzada sobre una **AI** devuelve un vector de velocidad. La correlación cruzada considerara el desplazamiento relativo de las **AI** que da la mejor coincidencia de patrones (picos). Este desplazamiento debe ser proporcional al promedio de la velocidad en dichas áreas de interrogación.

Las funciones de correlación cruzada se pueden calcular de diferentes maneras, aunque la forma directa para calcular una función de correlación cruzada de una muestra dimensional $R_{AB}(x, y)$, para cada $M \times N$ (ancho y alto) de las muestras puntuales $A(i, j)$ y $B(i, j)$, con $x < M$ e $y < N$ se definen por:

$$R_{AB}(x,y) = \begin{cases} \frac{1}{(M-|x|)(N-|y|)} \sum_{i=1}^{M-x} \sum_{j=1}^{N-y} A(i,j)B(i+x,j+y) & x,y \leq 0 \\ \frac{1}{(M-|x|)(N-|y|)} \sum_{i=1}^{M+x} \sum_{j=1}^{N-y} A(i-x,n)B(i,j+y) & x<0, y \geq 0 \\ \frac{1}{(M-|x|)(N-|y|)} \sum_{i=1}^{M-x} \sum_{j=1}^{N+y} A(i,j-y)B(i+x,j) & x \geq 0, y<0 \\ \frac{1}{(M-|x|)(N-|y|)} \sum_{i=1}^{M+x} \sum_{j=1}^{N+y} A(i-x,j-y)B(i,j) & x,y<0 \end{cases} \quad (3.2)$$

Para $x = 0, \pm 1, \pm 2 \ldots \pm N-1 \quad y = 0, \pm 1, \pm 2 \ldots \pm M-1$, y donde $A(x,y)$ es la subimagen correspondiente a la primera **AI**, $B(x,y)$ es la segunda zona, M es el ancho y N la altura en píxels de la **AI**.

3.1.1. Efecto de borde y pérdida de patrón

Uno de los principales problemas en la Velocimetría de Imágenes de Partículas es la pérdida del patrón. Si las dos capturas seleccionadas son del mismo tamaño, cuando se realiza el estudio de correlación en desplazamientos más o menos grandes, es bastante probable encontrar coeficientes pequeños de correlación debido al gran número de partículas que han salido de las regiones evaluadas. Por lo tanto, a todo esto se le denomina pérdida de patrón, ver Figura (3.2).

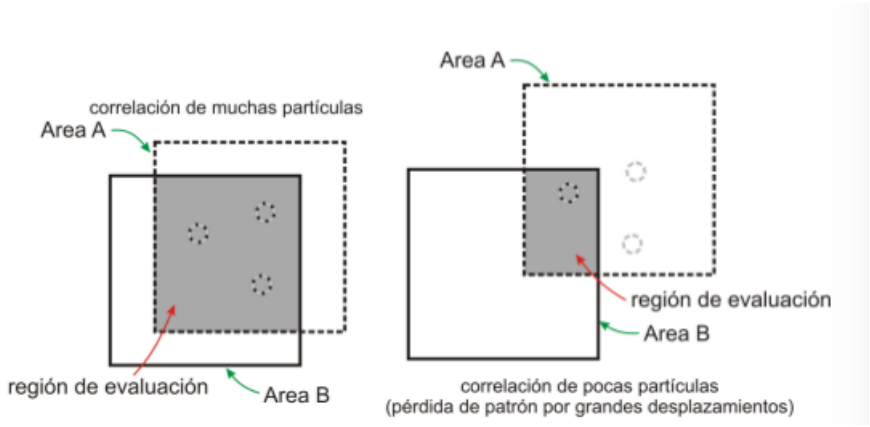


Figura 3.2: Representación de la pérdida de patrón. Fuente [13].

De esta manera al tener **AI** acotadas, es posible que aparezca el efecto de borde como consecuencia del cálculo numérico finito y además solo tiene sentido evaluar desplazamientos pequeños, sobretodo en esta técnica, pues será más sencillo captar el mismo patrón en la segunda captura, ver Figura (3.3).

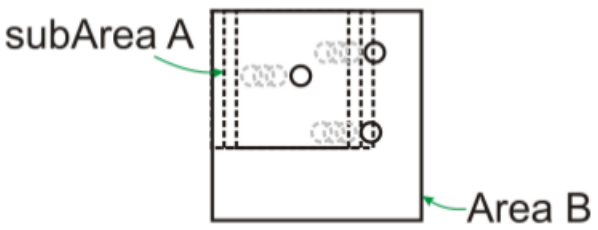


Figura 3.3: Representación del efecto de borde. Fuente [13].

La solución consiste en seleccionar áreas de **AI**, sean A y B de distinto tamaño. Una correcta selección de estos tamaños siempre tiene que respetar el máximo desplazamiento en cada una de las direcciones, *n (píxeles ancho)* y *m (píxeles alto)*, que se intuye de las capturas seleccionadas. Dicho desplazamiento máximo de píxeles se conoce como D y se tiene que cumplir que $n - m > D$.

De esta manera será importante interpretar y comprobar el cálculo de la velocidad máxima real de las partículas en la región seleccionada y que se encuentran en le plano del haz laser emitido. Usando los comandos de vídeo de MATLAB, el Δt entre las capturas seleccionadas y el máximo desplazamiento de las **AI**, conocido como (D).

Por lo tanto, esta velocidad máxima real es:

$$v_{max} = \frac{D}{M}\Delta t^{-1} \tag{3.3}$$

Donde D es desplazamiento máximo en píxeles y M es la relación *Píxeles/metro*.

3.1.2. Función de correlación cruzada a través de transformadas finitas de Fourier

El método directo para calcular la correlación cruzada se vuelve rápidamente bastante complicado de aplicar cuando se analizan conjuntos de datos más grandes. Una manera más eficiente de estimar las funciones de correlación cruzada es usando transformadas rápidas de Fourier (FFT_s). Esto reduce el cálculo de de las operaciones de $\mathcal{O}[N^4]$ a operaciones de $\mathcal{O}[N^2 \log_2 N]$ en el caso de una correlación bidimensional. Cuando se utilizan las transformadas de Fourier, se aprovecha el teorema de correlación que establece que la correlación cruzada de dos funciones es equivalente a una multiplicación compleja conjugada de sus transformadas de Fourier:

$$R_{AB} \leftrightarrow \hat{A} \cdot \hat{B}^* \tag{3.4}$$

Dónde $\hat{A}$ y $\hat{B}$ son las transformadas de Fourier de A y B, respectivamente y $\hat{B}^*$ representa el conjugado complejo de $\hat{B}$.

En el Anexo (1) se profundiza la manera de utilizar y en qué consisten las transformadas de Fourier.

El uso de FFT_s supone tratar los datos de forma periódica. La periodicidad puede dar lugar al *aliasing* si las partículas se han movido una distancia mayor que la mitad del tamaño del **AI**.

El *aliasing* es el efecto que causa que señales continuas distintas se vuelvan indistinguibles cuando se muestrean digitalmente.

La solución a los problemas de *aliasing* es incrementar el **AI** o reducir el tiempo entre las imágenes (Δt). Un problema quizás más grave con las FFT_s es que los errores de sesgo se producen si estos no se tienen en cuenta. Debido al tamaño finito de las **AI** la superposición de las imágenes se reduce con el aumento del desplazamiento. Este sesgo resulta de una subestimación de la magnitud máxima para todos los desplazamientos distintos de cero. Por tanto se debe aplicar una función de ponderación a la función de correlación cruzada para evitar este sesgo. Esta función de ponderación es la convolución de las funciones de ponderación de la muestra (que deben ser iguales para un punto de todos los de la imagen y cero en otro lugar). El sesgo se elimina mediante la división de la función de correlación con la función de ponderación efectiva que tienen forma de pirámide, véase la Figura (3.4).

Cabe señalar que muchas implementaciones de FFT_s (por ejemplo, la de Matlab) barajan los datos de salida para que el componente DC (componente de frecuencia cero de la transformada de Fourier) con índice 1 y la mayor frecuencia positiva y negativa en el índice $N/2-1$. Para conseguir un espectro creciente (y los desplazamientos resultantes) los datos han de ser reorganizados ($fftshift$ en Matlab), el cual reordena los puntos del espectro en función de la frecuencia.

3.1.3. Detección de picos e interpolación subpixel

El método consiste en mover la primera **AI** seleccionada sobre la segunda, de manera que se puedan sumar los productos de los valores que coincidan en ambas. Por lo tanto, cada valor es guardado el plano

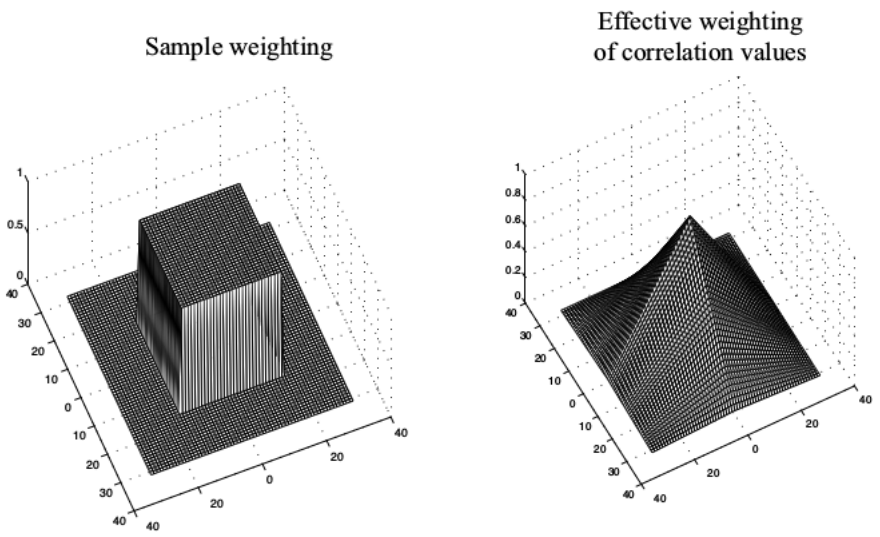


Figura 3.4: Efectividad de la correlación de valor ponderado en las FFT_s, basado en los cálculos de la correlación cruzada. La función de ponderación efectiva a la derecha es la convolución de las funciones de ponderación de dos muestras como una de las de la izquierda. Fuente [9].

de correlación cruzada [13].

En la Figura (3.5) se muestra el desplazamiento de las **AI**, la imagen (a) muestra dos cuadrados de 4×4 píxeles, el del centro (**AI** inicial) y el cuadrado desplazado arriba en gris (la segunda **AI**). La imagen (b) muestra una zona donde se superponen las dos imágenes al completo debido a que no se desplazan. La imagen (c) muestra un movimiento horizontal positivo de 2 píxeles donde se observa claramente la zona en la que hay solapamiento, de esta manera todas las sumas de los productos de estas superposiciones se guardan en el ya mencionado plano de correlación cruzada, cuyas dimensiones también están en píxeles y son de $2M - 1 \times 2N - 1$. Este tamaño es finito debido a que en la práctica únicamente se considera o se registra el desplazamiento mientras exista superposición (solape) para extraer los picos.

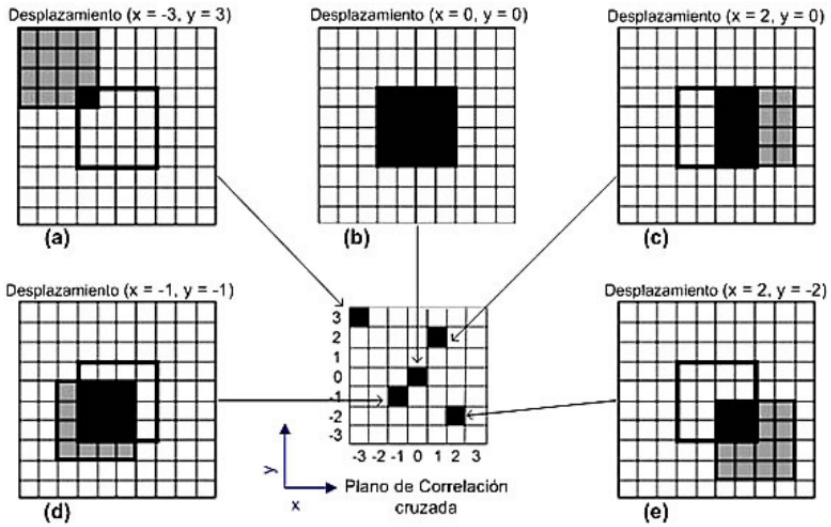


Figura 3.5: Desplazamiento de las **AI**. Fuente [12].

En el plano de correlación de la Figura (3.6), se observa de forma gráfica la suma de los productos, donde el máximo (pico) indica el promedio de los desplazamientos que tienen lugar las partículas.

Cuando se ha realizado la correlación cruzada la medida del desplazamiento se encuentra detectando la localización del pico de correlación más alto. En la detección de dicho pico se generará una incertidumbre de $\pm 1/2$ píxeles en su ubicación.

Sin embargo, la precisión se puede aumentar sustancialmente mediante un ajuste en la curva o mediante interpolación. Esto puede sonar como ïnventarün nueva información que no ha sido medida. Pero el procedimiento puede defenderse con el argumento de que la relación se basa en las imágenes de varias partículas. Si, por ejemplo, un **AI** contiene diez imágenes de partículas y ocho partículas tiene un desplazamiento de 3 píxeles y dos un desplazamiento de 2 píxeles, el pico máximo de correlación se situaría en 3 píxeles, pero una interpolación de subpixel podría predecir el desplazamiento correcto de 2.8 píxeles ya que la correlación en dos píxeles será mayor que el de cuatro píxeles.

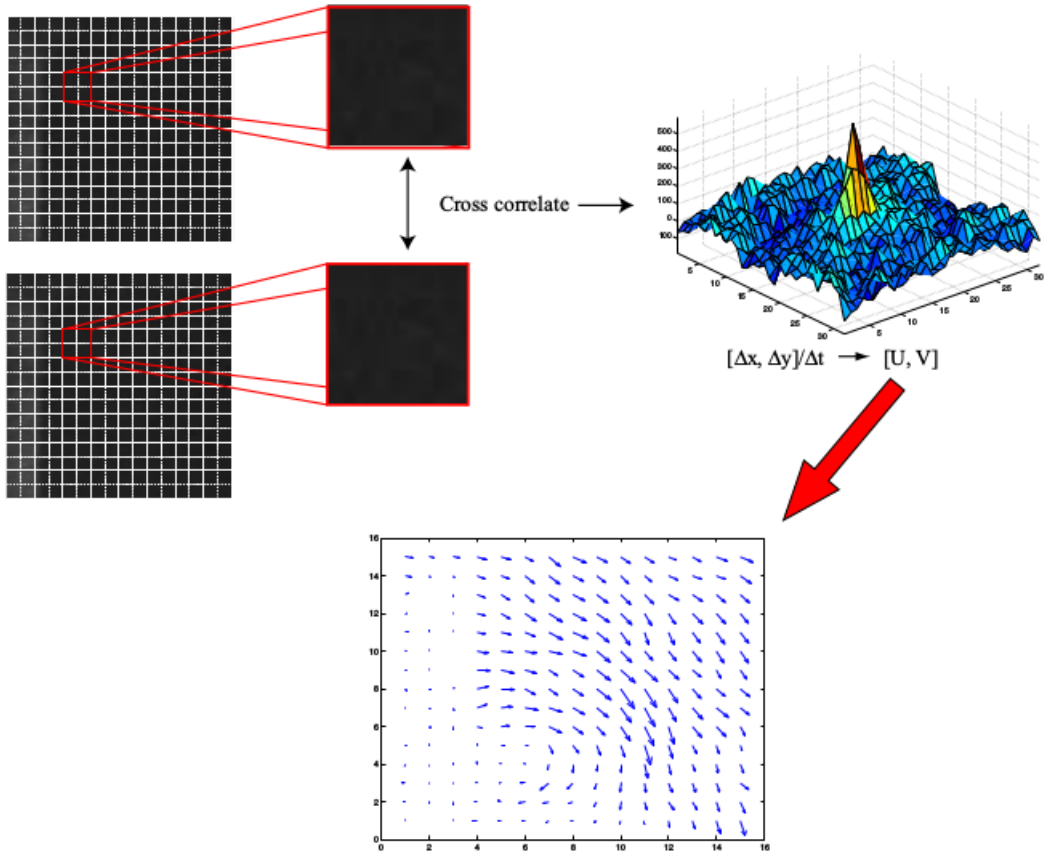


Figura 3.6: Ilustración de cómo se extrae la información de velocidad de un par de imágenes. Fuente [9].

La forma más común de realizar la interpolación de subpixel es usar un estimador de tres puntos. Cuando se ha detectado el pico máximo en $[i, j]$, los valores vecinos se utilizan para ajustar una función al pico. En el caso de un pico Gaussiano se ajusta cuando se supone que el pico tiene la forma $f(x) = Cexp[-(x_0 - x)^2/k]$, los desplazamientos se encuentran mediante la siguiente ecuación:

$$\begin{cases} x_0 = i + \frac{lnR_{(i-1,j)} - lnR_{i+1,j}}{2lnR_{(i-1,j)} - 4lnR_{(i,j)} + 2lnR_{(i+1,j)}} \\ y_0 = j + \frac{lnR_{(i,j-1)} - lnR_{i,j+1}}{2lnR_{(i,j-1)} - 4lnR_{(i,j)} + 2lnR_{(i,j+1)}} \end{cases} \tag{3.5}$$

A menudo se utilizan **AI** superpuestas (sobremuestreo), véase la Figura (3.6), para maximizar el uso de la información disponible y el cumplimiento del teorema de muestreo de Nyquist (que es matemáticamente

posible si la señal está limitada en banda y la tasa de muestreo es superior al doble de su ancho de banda).

De esta manera, en este proyecto se va a realizar una simulación en Matlab de este método de correlación, implementando una interfaz gráfica que sea didáctica y sencilla de comprender y usar.

Capítulo 4

Implementación de la interfaz gráfica

4.1. Planteamiento

Anteriormente se ha introducido y explicado en que consiste la técnica PIV. Se puede deducir que esta técnica muestra unos mejores resultados en el caso de que las imágenes sean de alta calidad, cuando se observe de una manera clara el patrón de partículas y cuando la distancia que se han desplazado unas de otras no sea muy grande. Sin embargo, debido a que el PIV está basado en correlaciones estadísticas de imágenes, está sujeto a ciertos errores provocando resultados menos precisos [15].

Dichos errores suelen provenir de:

- Número finito de partículas trazadoras.
- El tamaño del volumen de la muestra a analizar.
- Resolución de la imagen.
- Gradientes de velocidad.
- Variaciones en la intensidad de la imagen (iluminación no uniforme).

Estas fuentes de errores se pueden tratar y solucionar. Por lo tanto en este capítulo se describen y justifican las partes y procesos que tiene la interfaz gráfica para obtener los mejores resultados posibles, intentando corregir todos los factores que pueden provocar errores. De esta manera el código está dividido en tres partes, que corresponden a tres interfaces distintas creadas con el programa Matlab:

- La primera está dedicada al análisis de un vídeo, en el cual se puede elegir en que parte del mismo centrar la correlación y obtener dos capturas consecutivas conociendo el Δt que las separa. **Cargarvideopiv.m**.

- La segunda se utiliza para el análisis de las dos capturas obtenidas en la primera interfaz, aplicando distintos filtros sobre las imágenes y seleccionando el patrón que se desea estudiar. **Pruebaestudioimagen.m**.

- La tercera muestra los resultados de la correlación entre las dos imágenes, ya sea en campo de vectores (Vector que muestra el desplazamiento de las partículas) y en campo escalar (mostrando distinta intensidad de colores en función de la magnitud de la velocidad). **Resultados.m**.

En la Figura (4.1) se observan los estados en los que se divide el sistema implementado.
Por otro lado, en el Anexo (2) se muestra el código que corresponde a dichas interfaces, incluyendo las funciones empleadas que forman parte de los 'toolbox' contenidos en el programa Matlab [14].

En el Anexo (3) se encuentra la guía de usuario de la interfaz, donde se explica la manera de utilizarla y que orden seguir.

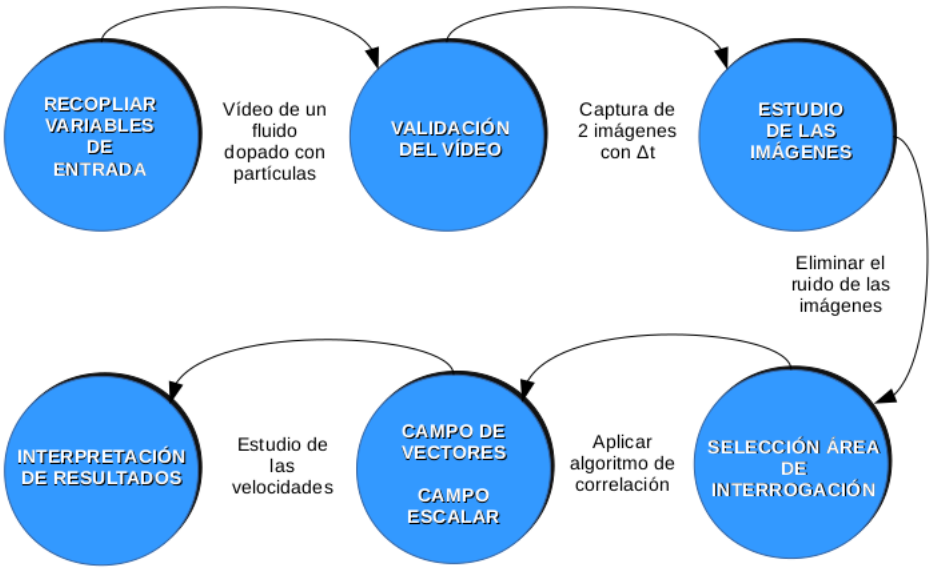


Figura 4.1: Diagrama de estados del sistema. Fuente [Autor].

4.2. Primera interfaz

El propósito de esta primera interfaz es seleccionar en que intervalo de tiempo de un vídeo se quiere realizar la correlación. En primer lugar como se describirá en el capítulo (x) se graba un vídeo con una cámara de alta velocidad en la instalación PIV construida en el laboratorio de Física Aplicada, o simplemente se descarga un vídeo de un fluido sembrado con partículas que obtenemos de la red, además existe la opción de cargar dos imágenes de alta calidad que muestren perfectamente el desplazamiento de las partículas.

En la Figura (4.2), se observa esta primera interfaz, distinguiendo todos los botones y casillas que la forman.

En primer lugar se definen los parámetros previos que harán falta para las funciones que implementa la interfaz en:

function cargarvideopiv_OpeningFcn(hObject, eventdata, handles, varargin)

Dentro de ella y una vez se obtiene cualquier vídeo, la interfaz lo carga mediante la función *VideoReader* de Matlab y mediante *VideoReader.getFileFormats()* se indica que formatos de vídeo admite el programa y los carga, esto varía en función del sistema operativo, ya sea Windows, Linux o Mac Os X. En este caso se ha implementado con Linux, por lo tanto únicamente reproduce archivos *.avi*, lo cual no supone un problema pues la mayoría de vídeos disponibles son de este formato y además muestran una gran calidad.

Para empezar el temporizador y utilizar sus funciones se utiliza la función *timer*. Gracias a esta se puede saber en que instante de tiempo se encuentra en cada momento el vídeo.

handles.hTimer = timer('ExecutionMode','fixedRate','Period',1)

Una vez definidos estos parámetros, se describen los *callbacks* de los botones y partes de la interfaz que se emplean durante el proceso.

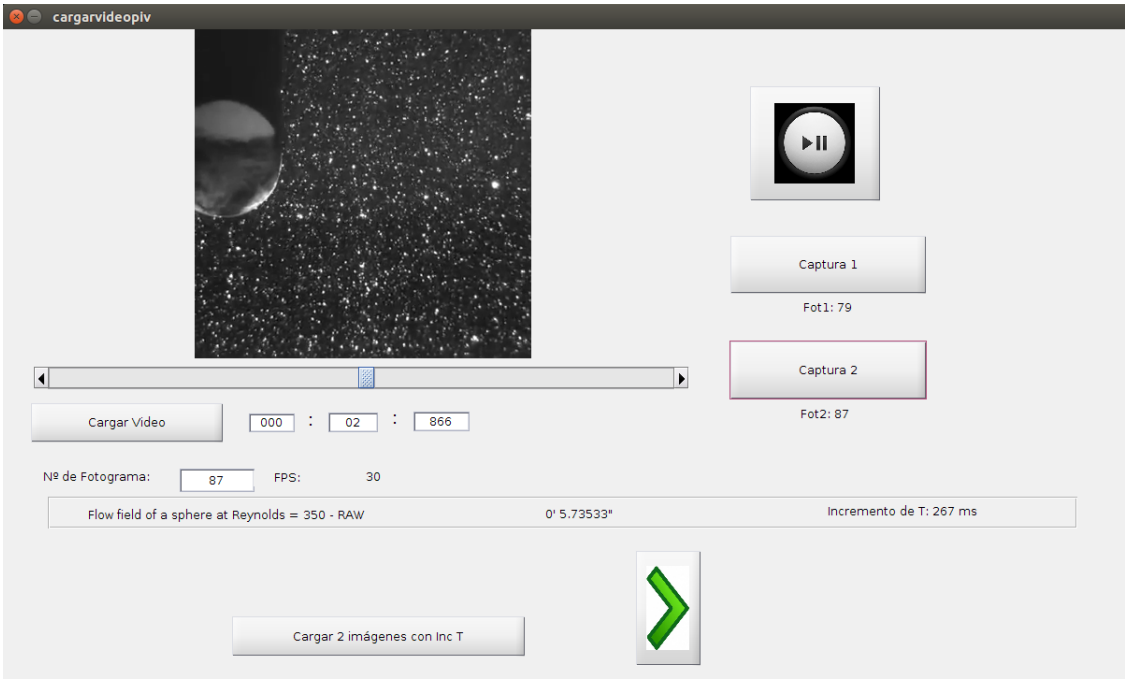


Figura 4.2: Primera interfaz 'Cargarvideopiv.m'. Fuente [Autor].

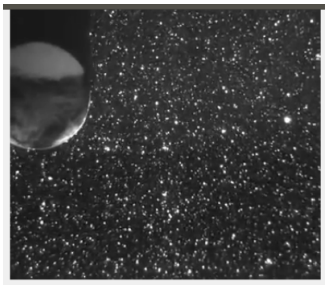


Figura 4.3: Axes donde se muestra el vídeo.

4.2.1. Botón 'Cargar Vídeo'

- *function cargarvideopiv_OpeningFcn(hObject, eventdata, handles, varargin)*

A la hora de cargar el vídeo utiliza la función *uigetfile()*, en la cual si se define como:

[Path, name]=uigetfile(), ofrece la ruta de acceso, nombre y formato del video seleccionado, permitiendo elegir cualquier fichero de vídeo aceptable disponible en el ordenador con el que se trabaje.

De esta manera mediante:

objVideo = VideoReader('filename')

Se crea el objeto *objVideo* para leer los datos del archivo llamado *'filename'*.

Una vez seleccionado el fichero de vídeo y cargado, se le asigna los ejes *(Axes)* en los que se quiera ver el vídeo reproduciendo.

set(handles.figureVideo,'CurrentAxes',handles.ejesVideo);

Figura 4.4: Barra de datos.

Por otro lado, es importante que se muestren los datos y características de vídeo, dichos parámetros se extraen mediante las funciones de Matlab:

- *objVideo.NumberOfFrames*, número de fotogramas del vídeo.
- *objVideo.Duration*, duración del vídeo.
- *objVideo.Width objVideo.Height*, tamaño del vídeo.
- *objVideo.FrameRate*, ratio entre cada fotograma (parámetro muy importante en el cálculo de Δt).

Algunos de estos parámetros se muestran en la barra blanca que se encuentra debajo del *Axes* del vídeo, así pues se indican el nombre, la duración del vídeo, los FPS (Fotogramas por segundo) y el incremento de tiempo entre la primera captura y la segunda. Los tres primeros datos se extraen una vez se ha cargado el vídeo, pues son parámetros que vienen por defecto con el vídeo. El Δt aparecerá en esta misma barra cuando se guarde la segunda captura y nos permita obtener este dato.

Por otro lado, al cargar el vídeo, es necesario hacer el seguimiento temporal completo, para ello, ya pulsando este botón se tienen que activar los textos que forman el cronómetro, incluyendo milésimas, segundos y minutos:

Figura 4.5: Botón Cargar Video.

- *set(handles.minutos,'String','00');*
- *set(handles.segundos,'String','00');*
- *set(handles.milesimas,'String','000');*

Para que estos parámetros se vayan actualizando se sincronizan con el temporizador *timer*, es importante definir las variables que posteriormente van a cambiar constantemente con el avance del vídeo, el comando global se utiliza para este cometido. Por otro lado está dividido en tres partes, los datos están definidos por la función *MinSegmil* ofreciendo una información continua de cada instante de tiempo en Minutos, Segundos y milésimas. Estos parámetros de tiempo se sincronizan por medio de la *'timerFcn'*, la cual es necesario activarla por medio de un Callback:

set(handles.hTimer,'TimerFcn',@timerCallback,handles);

Figura 4.6: Temporizador

4.2.2. Función 'MinSegMil'

Esta función define los valores de las milésimas, segundos y minutos según el número de fotograma en el que se encuentre el vídeo en ese momento.

De esta manera mediante los parámetros de entrada *num*, que define el fotograma del momento y *ratio*, que indica la diferencia que hay entre cada fotograma. Utilizando estos datos se puede estimar los valores de las tres unidades de medida temporal $[Min \quad Seg \quad Mil]$, como se muestra en la Figura (4.7).

```
function [M,S,m] = MinSegMil(num,ratio)
    seg = (num-1)/ratio;
    M = floor(seg/60);
    S = floor(seg - 60*M);
    m = floor((seg - M*60 - S)*1000);
```

Figura 4.7: Función para el cálculo de minutos, segundos y milésimas. Fuente [Autor].

4.2.3. Slider de seguimiento del vídeo

- *function slidervideo_Callback(hObject, eventdata, handles)*

Esta barra va avanzando conforme avance el vídeo mientras está en reproducción. El funcionamiento consiste en grabar continuamente para cada instante el fotograma que se está reproduciendo.

Para ello se define mediante *global* el valor de *cont*, el cual recoge el fotograma y lo guarda, actualizándose constantemente. *Global* declara variables que posteriormente van a ser modificadas.

```
global cont;
cont = round(get(hObject,'Value'));
```

Una vez declarado el fotograma, mediante la función *MinSegMil*, se extraen los datos temporales puntuales de cada fotograma que se está observando. De esta manera, es mucho más sencillo obtener un par de capturas del vídeo con alta precisión espacial y temporal, pues será más asequible detener el vídeo en el momento en el que más convenga, ya sea por cantidad de partículas o porque se observa claramente le patrón deseado. Además, gracias a esta barra, no es necesario pulsar *Play-Pause* para avanzar el vídeo a la posición deseada.

```
[Min, Seg, Mil] = MinSegMil(cont,handles.video.ratioFotogramas);
```

4.2.4. Botón Play-Pause

- *function botonPlayPause_Callback(hObject, eventdata, handles)*

Figura 4.8: Slider del movimiento.

Para pausar y comenzar la reproducción en cualquier momento emplea el Callback *Play-Pause*. De manera que cada vez que se pulse este botón la reproducción se detenga en el instante deseado y que mejor convenga.

Figura 4.9: Botón Play-Pause.

Para ello se declaran las variables *cont* y *temp* mediante *Global*. Una vez hecho esto, se inicia el temporizador desde el valor de *temp* 0, hasta aquí aún no se ha pulsado el botón 'off'. La reproducción comenzará cuando se permita 'enable', es decir, se pulse el botón 'on' y la variable *cont* almacena desde el primer fotograma hasta el último. Si se vuelve a pulsar el botón 'on', la variable *temp* no tiene valor y el temporizador se para, de esta manera se repite el procedimiento cada vez que se pulse el botón, ver Figura (4.10).

```
global cont temp
    if temp == 0
        start(handles.hTimer);
        enable_reproduccion(handles,'off');
        if handles.video.numFotogramas == cont
            cont = 1;
            enable_reproduccion(handles,'on');
        end
    else
        enable_reproduccion(handles,'on');
        stop(handles.hTimer);
    end
    temp = ~temp;
```

Figura 4.10: Código del botón Play-Pause

4.2.5. Botones Captura 1 y Captura 2

Debido a que el objetivo es obtener dos imágenes consecutivas mediante dos pulsadores se ordena la captura de los fotogramas requeridos. Destacando que cuando se captura la segunda imagen, aparece automáticamente el cálculo de la ecuación (4.1), que indica el intervalo de tiempo que ocurre entre cada fotograma.

Para captar el fotograma seleccionado se declara la variable *fot_grab*, que estará formado por dos valores (dos capturas), de manera que cada vez que se pulse cualquiera de los dos botones, el fotograma definido por *cont* se graba. Además se indica el número de fotograma seleccionado en los recuadros *Fot1* y *Fot2*:

- *set(handles.labelFot1,'String',sprintf('Fot1: %d',cont));*
- *set(handles.labelFot2,'String',sprintf('Fot2: %d',cont));;*

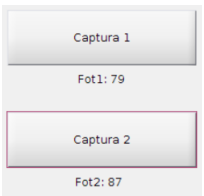


Figura 4.11: Botones Captura 1 y 2.

Existe la opción de guardar en el ordenador cada imagen seleccionada, mediante *rgb=getimage('Axes')* e *imwrite(rgb,fName);* de esta manera siempre podremos cargar las que queramos en la siguiente interfaz.

Una vez se ha grabado el segundo fotograma, aparece en la barra blanca de los datos el Δt en segundos que hay entre los fotogramas seleccionados, siguiendo la ecuación:

$$\Delta t = round[\frac{fot\_grab(2) - fot\_grab(1)}{RatioFotogramas \;\; 1000}] \quad segundos \tag{4.1}$$

4.2.6. Siguiente interfaz 'Next_Callback'

Por último, una vez obtenidos todos los parámetros necesarios, se llama a la siguiente interfaz, que únicamente se abrirá en el caso de que estén definidas una serie de variables de entrada, dichas variables se obtienen de los parámetros obtenidos en esta primera interfaz.

El comando para llamarla y que incluye las cuatro variables de entrada es:

Figura 4.12: Botón siguiente interfaz.

```
pruebaestudioimagen(handles.video.videoFotogramas(:,:,:,fot_grab(1)),
   handles.video.videoFotogramas(:,:,:,fot_grab(2)),
                        (fot_grab(2)-
    fot_grab(1)),handles.video.ratioFotogramas);
```

4.3. Segunda interfaz

El propósito de esta segunda interfaz es el tratamiento y la mejora si es necesario de las imágenes, es decir, realiza el estudio de las imágenes y las mejora en el caso de ser necesario corregir ciertos errores experimentales. La etapa de selección de la región a analizar es importante, pues no siempre se puede ajustar dicha zona con la cámara. Además los resultados pueden verse afectados debido a que la selección de estas zonas puede no estar en las mismas posiciones y no tiene sentido analizar zonas de la imagen en las que no interesa conocer la velocidad.

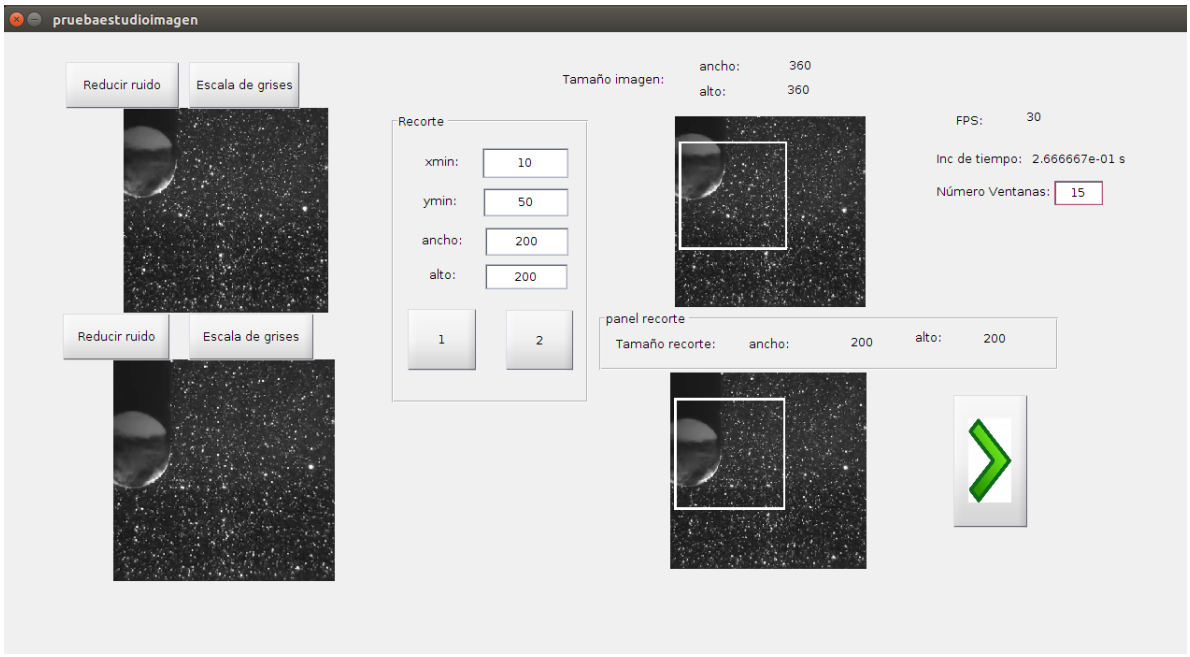


Figura 4.13: Segunda interfaz 'Pruebaestudioimagen.m'. Fuente [Autor].

En la Figura (4.13), se observa esta segunda interfaz, distinguiendo todos los botones y casillas que la forman.

Las variables de entrada de esta interfaz se introducen en la función inicial:

function pruebaestudioimagen_OpeningFcn(hObject, eventdata, handles, varargin)

- *varargin {1} :* La primera imagen capturada.
- *varargin {2} :* La segunda imagen capturada.
- *varargin {3} :* El incremento de los dos fotogramas seleccionados. *Fot2-Fot1.*
- *varargin {4} :* Los fotogramas por segundo del vídeo estudiado. *FPS.*

Al tener definidos el incremento de fotogramas (Fot2 - Fot1) y los fotogramas por segundo (FPS), se puede definir de nuevo el Δt.

- *set(handles.textoFPS,'String',num2str(handles.datos.fps));*
- *set(handles.deltaT,'String',num2str(sprintf(' %d s',(handles.datos.delta_fot)/handles.datos.fps)));*

Una vez definidos estos parámetros, se describen los *callbacks* de los botones y partes de la interfaz que se emplean durante el proceso.

4.3.1. Botón Quitar Ruido

Este botón se encarga de eliminar el ruido de las capturas seleccionadas, en el caso de que fuese necesario. Para comprender lo que provoca la contaminación en las imágenes se van a estudiar los tipos de ruido y el filtro empleado en esta interfaz para su eliminación [16].

function outruido1_Callback(hObject, eventdata, handles)

Este comando se aplica a las dos capturas, pulsando los dos botones correspondientes, ver Figura (4.14).

La presencia de polvo, ruido u objetos en movimiento generan una fuente de error que dificulta la medida de la velocidad del fluido, ya que es posible que no siga el patrón adecuado del flujo. Con lo cual aplicar un filtro previo, será bueno para el análisis, pues se utilizarán imágenes en las que únicamente salgan partículas.

Como ya se ha dicho, la primera opción en el estudio de la imágenes es aplicar un filtro de ruido, esta parte es opcional, pues si las imágenes que hay que estudiar son de alta calidad y no se aprecia la presencia de contaminación en la imagen, no es necesario aplicar filtro. No obstante, es necesario saber interpretar y conocer los diferentes tipos de ruido que pueden aparecer y los filtros que se pueden aplicar en Matlab.

A la hora de tratar el ruido de la imagen, hay que tener en cuanta varios factores, definidos por el esquema de la Figura (4.15).

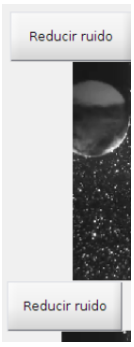


Figura 4.14: Botón quitar ruido

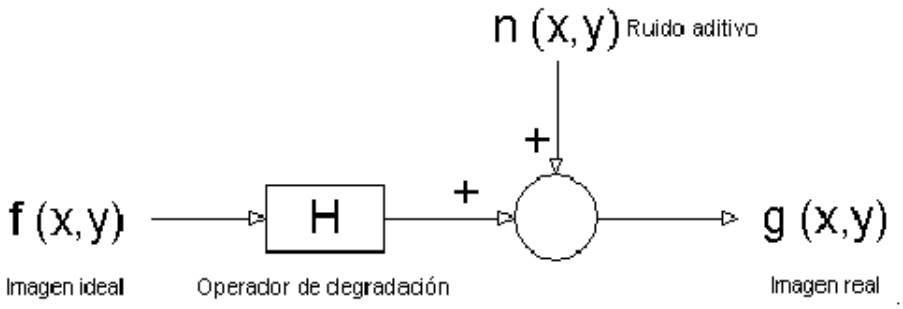


Figura 4.15: Esquena de la influencia del ruido

Concepto de ruido y tipos

Se define ruido como una información no deseada que contamina (degrada) la imagen. Se manifestará generalmente en píxeles aislados que toman un valor de gris diferente al de sus vecinos. En las imágenes pueden haber diferentes tipos de ruido:

- Ruido Gaussiano.
- Ruido Impulsional.
- Ruido Uniforme.

  Frecuencial.

Multiplicativo.

Ruido Gaussiano

- El valor final del píxel es el ideal más una cierta cantidad de error.
- Puede describirse como una variable *gaussiana* que sigue una distribución normal:

$$P \quad (g(x,y) - \sigma < f(x,y) < g(x,y) + \sigma) \quad = \quad 70\,\%$$

$$P \quad (g(x,y) - 2\sigma < f(x,y) < g(x,y) + 2\sigma) \quad = \quad 90\,\%$$

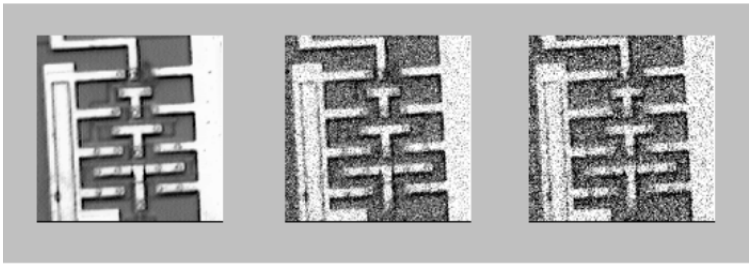


Figura 4.16: Ruido Gaussiano.

- Provoca algunas imperfecciones en la imagen a estudiar.
- La mayoría de veces es provocado a causa de algunos componentes electrónicos, ya sean sensores o digitalizadores.
- El espectro de energía se mantiene igual para cada una de las frecuencias:

  Provoca cambios en la imagen entera.

  La intensidad de todos y cada uno los píxeles muestra irregularidades.

Ruido Impulsional 'Sal y Pimienta'

- El valor tomado por el píxel carece de relación alguna con su valor ideal, pues coge valores demasiado altos o demasiado bajos.

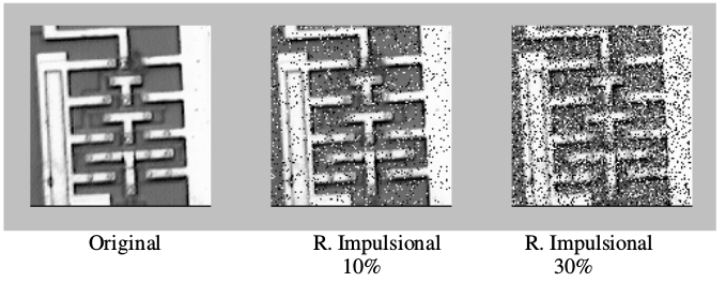


Figura 4.17: Ruido 'Sal y pimienta'

- Cuando coge el valor máximo es *'Sal'* y cuando coge el valor mínimo es *'Pimienta'*.

Ruido uniforme

- El ruido provocado en la imagen muestra una distribución homogénea.
- La probabilidad de coger algún valor gris dentro de cualquier intervalo acotado es constante.

  Ruido uniforme frecuancial: La imagen obtenida es la real más una interferencia de señal periódica, (senoide, cosenoide...).

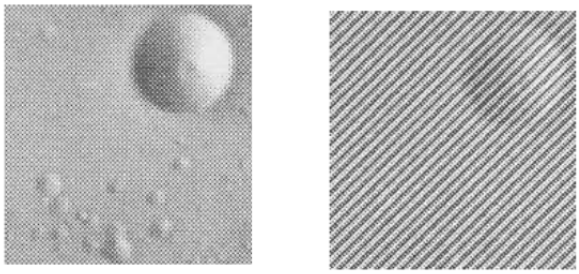


Figura 4.18: Ruido uniforme frecuencial

Ruido uniforme multiplicativo: La señal obtenida es el resultado de la superposición de dos señales.

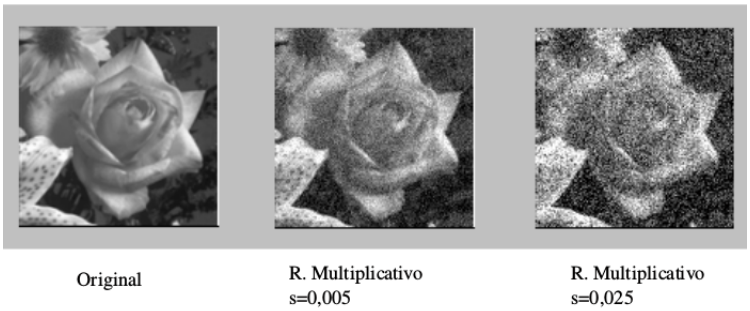


Figura 4.19: Ruido uniforme multiplicativo

Implementación 'Filtro de Mediana'

La interfaz implementada aplica un filtro de mediana, este se emplea bastante en la corrección del ruido aleatorio ya que mantiene perfectamente la nitidez de la imagen. Por lo tanto, dicho filtro elimina el tipo de ruido conocido como *'sal y pimienta'*, en el cual los píxeles de la imagen son bastante diferentes en color o intensidad a los píxeles periféricos, es decir, que el píxel que provoca ruido carece de relación alguna con estos píxeles que se encuentran en la periferia. Por lo general, el ruido descrito afectará únicamente a una pequeña parte de los píxeles que forman la imagen.

Al observar la imagen se podrán distinguir los puntos blancos sobre los puntos negros y viceversa, de ahí el término sal y pimienta. En este caso las partículas son los píxeles a corregir, las cuales representan el ruido 'sal' (puntos blancos) de la imagen. Con lo cual se consigue una imagen con objetos que no se desean y posteriormente dicha imagen se resta a la original, formando una imagen en la que sólo aparezcan partículas.

La idea de este filtro es recorrer píxel a píxel toda la imagen, reemplazando cada valor de intensidad del píxel correspondiente por la mediana de los valores de intensidad de los píxeles de su vecindad (incluyéndolo a él mismo). Esta vecindad también se conoce como ventana, y su tamaño más común es de 3×3

obteniendo un total de nueve elementos.

La mediana se consigue ordenando todos los valores de los píxeles vecinos y se reemplaza por el valor del píxel que se encuentra en medio de todos y cada uno de los valores numéricos organizados y ordenados, ver figura (4.20).

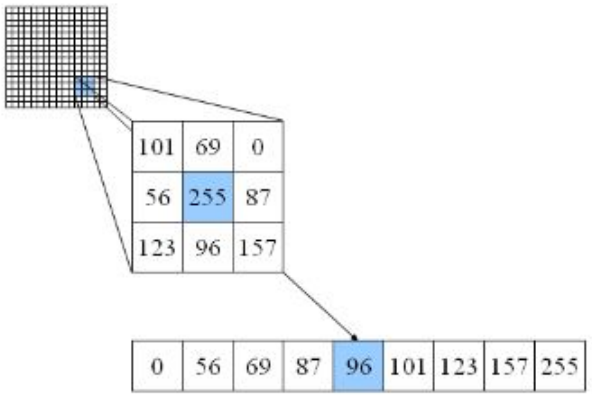


Figura 4.20: Ejemplo de procesamiento de píxel de una imagen mediante el filtro de mediana. Fuente [16].

En el medio del vecindario de la Figura (4.20) está marcado el píxel a reemplazar (en azul) con valor de 255, y abajo se ven los valores a considerar ordenados. Como se puede ver, el valor del píxel central es el 96 (en azul) y este es el que se va a sustutuir por el valor anterior (255), pues se considera que el valor 96 es más representativo que el resto de la región tratada.

Para implementar la corrección de este tipo de ruido, se utiliza la función de MATLAB *medfilt2*, el cual aplica un filtro de mediana las dos capturas seleccionadas.

Por lo tanto, se podría decir que este tipo de ruido son las motas de polvo que se encuentran dentro de las ópticas de la cámara, o bien una cámara CCD que no funcione correctamente y por tanto realizará las capturas de forma errónea. Utilizar este tipo de filtros conlleva unas ventajas y unas desventajas.

- **Ventajas:**

  Disminuye el ruido impulsional (Sal y Pimienta).

  Anula efectos erróneos.

  Conserva el contorno de las imágenes.

- **Inconvenientes:**

  Disminuya la calidad mediante la pérdida de detalles (Puntos, lineas...).

  Achata las esquinas de los objetos.

  Movimiento de los bordes de la imagen.

Para aplicar este filtro, se cogen las capturas seleccionadas en la interfaz anterior y se les aplica el comando *medfilt2*, el funcionamiento consiste en restar la imagen original a la imagen resaltando el ruido impulsional, de esta manera la imagen resultante estará libre de este tipo de ruido, siguiendo el siguiente código:

```
imagen1=getimage(handles.axes1);
imagen1=imnoise(imagen1,'salt&pepper',0.1);
f_media=medfilt2(imagen1);
```

Una vez aplicados los filtros, únicamente se observarán en los mismos *Axes* las imágenes resultantes.

4.3.2. Panel Recorte

Una vez seleccionado el filtro de ruido adecuado para las dos imágenes, si fuera necesario, la interfaz ofrece la función de seleccionar que zona queremos analizar y de esta manera obtener los dos patrones definitivos.

Para ello se busca el área seleccionada de manera que en estas dos capturas se observe perfectamente el grupo de partículas que sigue un patrón determinado.

(a) (b)

Figura 4.21: (a) Panel de selección de las zonas deseadas y (b) Descripción gráfica de los parámetros.

En el centro está el *Panel Recorte*, ver Figura (4.21). Mediante este cuadro se crea un rectángulo de cierto tamaño sobre cada imagen. Dichos rectángulos corresponden con los patrones definitivos sobre los cuales se va realizar la correlación. El modo de obtenerlo es sencillo, hay cuatro casillas para rellenar, las dos primeras indican el punto (x, y) mínimo sobre el que nacerá el rectángulo seleccionado. Por otro lado las dos casillas restantes sirven para indicar el tamaño del rectángulo $(anchura, altura)$. Cabe destacar que es importante fijarse en el tamaño de la imagen original para que la zona seleccionada siempre esté dentro de esas medidas, pues el programa no va admitir que se seleccione un rectángulo cuyos bordes se salgan de la imagen original [17].

El comando para la creación del rectángulo es el siguiente:

handles.h1rect=rectangle('Position',[xm ym anch alt],'LineWidth',2,'EdgeColor','w');

Donde $[xm \quad ym \quad anch \quad alt]$ son los datos de (x, y) y $(anchura, altura)$, además se indica el espesor de la línea y el color *'White'*.

En la parte derecha están los dos patrones definitivos, es decir, las imágenes sobre las cuales se va a realizar la correlación y que por lo tanto sólo van a incluir el patrón de partículas seleccionado cómo si fuera aplicar zoom sobre las zonas deseadas.

4.3.3. Número de ventanas

Posteriormente con todos los datos definidos, se selecciona el número de ventanas (Áreas de interrogación) que unidas dividirán los fotogramas, dentro se encuentran los conjuntos de partículas sobre las que se aplicará la correlación cruzada. Por lo tanto, dicho número de ventanas viene determinado por las características de estas agrupaciones, este comando se escribe en el recuadro de la Figura (4.22).

La teoría afirma que debe haber al menos entre 5 y 15 partículas por región de interrogación, aunque lo óptimo es de entre 20 y 30 partículas, de esta manera los picos de la función de correlación serán más

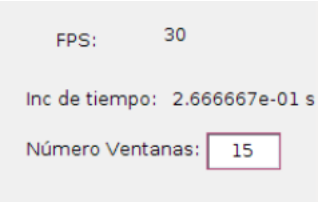


Figura 4.22: Selección del número de ventanas y los datos obtenidos de la interfaz anterior (FPS y Δt).

fáciles de detectar. Dichas partículas deberán tener un tamaño de entre 1.5 y 5 píxeles de la imagen. Todo esto implica que el número de ventanas en las que dividir la imagen patrón va de 30×30 a 100×100 píxeles. No obstante, todos estos parámetros teóricos son extraídos de un laboratorio dedicado exclusivamente al análisis PIV, por lo que de acuerdo a este proyecto la variación en la selección de los parámetros y la dificultad de imponerlos será amplia. De esta manera, las dimensiones de las ventanas en el análisis dependerá del vídeo grabado o descargado y que se está analizando, para que se ajusten lo máximo posible a las condiciones que se desean.

Hay que tener en cuenta no es posible efectuar la correlación para las partículas de forma individual, por lo tanto cuanto mas pequeñas sean las regiones de interrogación, es decir, que haya mayor cantidad de ventanas, habrá un mayor número de picos óptimos a la hora de obtener el desplazamiento.

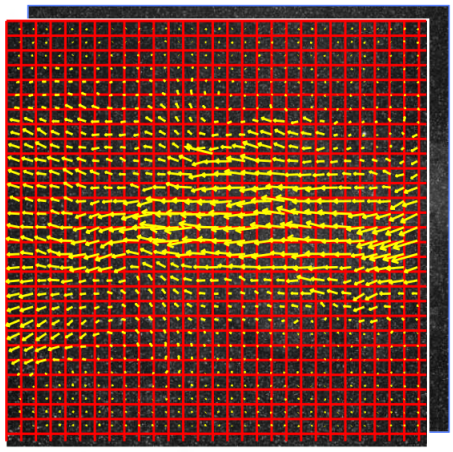


Figura 4.23: Imagen dividida en áreas de interrogación. Fuente [17].

Es necesario saber que hay que tener cuidado tanto si aumenta la cantidad de ventanas como si disminuye, pues no es bueno ni la falta ni el exceso. si el tamaño es demasiado grande cabe la posibilidad de que haya partículas que no sigan el mismo patrón y podrían tener velocidad y desplazamiento distintos, además de que la calidad y precisión de los resultados sería mucho menor. Por otro lado, si el tamaño es demasiado pequeño es posible que se pasen partículas de un volumen a otro lo que conllevará a un error en la correlación.

4.3.4. Siguiente interfaz 'Next_Callback'

Por último queda darle que botón que da acceso a la siguiente interfaz (flechita verde), sobre el cual va asignada una *function callback*, que define los parámetros de entrada sobre la función de correlación y sumándolos a los argumentos que va a extraer la función de correlación, se obtiene todas las variables de entrada que se necesitan para la tercera y última interfaz.

Este *Callback* tiene la función de obtener mediante la correlación cruzada una serie de variables necesarias para la obtención del campo de velocidades, ya sea de forma vectorial o escalar. De esta manera

incluye el algoritmo de correlación cruzada *corr.m* y la llamada de la tercera y última interfaz *Resultados.m*.

Primeramente se definen las variables *rect1 y rect2* mediante el comando *get(rectángulo,'Position')* que recoge la información de los parámetros de posición que tienen los rectángulos seleccionados. Dicha posición indica $[xmin \quad ymin \quad anchura \quad altura]$.

También se incluye mediante el comando *isequal(posición seleccionada, [0 0 1 1])* la condición de que si las propiedades de los valores de posición de ambos rectángulos considera y prueba que coincide con la forma *[0 0 1 1]* retorna 1 y por lo tanto la cumplen. Dichas condiciones de denominan como *rect1_act* y *rect2_act*. Esto significa que el software recibe si se ha seleccionado una zona para su estudio mediante el rectángulo o si simplemente realiza la correlación de la imagen completa.

Posteriormente se recogen las dos imágenes capturadas en la interfaz anterior mediante el comando *getimage(axes de la imagen)*, siendo *patron* la primera captura y donde se busca el patrón e *imagen* la segunda captura y sobre la buscar el mismo patrón. Una vez identificadas, el comando *size* indica el tamaño en píxeles de cada una de ellas de la forma *[ancho alto]*. Además se recoge la información de los textos que indican los *FPS* y el *número de ventanas* mediante el comando *str2double*.

El siguiente parámetro definido sirve para el cambio de unidades de píxeles de milímetros *px2mm*. Es importante conocer la relación entre estas dos unidades de medida en función de la calidad y resolución del vídeo que se ha seleccionado para el análisis. El píxel es la unidad de medida de las imágenes digitales y dependiendo de la calidad de imagen habrá mayor o menos densidad de píxeles en una pulgada. Esto significa que cuanto mayor de el número de puntos (píxeles) por pulgada *'dpi'*, mayor será la resolución. Se sabe que una pulgada equivale a $25{,}4\ mm$ y que por lo tanto se puede aplicar una relación constante de estas medidas asumiendo que el número de puntos en una pulgada es de más o menos 25, por lo tanto *px2mm = 1*. Medida que se corresponde a los vídeos que normalmente se han grabado en el laboratorio. No obstante, es bastante probable que la mayoría de los vídeos descargados de la red sean de mayor calidad y que este parámetro varíe en consideración.

A continuación se declara de nuevo la posición de *rect1* y *rect2* pero redondeadas al número entero más cercano mediante el comando *floor* y también declara los *offset* del patrón y de la imagen, los cuales indican la diferencia en [x y] entre una posición de referencia y el patrón o la imagen respectivamente.

Una vez declaradas todas las variables que serán imprescindibles para el cálculo de la correlación, son aplicadas las condiciones *rect1_act* y *rect2_act*, las cuales si son cumplidas definen el tamaño y la posición real del patrón y la imagen en función del rectángulo seleccionado. El patrón y la imagen recogerán la posición de su respectivo *offset* en $[x \quad y]$ y su $[ancho \quad alto]$, siendo el nuevo tamaño de cada uno de ellos, el tamaño de los rectángulos seleccionados. Todos estos datos corresponderán con la estructura de la *'Position'* que poseen dichas ventanas seleccionadas.

$$rect1 = [x_1 \quad y_1 \quad ancho_1 \quad alto_1] \qquad rect2 = [x_2 \quad y_2 \quad ancho_2 \quad alto_2]$$

$$offset\_patron = [x_1 \quad y_1] \qquad offset\_imagen = [x_2 \quad y_2]$$

$$patron = patron(ancho\_1, alto\_1) \qquad imagen = imagen(ancho\_2, alto\_2)$$

Lo único que faltaría es aportar el tamaño de cada una de las *'ventanitas' (áreas de interrogación)* en las que se dividirá la región seleccionada, esto va a variar en función de si se ha seleccionado un rectángulo o si se analiza la imagen completa. Lógicamente al igual que las imágenes, estas divisiones tendrán estructura de [ancho alto], con lo cual dichas dimensiones se obtienen de la siguiente manera:

- Si se ha utilizado rectángulo el tamaño de cada división será:

  ancho = ancho del rectángulo seleccionado / número de ventanas.

  alto = alto del rectángulo seleccionado / número de ventanas.

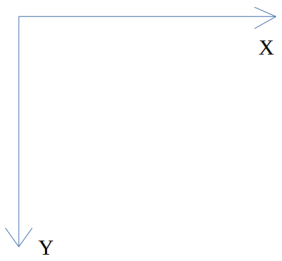


Figura 4.24: Sistema de referencia en la correlación.

- Si se analiza la imagen patrón completa el tamaño de cada división será:

  ancho = ancho de la imagen patrón / número de ventanas.

  alto = alto de la imagen patrón / número de ventanas.

Por último utilizando todas las variables declaradas se activa el algoritmo de correlación *corr.m*, incluido dentro del *Callback* y se llama a la tercera y última interfaz *Resultados.m*.

Algoritmo correlación *'coor.m'*

Este algoritmo se encarga de localizar las ventanas seleccionadas sobre los fotogramas de la imagen, para ello una vez establecido el número de ventanas en las que dividir la región se establece cada una en una ubicación que viene dada por coordenadas $[x \quad y]$, dicha posición se localiza en su punto (píxel) central [18].

Para conseguir todo esto se crea una función que viene definida por:

function [U, V, X, Y] = corr(patron,imagen,tam_ventana,offset_patron,offset_imagen)

Donde las salidas son:

U es la componenete del vector desplazamiento horizontal, *V* es la componente del vector desplazamiento vertical, *X* es la posición central de cada ventana sobre el patrón en el eje x, *Y* es la posición central de cada ventana sobre el patrón en el eje y.

Y las entradas son:
Patrón es la imagen sobre la que se hará la división de ventanas, *imagen* es la imagen sobre la que se va a buscar el patrón de partículas, *tam_ventana* es el vector [ancho alto] que indica el tamaño de cada ventana que va a dividir el patrón, *offset_patron* es la posición sobre la cual empieza la ventana seleccionada en [x y] sobre el patrón, *imagen* es la posición sobre la cual empieza la ventana seleccionada en $[x \quad y]$ sobre la imagen.

Cabe destacar que las posiciones de las ventanas, se redondean a la baja con el comando *floor*. También que el sistema de referencia es el de la Figura (4.24).

En primer lugar se declaran los tamaños en vector $[ancho \quad alto]$ de cada ventana y también del patrón seleccionado, estos datos vienen ya dados anteriormente y se asignan a la entrada de la función.

Posteriormente se asigna el número de ventanas que van a haber en el eje Y y en el eje X, *numY* y *numX*. Para ello se necesita comprender el comando que obtiene las dimensiones del patrón, este es el *size*, el cuál devuelve unas dimensiones de la forma número de renglones (eje Y) $\times$ número de columnas (eje X) $\times$ número de planos. Por lo tanto el número de ventanas en el eje Y '*numY*' es el número de columnas del patrón dividido por el ancho de cada ventana, y a su vez el número de ventanas en el eje X '*numX*' es el

número de renglones del patrón dividido por el alto de cada ventana.

A continuación se declaran las matrices U y V cuyos tamaños corresponden con el número de ventanas en cada eje y también M y N que recogen en una línea el número de ventanas en Y y en X respectivamente, para posteriormente realizar el mallado.

$$U = (numX, numY) \qquad V = (numX, numY)$$
$$M = (numY, 1) \qquad N = (numX, 1)$$

Una vez declaradas las variables, comienza el algoritmo, siendo (i, j) las variables que recorren en un eje y en otro la longitud del número de ventanas. Por lo tanto se declara un vector llamado *ventana* en coordenadas (x, y) de cada 'ventanita' en la que dividimos el patrón. Con lo cual mediante una iteración se le atribuye a cada ventana una ubicación característica dentro del patrón de la siguiente forma:

$$patron(alto * (j-1) \quad , \quad ancho * (i-1))$$

Para que recorra cada uno de los renglones y columnas en las que se divide el patrón se suma el vector que indica la longitud de dichos parámetros en función de las variables (i, j). Con lo cual el vector ventana se define de la siguiente manera:

$$ventana = patron((alto * (j-1) + 1 : alto * j \quad , \quad ancho * (i-1) + 1 : ancho * i)$$

Establecidas las posiciones de cada ventana en el patrón, se realiza la búsqueda de cada una de ellas en la imagen que representa el segundo fotograma, una por una mediante la iteración.

Todo esto se realiza mediante la función de correlación normalizada en 2D, que se encuentra en el *toolbox* de MATLAB, dicho comando se conoce como *normxcorr2*.

Esta rutina interna de Matlab realiza una correlación cuzada bidimensional normalizada utilizando la siguiente sintaxis:

$cc = normxcorr2(ventana, imagen)$.

Donde *'ventana'* es la matriz que representa cada una de las divisones en que se divide el patrón seleccionado (la plantilla dividida en áreas de interrogación). *'Imagen'* es la captura sobre la que se buscan los patrones, es decir, se realiza la correlación. Por lo tanto, se leen las imágenes en el espacio de trabajo y se muestrean lado a lado. De esta manera la matriz *'imagen'* debe ser mayor que la matriz *'ventana'*.

La matriz resultante $[cc]$, contiene valores del coeficiente de correlación, que pueden variar de -1 a 1, siendo 1 la correlación ideal.

Este algoritmo utiliza la fórmula (4.2).

$$\gamma(u,v) = \frac{\sum_{x,y}[f(x,y) - \overline{f}_{u,v}][t(x-u, y-v) - \overline{t}]}{\sum_{x,y}[f(x,y) - \overline{f}_{u,v}]^2 \sum_{x,y}[t(x-u, y-v) - \overline{t}]^2{}^{0,5}} \tag{4.2}$$

Donde $f(x, y)$ es la matriz imagen, $\overline{t}$ es la media de la matriz $ventana$ y $\overline{f}_{u,v}$ es la media de la matriz imagen en la región que está bajo la matriz ventana (la plantilla).

Si se sigue el funcionamiento establecido por la correlación cruzada:

- Calcula la correlación cruzada en el dominio espacial o frecuencial, dependiendo del tamaño de las imágenes.
- Calcula los sumatorios mediante iteración.

- Utilizan los sumatorios para normalizar la correlación cruzada y obtener los coeficientes de correlación.

El resultado $\gamma(u, v)$ es una función de dos variables, puesto que si se quiere localizar cada ventana en la imagen, el resultado será para cada una de las posibles posiciones de la ventana de forma (u, v). Los sumatorios que representa $f(x, y)$ recorre el fotograma en busca de los colores (las partículas), mientras que el sumatorio $t(x, y)$ recorre el fotograma buscando cada uno de los valores de la matriz ventana.

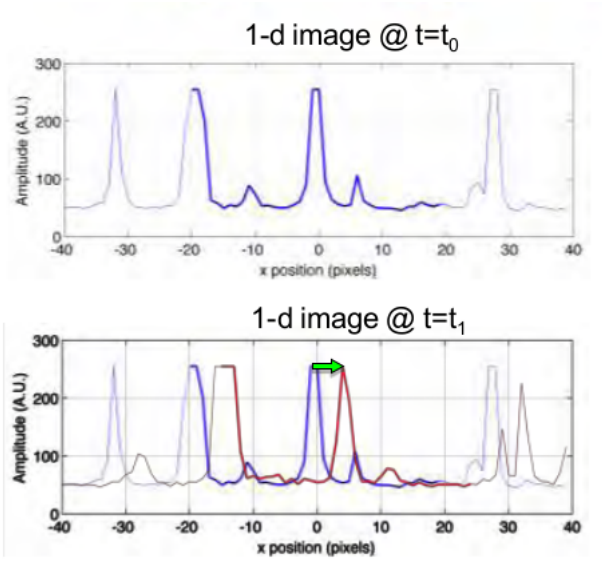
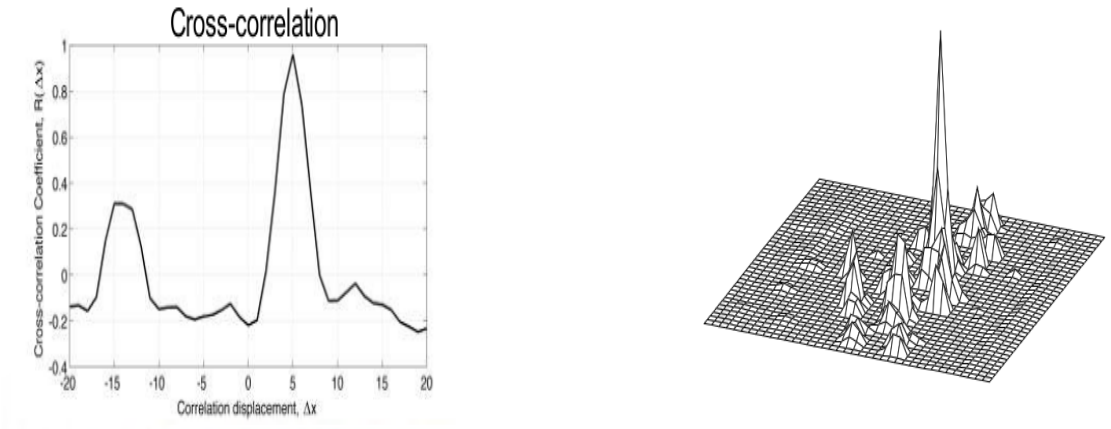


Figura 4.25: Seguimiento de la coincidencia del patrón para la extracción de picos.

Como es una función de distribución en dos dimensiones se obtiene la superficie de correlación en la cual se obtiene y se localiza en su posición el máximo, ver Figura (4.26). Dicho máximo indica el lugar y las coordenadas donde es más probable que se encuentre la ventana del patrón que se quiere encontrar en la segunda imagen (capturada un intervalo de tiempo después), ver Figura (4.25). De esta manera se repite el proceso para cada una de las ventanas (áreas de interrogación).



(a) (b)

Figura 4.26: Extracción del valor máximo (pico) en el plano de correlación

Por lo tanto si se identifican los máximos, $[y_{peak}, x_{peak}]$ y se pasa de valor lineal a múltiple mediante *ind2sub*, se pueden definir los vectores desplazamiento $[U(j,i), V(j,i)]$.

Siendo $U(j,i)$ la diferencia entre el máximo en la posición x y la anchura de cada ventana, sumando el incremento entre de las posiciones de referencia entre la imagen y el patrón, incremento de la componente x de los offset.

A su vez $V(j,i)$ es la diferencia entre el máximo en la posición y y la altura de cada ventana, sumando el incremento entre de las posiciones de referencia entre la imagen y el patrón, incremento de la componente y de los offset.

```
for i=1:numY
    for j=1:numX
        cont = cont + 1;

        ventana = patron(alto*(j-1)+1:alto*j,ancho*(i-1)+1:ancho*i);
        if ~prod(double(ventana(:)==ventana(1,1))) % No son todos iguales
            fprintf('%d %%\n',round(cont/(numX*numY)*100));

            cc = normxcorr2(ventana,imagen);
            [~, imax] = max(abs(cc(:)));
            [ypeak, xpeak] = ind2sub(size(cc),imax(1));

            U(j,i) = xpeak - ancho*i + offset_imagen(1) - offset_patron(1);
            V(j,i) = ypeak - alto*j  + offset_imagen(2) - offset_patron(2);
        end
        M(i) = floor(ancho*(i-1)+ (ancho+1)/2);
        N(j) = floor(alto*(j-1) + (alto+1) /2);
    end
```

Figura 4.27: Algoritmo de correlación. Fuente [Autor].

Si se define $M(i)$ y $N(j)$ como la posición central en (x,y) de cada ventana respectivamente, ver Figura (4.27), entonces el mallado que extrae las posiciones centrales de cada ventana sobre el patrón en (x,y) se define como:

$$[X,Y] = meshgrid(M,N).$$

Determinando completamente las líneas de cuadrícula verticales y horizontales que marcan las regiones de interrogación.

Obteniendo de esta manera las cuatro variables de salida del algoritmo $[U,V,X,Y]$. Volviendo al *next_callback*, gracias a este algoritmo se puede llamar a la tercera y última interfaz.

Esta última interfaz es llamada con el comando:

Resultados(patron,X,Y,U,V,ancho,alto,cte);

4.3.5. Tercera interfaz

En esta interfaz, se va a realizar un análisis de resultados obtenidos de la correlación mediante distintas opciones. Ya sea mediante un campo vectorial o un campo escalar.

En la Figura (4.28), se observa esta tercera y última interfaz, distinguiendo todos los botones y casillas que la forman.

Las variables de entrada de esta interfaz se introducen en la función inicial:

function Resultados_OpeningFcn(hObject, eventdata, handles, varargin)

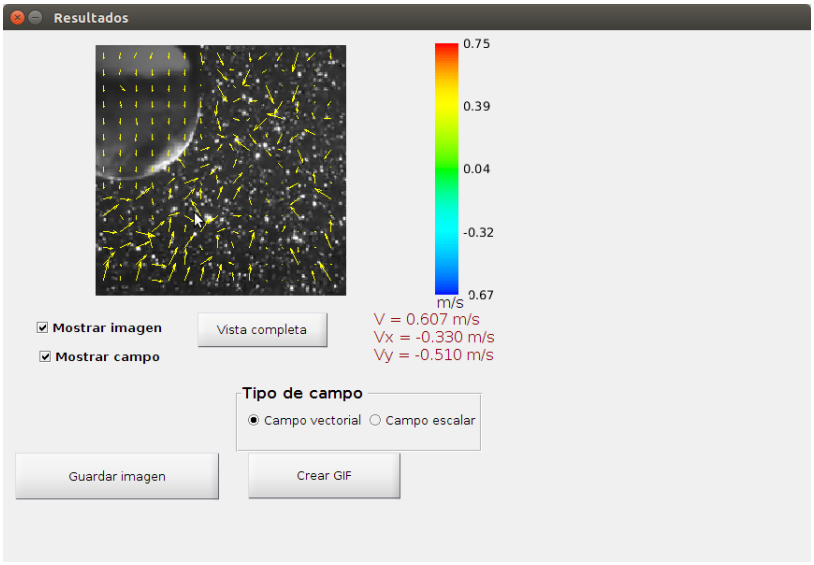


Figura 4.28: Tercera interfaz 'Resultados.m'. Fuente [Autor].

- *varargin {1} :* La primera imagen capturada.
- *varargin {2} :* La componente horizontal de la posición central de cada *ventanita* (área de interrogación).
- *varargin {3} :* La componente vertical de la posición central de cada *ventanita* (área de interrogación).
- *varargin {4} :* El vector desplazamiento (bidimensional) en el eje horizontal.
- *varargin {5} :* El vector desplazamiento (bidimensional) en el eje vertical.
- *varargin {6} :* La medida en píxeles del $ancho$ de la ventana seleccionada para analizar.
- *varargin {7} :* La medida en píxeles del $alto$ de la ventana seleccionada para analizar.
- *varargin {8} :* El factor de escala.

Como se ha indicado, *'cte'* es una constante que expresa el factor de escala, dato que se utilizará para el cálculo de la velocidad. Este factor da la relación entre los píxeles que tiene la imagen escogida y las unidades de medida de longitud. De esta manera únicamente es necesario calcular o determinar el desplazamiento de las partículas inmersas en el fluido en las direcciones x e y, para obtener los valores de velocidad (en unidades del en esas direcciones $[Vx \quad Vy]$.

Este coeficiente sigue la fórmula (4.3).

$$cte\ (factor\ de\ escala) = \frac{(px2mm \quad FPS)}{(1000 \quad (Fot2 - Fot1))} \tag{4.3}$$

Donde $px2mm$ es la relación entre píxeles y milímetros de la imagen, FPS son los fotogramas por segundo que tiene el vídeo y $\Delta Fotogramas$ es el incremento entre las dos capturas seleccionadas.

Figura 4.29: Panel selección del tipo de campo (Vectorial o Escalar.

Esta interfaz se inicia con la ventana de resultados, en la cual se va a poder ver el campo escalar y el campo vectorial, seleccionándolo desde el recuadro de la figura (4.29).

En primer lugar se definen las velocidades, cabe destacar que el tamaño de la matriz de velocidades, tiene que coincidir con las dimensiones que presente el patrón seleccionado. Por lo tanto mediante iteración, se irá recorriendo cada ventana obteniendo de esta manera las matrices de velocidad en el eje x y en el eje y, siendo U_tot y V_tot respectivamente.

Para calcular la velocidad mediante la técnica PIV se utiliza la ecuación (4.4).

$$Velocidad = Cte \times Desplazamientodelfluido \tag{4.4}$$

La variable temporal se incluye con la relación $\frac{FPS}{\Delta Fot}$, que ya viene incluida en el factor de escala *'Cte'*. De esta manera ya se puede calcular U_tot y V_tot. El desplazamiento en la dirección (x, y) viene determinado por las variables de entrada $[U(j,i) \quad V(j,i)]$. Por lo tanto las matrices de velocidad en los ejes x e y vienen dadas por la siguiente ecuación:

```
U_tot(alto*(j-1)+1:alto*j,ancho*(i-1)+1:ancho*i) = handles.datos.U(j,i)*ones(alto,ancho)*cte;
V_tot(alto*(j-1)+1:alto*j,ancho*(i-1)+1:ancho*i) = handles.datos.V(j,i)*ones(alto,ancho)*cte;
```

Por consiguiente, una vez obtenidas las dos componentes, se puede calcular el módulo de la velocidad.

$$Mag = \sqrt{U_{tot}^2 + V_{tot}^2};$$

Estas velocidades se podrán ver en el recuadro de la Figura (4.30).

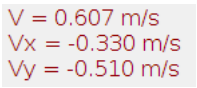


Figura 4.30: Recuadro que indica la velocidad en m/s de cada zona de la imagen seleccionada con el puntero.

Una vez obtenidas las velocidades reales, se de define la posición central de los ejes, además de la altura y anchura máxima para cuadrar la imagen patrón seleccionada de forma adecuada en el axes donde se observan los resultados.

Mediante *get(axes,'Position')*, recoge la posición total de los ejes.

Siendo la tercera y la cuarta componente la anchura y altura máxima. Por lo tanto la posición central será:

```
pos = get(handles.ejesPatron,'Position');
ancho_max = pos(3);
alto_max = pos(4);
pos_central = [pos(1)+pos(3)/2, pos(2)+pos(4)/2];
```

Para ajustar las proporciones de lo que se ha seleccionado con los ejes definidos, se realiza un estudio en función de la proporción, siendo:

proporción_gráfica=ancho max/alto max;

proporción_patrón=tam_patron(2)/tam_patron(1);

Por lo tanto, si se da la condición de que la proporción del patrón seleccionado es mayor que la de los ejes de la gráfica, significa que domina el ancho de la ventana patrón y se tendrá que bajar el alto para ajustar.

pos(4)=ancho_max/prop_patron;

pos(2)=pos_central(2)-pos(4)/2;

De esta manera la segunda y la cuarta componente de la variable posición ya se encuentran ajustadas.

Por otro lado, si la proporción de los ejes de la gráfica es mayor que la del patrón (caso contrario al anterior), significa que domina el alto de la ventana patrón y se baja el ancho para ajustarlo.

pos(3)=alto_max $\times$ *prop_patron;*

pos(1)=pos_central(1)-pos(3)/2;

Una vez ajustadas la primera y la tercera componente, se muestra la imagen del patrón que se ha seleccionado ajustada a los ejes para poder ver perfectamente el desplazamiento (vectores) o la magnitud del campo de velocidad escalar sobre dicha captura.

Por lo tanto queda definir de dónde se obtienen los campos escalares y los campos vectoriales.

4.3.6. Campo vectorial

Para poder representar el campo de vectores, se emplea el comando *quiver*, que muestra los vectores de velocidad como flechas con componentes (U, V) en los puntos (X, Y), es decir, si el primer vector está definido por las componentes $U(1)$ y $V(1)$, se muestra en el punto $X(1)$ e $Y(1)$. Es importante destacar que las matrices [X], [Y], [U], [V] deben ser del mismo tamaño. En este caso en principio [X] e [Y] eran vectores, los cuáles se expanden mediante el comando *meshgrid* ya mencionado.

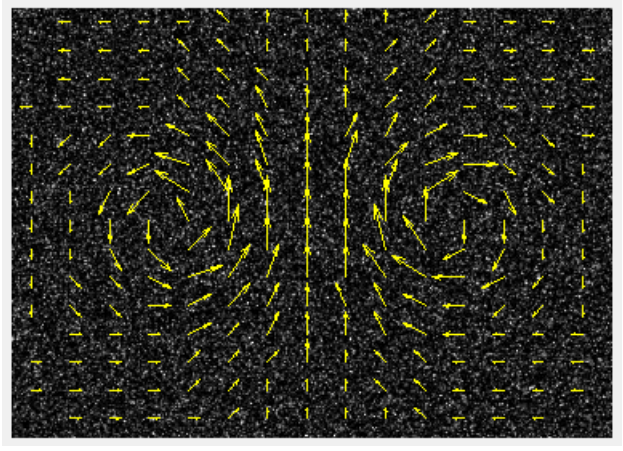


Figura 4.31: Campo de vectores sobre la imagen, siguiendo el mallado de las áreas de interrogación. Fuente [Autor].

Por lo tanto, en este caso el vector [X] corresponde a las columnas de [U] y [V], y el vector [Y] corresponde a las filas de [U] y [V]. De esta manera se le asigna el *Axes* donde se representa el patrón, viendo los vectores desplazamiento sobre la imagen, Figura (4.31).

4.3.7. Función escala de colores

El campo escalar representa la distribución espacial de la velocidad, asociando un valor a cada punto del espacio, dicha distribución viene asignada por distintas tonalidades de colores.

Por lo tanto, se define la función *escala_colores.m*, donde la entrada $'t'$ es una matriz que viene dada por $[fil, col]$ y la salida RGB, representa tres vectores definidos por unas relaciones, correspondiendo con los colores 'Red', 'Green' y 'Blue'.

Esta función crea colores correspondientes a las intensidades que se dan entre, 0 (mínimo) y 1 (máximo), de los colores primarios rojo, verde y azul.

Se definen las matrices [R], [G] y [B] del mismo tamaño que $'t'$ y se le asigna el valor de $t(i,j)$ a la variable 'l'. Entonces se definen los coeficientes las posiciones de 0 a 1 de los colores cian y amarillo y el exponente *n* para determinar mediante relaciones los vectores R, G y B.

n (exponente) = 1.
rC (posición del cian [0 0.5]) = 0.25.
rA (posición del amarillo [0.5 1]) = 0.75.

Por lo tanto se establecen los valores de cada vector en función de unas condiciones sobre la posición de 'l':

$Si \quad l < rC \quad y \quad l \geq 0$

$$R(i,j) = 0;$$
$$G(i,j) = (\frac{1}{rC})^{1/n};$$
$$B(i,j) = 1;$$

$Si \quad l < 0{,}5$

$$R(i,j) = 0;$$
$$G(i,j) = 1;$$
$$B(i,j) = 1 - (\frac{(1-rC)}{(0{,}5-rC)})^{n};$$

$Si \quad l < rA$

$$R(i,j) = (\frac{(1-0{,}5)}{(rA-0{,}5)})^{(}1/n);$$
$$G(i,j) = 1;$$
$$B(i,j) = 0;$$

$Si \quad l \leq 1$

$$R(i,j) = 1;$$
$$G(i,j) = 1 - (\frac{(1 - rA)}{(1 - rA)})^{n};$$
$$B(i,j) = 0;$$

De esta manera se acaban definiendo las salidas RGB.

RGB(: , : , 1) = R;
RGB(: , : , 2) = G;
RGB(: , : , 3) = B;

4.3.8. Campo escalar

Una vez definida la función para la escala de colores, se define la búsqueda de los valores máximos y mínimos del campo de velocidades. De esta manera utilizando los valores de la velocidad en el eje X y en el eje Y, además del módulo, se emplea la función *escala_colores* para relacionar cada velocidad con un nivel de color.

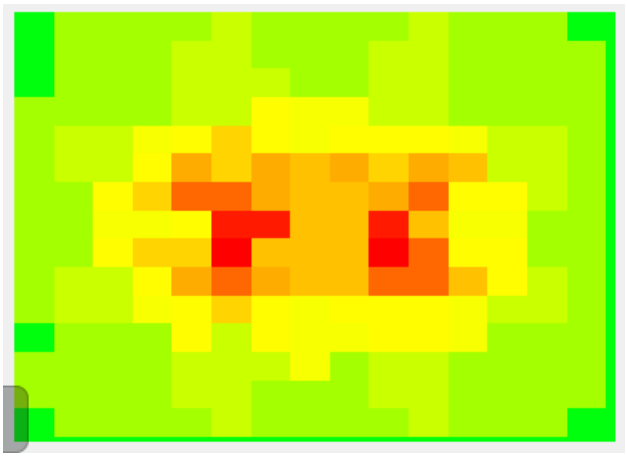


Figura 4.32: Campo escalar, siguiendo el mallado de las áreas de interrogación. Fuente [Autor].

Se puede seleccionar la magnitud de la velocidad en el eje X e Y también, pues antes ya se ha calculado.

V color = escala_colores(Mag - mínimo) / (maximo - minimo);
Vx color = escala_colores(U_tot - mínimo) / (maximo - minimo);
Vy color = escala_colores(V_tot - mínimo) / (maximo - minimo);

Después ver asignados los valores de velocidad en la escala de colores, se define dicha escala mediante cinco marcas que van del mínimo al máximo de los valores de velocidad como si fuera de 0 a 1:

Marcas 1 = mínimo, (0);
Marcas 2 = mínimo + (máximo - mínimo)/4, (0.25);
Marcas 3 = (máximo + mínimo)/2, (0.5);
Marcas 4 = máximo - (máximo - mínimo)/4, (0.75);
Marcas 5 = máximo, (1);

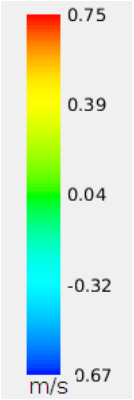


Figura 4.33: Ejes referencia, relacionando la tonalidad con la velocidad.

Todo esto está asignado a los ejes de la escala de colores, representando cada color que se ve en la ventana de resultados con el valor de la velocidad en m/s, ver Figura (4.33).

Finalmente existe la opción de guardar la imagen de resultados, ya sea con el campo de vectores o con el campo escalar. Para posteriormente mediante la sucesión de varias imágenes, representar en la misma ventana una imagen GIF en movimiento.

Capítulo 5

Instalación PIV elaborado en el laboratorio

Gracias a la información definida en los capítulos 2 y 3, se ha creado en el laboratorio de Física aplicada de la ETSID una instalación que simula un sistema PIV. De esta manera se consigue uno de los objetivos principales de este proyecto, que es el entendimiento y la experimentación a nivel docente de la técnica PIV.

Esta instalación fue creada en gran parte por el alumno Antonio José Pérez Vidal [bibliografía], del grado en ingeniería mecánica, durante la realización de su trabajo de Fin de Grado. Así pues, en este proyecto se pretende conseguir la grabación de un vídeo con el que poder ver los resultados que devuelve el algoritmo de correlación.

Como ya se ha introducido, PIV utiliza diferentes aparatos y sistemas en la realización de sus experimentos. En este caso, se han utilizado elementos que se encontraban en el laboratorio, por tanto han sido fáciles de obtener y son bastante baratos en comparación a una instalación de laboratorio profesional dedicada exclusivamente a estos experimentos. Aunque la calidad de los vídeos (uniformidad de partículas, calidad de imagen...) no sea la mejor, los experimentos realizados en este proyecto sirven perfectamente para obtener unos buenos resultados, o al menos contrastar entre distintos vídeos resultados buenos y resultados malos, sirviendo esto para dar a conocer mejor las propiedades de esta técnica y su importancia.

De esta manera, conociendo los elementos que forman esta instalación y comparándolo con una instalación profesional, se podrá valorar y admirar la inversión en investigación que supone el estudio de calidad de flujo de fluidos mediante este tipo de técnicas.

5.1. Componentes de la instalación.

Esta instalación está compuesta por los siguientes elementos que se van a describir en esta sección:

(a) Cajón que contiene el fluido.

(b) Fluido (agua destilada)

(c) Partículas.

(d) Bomba hidráulica.

(e) Láser.

(f) Configuración de lentes.

(g) Cámara de alta velocidad.

5.1.1. Cajón que contiene el fluido.

Consiste en un prisma de paredes resistentes dentro del cual estará inmerso el el fluido dopado con partículas. Este cajón está hecho con metacrilato y más o menos tiene una longitud de 1,5 metros, ver Figura (5.1).

Esta longitud es perfecta para obtener un flujo laminar, pues tiene las medidas suficientes para compensar la turbulencia inicial que produce la bomba, para posteriormente con esa potencia empujar el flujo uniformemente. Para incluir las mangueras de la bomba (por las cuales recirculará el agua), se han hecho dos agujeros para mantener las boquillas de cada manguera dentro del prisma y por tanto toda el agua que circule por ellas caerá dentro del cajón.

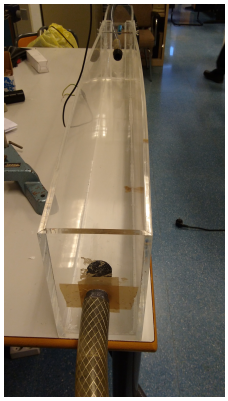


Figura 5.1: Cajón de metacrilato que contiene el fluido. Fuente [Autor].

5.1.2. Fluido (agua destilada).

Se utiliza este fluido destilado para que el flujo de las partículas esté sumergido y distribuido uniformemente dentro del mismo. Esto se consigue gracias a que el agua destilada esta libre de impurezas e ionización, por tanto las partículas no reaccionarán ni se irán al fondo o a la superficie, consiguiendo que se queden por cualquier lado del fluido para que se puede establecer una relación fiable entre el movimiento del fluido y el movimiento de las partículas.

Figura 5.2: Botella de agua destilada utilizada. Fuente [Autor].

A pesar de esto siempre hay problemas con algunas partículas que se van a la superficie, interponiéndose entre el plano láser y el centro del fluido, de manera que no se observará bien la dispersión de luz de las partículas.

5.1.3. Partículas.

Como ya se ha mencionado en el proyecto, estas partículas tienen que seguir el flujo de fluido perfectamente, además se encargan de dispersar (reflejar) la luz que el láser incide sobre ellas y esto lo recibe la cámara de alta velocidad.

Las propiedades que tienen las partículas utilizadas en esta instalación docente, viene determinada por la etiqueta del bote, ver Figura (5.3).

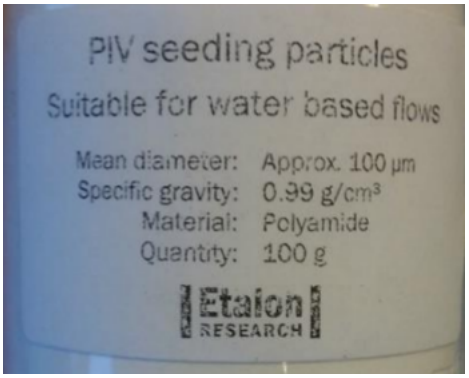


Figura 5.3: Bote de las partículas trazadoras utilizadas. Fuente [Autor].

Cabe destacar que la densidad del agua destilada es de $1g/cm^3$, en las propiedades de las partículas se indica que la densidad de las partículas es de $0{,}99g/cm^3$, con lo cual son muy similares. Esto justifica la distribución uniforme que tendrán estas partículas cuando estén inmersas en el fluido, de esta manera seguirá al fluido perfectamente y no se irán a los extremos del cajón.

5.1.4. Bomba hidráulica.

Para conseguir un flujo se emplea una bomba de agua EHEIM, ver Figura (5.4), esta marca fabrica bombas de agua para acuarios. Por tanto, su potencia será más que suficiente para lograr el flujo laminar deseado.

(a) (b)

Figura 5.4: Bomba de agua utilizada en la instalación. Fuente [Autor].

El funcionamiento de este tipo de bombas es sencillo, únicamente se puede pulsar el botón de encendido y mover la aguja que regula la potencia. Para conseguir el flujo laminar en el tamaño del cajón, es necesario que esté a máxima potencia. Por otro lado, para garantizar la circulación de corriente, posee dos mangueras que nacen de la propia bomba, cada una tiene su extremo que le servirá para cumplir su función.

La primera manguera es la propulsora (el agua que circula por ella se expulsa al cajón), por tanto se coloca al inicio del cajón, siendo esta zona donde existe más turbulencia. Este tubo marcará el fuerte inicio de la corriente, dando impulso para que llegue al otro extremo del cajón.

La segunda manguera es la que absorbe el flujo de corriente que le llega, de esta manera se garantiza que haya recirculación de flujo. La fuerza de absorción va a ser que siempre menor que la de impulso, no obstante, se aprovecha la longitud del cajón de manera que al final llegue con poca fuerza y simplemente con la fuerza de absorción se mantenga un flujo laminar. Esta última zona (siendo el último tercio del cajón), es la que se va a utilizar para hacer las grabaciones y donde se va a introducir la lámina láser.

Figura 5.5: Filtro recogedor de partículas. Fuente [Autor].

Entre la bomba y el extremo de la manguera de absorción existe un circuito extra que actúa de filtro de impurezas y partículas sobrantes. Este circuito alternativo se puede cerrar y abrir cuando se desee para filtrar las partículas, ver Figura (5.5).

5.1.5. Láser.

Un láser es un haz coherente y enfocado de fotones. Coherente, en este contexto, significa que es todo en una misma longitud de onda, a diferencia de la luz ordinaria, que existe en muchas longitudes de onda.

El acrónimo 'LÁSER' significa *'amplificación de luz por emisión estimulada de radiación'*. Los láseres funcionan como resultado de efectos de resonancia. El haz saliente de un láser es un campo electromagnético coherente. En un haz coherente de energía electromagnética, todas las ondas tienen la misma frecuencia y fase.

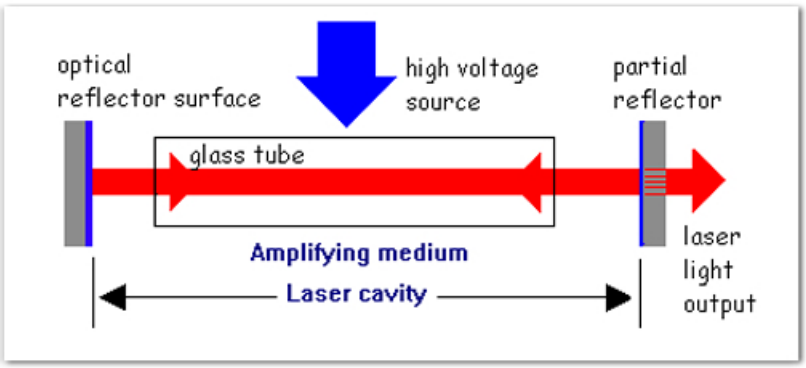


Figura 5.6: Funcionamiento de un láser. Fuente [*Wikipedia.com*].

En cualquier láser básico, una cámara llamada cavidad está diseñada para reflejar internamente las ondas infrarrojas (IR), de luz visible o ultravioleta (UV) para que se superpongan entre sí. La cavidad puede contener sustancias gaseosas, líquidas o sólidas. La elección del material de la cavidad determina la longitud de onda de la salida. En cada extremo de la cavidad, hay un espejo. que totalmente reflectante, sin permitir que pase la energía, el otro espejo que hay en el otro extremo es parcialmente reflectante, permitiendo que aproximadamente el 5 por ciento de la energía pase a través del mismo. La energía para provocar la reacción se introduce en la cavidad desde una fuente externa, esto se llama bombeo, ver Figura (5.6).

Como resultado del bombeo, aparece un campo electromagnético dentro de la cavidad láser a la frecuencia natural (resonante) de los átomos del material que rellena la cavidad. Las olas se reflejan hacia adelante y hacia atrás entre los espejos y la longitud de la cavidad es tal que los frentes de onda reflejados y se refuerzan mutuamente en fase a la frecuencia natural de la sustancia de la cavidad. Las ondas electromagnéticas a esta frecuencia resonante emergen desde el extremo de la cavidad que tiene el espejo parcialmente reflectante. La salida puede aparecer como un haz continuo, o como una serie de pulsos breves e intensos.

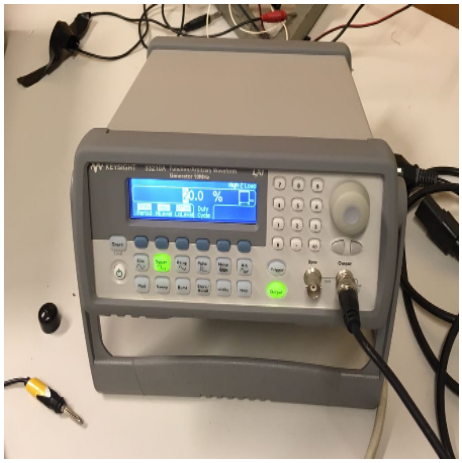
En este caso, se va a utilizar el láser **ACCULASE-PWM-650-5-S** (650 nm y 5 mW) con modulador y control PWM de patrón de puntos +4.75 a +5.25 V , ver Figura (5.7) que posee las siguientes características:

Figura 5.7: Láser empleado en la instalación. Fuente [Autor].

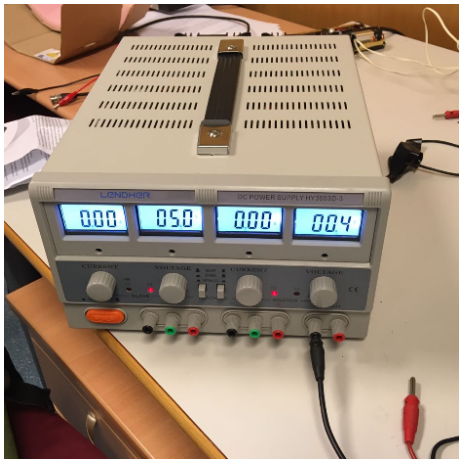
- Módulo láser LED rojo de alta eficiencia 635 o 650nm.
- Alta precisión apuntando <1mRad.
- La lente equipada proporciona un haz circular / elíptico.
- Posee circuito interno del conductor.
- 0 - 1V de tensión de control lineal de entrada (tipos LC).
- Entrada de 5V TTL PWM (tipos de PWM).
- Tapa aislada eléctricamente.
- Protección de polaridad inversa.

Para su funcionamiento y alimentación se han empleado un generador de onda de 10 mW y una fuente de alimentación de corriente continua. Ver Figura (5.8).

El láser está conectado por 4 cables (rojo, negro, amarillo y azul), de los cuales el rojo y el negro se conectarán a la fuente alimentación a 5V. Los otros dos se conectarán al generador de ondas a modo de control y seguridad, siendo el amarillo la toma de tierra, que se conectará al cable rojo.



(a) Generador de ondas.



(b) Generador de ondas.

Figura 5.8: Elementos para la puesta en funcionamiento del láser. Fuente [Autor].

5.1.6. Óptica (Configuración de lentes).

Para la obtención del plano láser, se realiza una construcción que relaciona el láser con una lente plana-convexa, variando la distancia entre estos elementos lo que se necesite para conseguir un plano láser coherente y lo suficientemente potente para iluminar la región deseada, ver Figura (5.9).

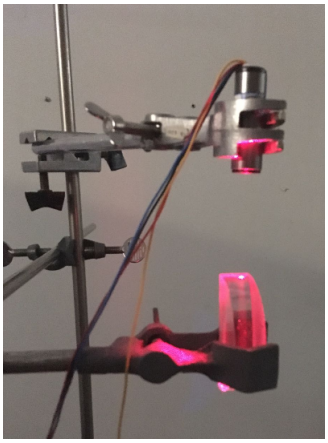


Figura 5.9: Configuración láser-lente utilizada en el laboratorio. Fuente [Autor].

Para conseguir el objetivo de un plano, se deben probar diferentes configuraciones o posiciones de la lente con el láser:

- Si se incide perpendicular y directamente en el centro de la zona plana de la lente, el haz de luz se concentrará en un punto del extremo y divergirá, el problema es que se expandirá bastante poco, siendo insuficiente para lograr un plano analizable, ver Figura (5.10).

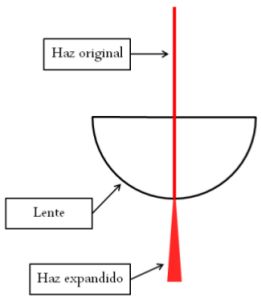


Figura 5.10: Láser incidiendo perpendicularmente en el centro de la lente. Fuente [Autor].

- Por otro lado, si se incide desde un poco más hacia el extremo, cuando llega a la superficie convexa, el haz diverge completamente y se expandirá bastante más (obteniendo un plano más amplio).
- Para conseguir un plano lo suficientemente amplio para el estudio, se incidirá desde una zona muy próxima al extremo de la superficie plana, de manera que al llegar a la segunda superficie se obtendrá un plano bastante más amplio y digno para poder grabar una zona del cajón, ver Figura (5.11).

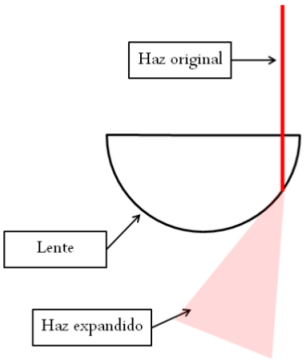


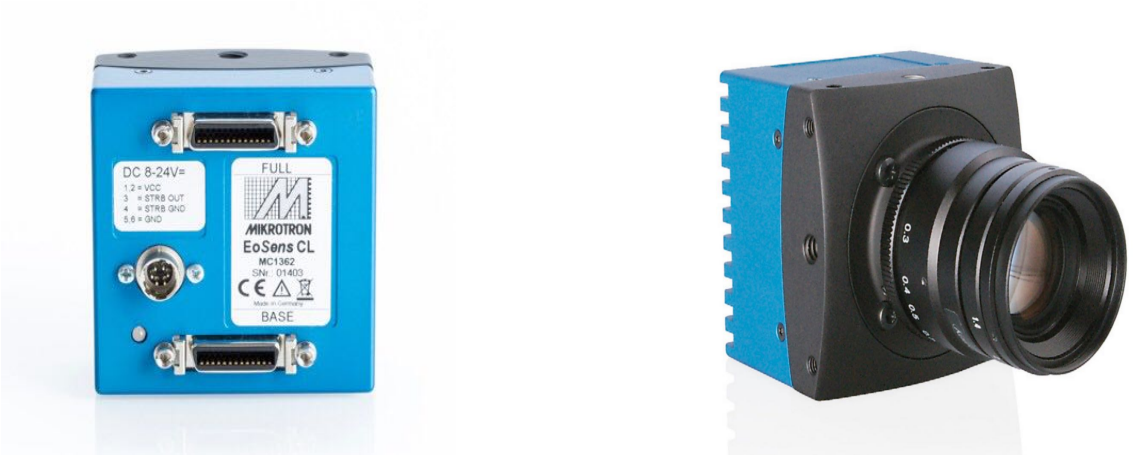
Figura 5.11: Láser incidiendo perpendicularmente desde el extremo de la lente. Fuente [Autor].

5.1.7. Cámara de alta velocidad.

Para la captura del vídeo que sigue el comportamiento de las partículas se utiliza una cámara de alta velocidad, permitiendo grabar una gran cantidad de fotogramas por segundo. Como se ha descrito en el capítulo 2, para captar este movimiento se utilizan sensores CCD o CMOS.

En este caso la cámara utiliza un sensor CMOS, que como se ha visto, permite que la digitalización de los píxeles se realice internamente en unos transistores que lleva cada celda de registro, por lo que todo el trabajo se lleva a cabo dentro del sensor y no se necesita un chip externo que se encargue de esta función. Con esto se reducen costes y se pueden emplear equipos más pequeños. Además, tiene la ventaja de que los sensores CMOS son más sensibles a la luz, por lo que en condiciones de baja dispersión de luz de las partículas, (como es el caso de esta instalación docente) se comportan mucho mejor.

Más en detalle, la cámara utilizada en concreto es una MIKROTRON EO SENS 1362, ver Figura (5.12), que posee estas características y especificaciones:



(a) (b)

Figura 5.12: Cámara MIKROTRON empleada en la instalación. Fuente [Autor].

(a) Es importante tener en cuenta la velocidad con la que se decide grabar el vídeo, pues se está empleando plano láser continuo, con lo cual la velocidad a la que se grabará podrá ser más reducida. Permitiendo distinguir las zonas del vídeo que se quieren estudiar y poder seleccionar los instantes deseados con bastante precisión, para ello se puede emplear un láser de menor potencia (al ser continuo). Resumiendo, se podría decir que no se necesita un láser de doble pulso y una cámara más precisa, pues con un vídeo normal se pueden seleccionar dos capturas consecutivas más que decentes para comprender los principios básicos de esta técnica, con lo cual grabar entre 20 y 150 FPS (fotogramas por segundo) será suficiente.

(b) Para la velocidad de grabación se ha de tener en cuenta otro factor, la apertura del obturador. Este parámetro se ha de controlar bastante bien, pues un elevado tiempo de obturación puede hacer que se graben elementos no deseados (polvo o luz externa), incrementando la dificultad para distinguir la diferencia entre lo que es partícula y lo que no. No obstante, si este tiempo es pequeño, lo que se graba puede no ser del todo preciso o completo, obteniendo vídeo con muy pocas partículas (sea el vídeo 1 del capítulo 6).

Si se unen todas las condiciones descritas hasta ahora, se podría establecer un criterio:

- Es necesario una velocidad de grabación pequeña, pero lo suficientemente grande para para seguir perfectamente el flujo.
- El tiempo de apertura de obturador no tiene que ser muy grande (sino dejaría entrar mucha luz y eso es perjudicial), pero lo suficiente para conseguir un patrón claro y con bastantes partículas.

Por último, es necesario mencionar la importancia de calibrar la cámara antes de cada grabación y colocándola muy cerca de la región del cajón donde se va a grabar. De manera se facilita el enfoque y la amplitud de grabación será suficiente.

5.2. Desarrollo del proceso.

En esta sección se va a describir, paso a paso, todo el procedimiento seguida hasta obtener algunos vídeos decentes y que merezca la pena su estudio en la interfaz implementada en el proyecto.

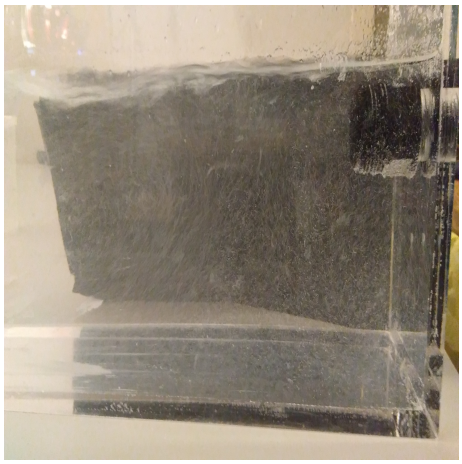
En primer lugar se trató de realizar una grabación con los restos de proyectos anteriores, el problema es que el agua llena de partículas estaba cargada de impurezas y otras sustancias que hacía que el agua no se viera transparente, por lo tanto no se distinguía lo que era partícula y lo que no. De esta manera se procedió

a limpiar todo el cajón de metracrilato.

Una vez realizada la limpieza colocaron las mangueras de la bomba de agua, cada una en su extremo del cajón, para ello se dispone de dos agujeros (uno en cada lado), donde mediante una rosca se deja fijo e impidiendo que el agua de dentro del cajón se salga por lo bordes.

Figura 5.13: Colocación de la bomba. Fuente [Autor].

Después, una vez selladas las salidas no deseadas de agua, se llena el cajón entero hasta sobrepasar las dos roscas de la manguera, facilitando el flujo a través de las tuberías y la bomba. Acto seguido se conecta la bomba a la corriente y el agua se empezará a mover, para favorecer la corriente se coloca la bomba por debajo del cajón para que la tubería esté alineada, ver Figura (5.13). Antes de esto se debe esperar un tiempo hasta que todo el circuito interno de la bomba esté lleno para poder hacer la recirculación.



(a) Boquilla propulsora.

(b) Boquilla de absorción.

Figura 5.14: Extremos de la bomba que recirculan el flujo en el interior del cajón. Fuente [Autor].

Cuando ya se tiene una corriente de agua uniforme, se comienza a introducir las partículas trazadoras, es recomendado echarlas poca a poco y cerca de la boquilla de absorción como se ve el Figura (5.14), de esta manera saldrán por el otro lado con potencia y se podrán repartir de una forma más uniforme y homogénea. Este proceso se realiza sucesivamente hasta que se observe una cantidad adecuada de partículas, ver Figura (5.15). Para ello se ha de tener en cuanto que para que el estudio sea más preciso y bueno, el flujo debe tener muchas partículas que marque el seguimiento del flujo, cabe destacar que hay que tener cuidado con las partículas que se quedan en la parte superior del flujo, pues puede impedir el paso del haz de láser al centro del fluido.

Una vez se tiene un flujo laminar una gran cantidad de partículas uniformemente repartidas, se introduce un tubo para conocer y estudiar como se comporta el fluido ante un obstáculo, permitiendo ver la desviación, aumento y descenso de la velocidad y la trayectoria que seguirá el flujo.

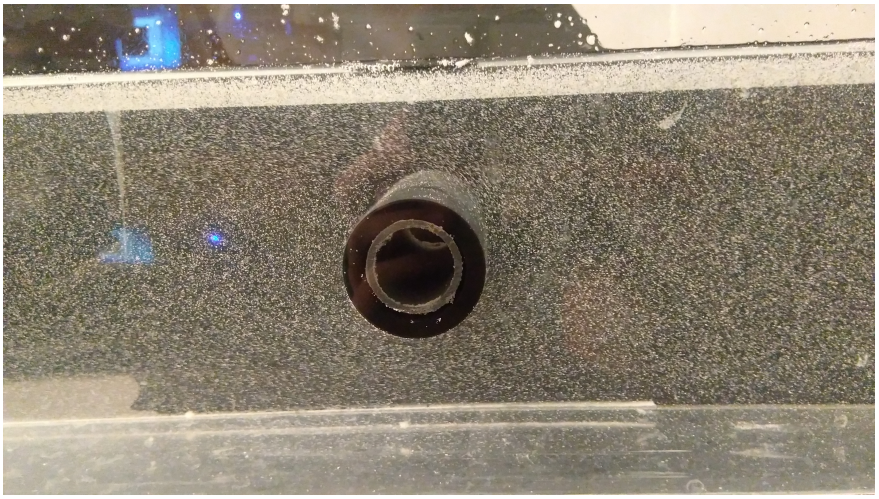


Figura 5.15: Fluido con una gran cantidad de partículas, lo que garantiza mejores resultados. Fuente [Autor].

Para realizar la grabación e introducir el haz láser, se han seguido los pasos realizados por Antonio [23]. En primer lugar se selecciona la parte trasera del cajón (donde se encuentra la boquilla de absorción) para colocar una manta negra que impida el paso de luz del exterior. Esto es debido a que en esta zona el flujo es laminar y se observa perfectamente el avance de partículas de manera uniforme, de esta manera se podrá realizar un estudio bastante coherente.

La cámara de alta velocidad se coloca justo delante de la región del cajón a estudiar, ver figura (5.16), de esta manera para impedir el paso de la luz, también se va a cubrir la cámara con la manta negra.

Para la obtención del plano láser, se monta una configuración totalmente perpendicular al cajón, dejando que el láser enfoque hacia abajo, donde se encontrará con la lente plana-convexa, formando así un plano de luz. Al ser un láser continuo se puede ir variando la distancia del láser con respecto a la lente, consiguiendo un plano lo suficientemente ancho para analizar la región que ilumine.

Para que pueda pasar el láser y a su vez no se deje entrar mucha luz exterior, se realiza un corte en la zona superior de la manta negra, es pues que moviendo la configuración de láser-lente hacia la zona del corte, se podrá dejar que el plano ilumine la zona y no se deje pasar mucha luz.

Por último, queda el paso de grabar el vídeo. Para poder realizarlo se emplea el programa *tracker*, donde se puede seleccionar el tiempo de obturación y los fotogramas por segundo que se desean. Una vez grabados, se exportan y se guardan, permitiendo analizarlos posteriormente con la interfaz.

Figura 5.16: Cámara colocada justo delante de la región a estudiar. Fuente [Autor].

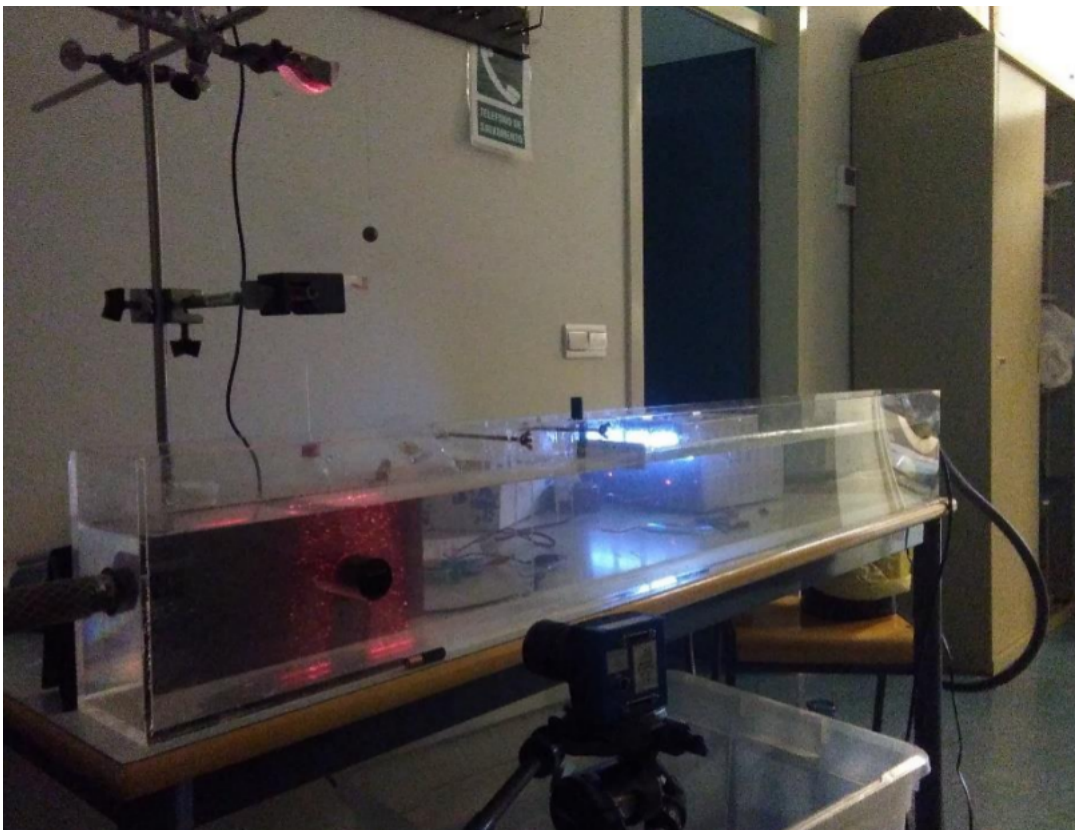


Figura 5.17: Visión general de toda la instalación PIV docente. Fuente [Autor].

Capítulo 6

Análisis e interpretación de los resultados

En esta parte se analizan todos los resultados que produce cualquier selección de dos capturas con un Δt en la interfaz descrita en capítulos anteriores. Es importante conocer que la interfaz devuelve resultados aproximados, con lo cual, en algunos experimentos devolverá resultados que no son coherentes, la finalidad de esta parte será dar a conocer la manera de obtener, con cualquier tipo de vídeo, unos resultados reales y concisos.

6.1. Importancia de la cantidad de partículas y el número de ventanas

Una vez descritas y desarrolladas todas las interfaces, es necesario centrarse en la última, pues es la que devuelve los campos de velocidad de cualquier experimento que se realice. Es importante comprobar que dimensiones tiene que tener la ventana seleccionada como patrón, para ello se pueden realizar distintos experimentos con diferentes tamaños de patrones escogidos.

Para comprobar que tamaño de ventana seleccionada es adecuada se va variando el número de ventanas seleccionado y se observan los resultados. Ya se ha introducido en el proyecto la importancia de la cantidad de ventanas seleccionadas, pues a mayor cantidad y por tanto menos tamaño existe una mayor probabilidad de que se superpongan partículas de una ventana con las de otra. Por otro lado, a menor cantidad de ventanas (más grandes), hay más probabilidad de que no sea fiable la uniformidad y dirección de las partículas seleccionadas.

La figuras de los resultados que se van a mostrar para los distintos casos muestran la velocidad absoluta, permitiendo deducir el comportamiento del flujo para cualquier dirección definido por las partículas. Para comprobar que número o rango de números es adecuado, hay que fijarse precisamente en la coherencia del comportamiento del flujo, es decir, que las velocidades se asemejen a la realidad de los vídeos seleccionados, que las dirección de las partículas sea la adecuada y conforme a lo que se ve en los vídeos.

Se han seleccionado dos vídeos bien distintos conseguidos en el laboratorio de Física de la ETSID, distinguiendo entre uno y otro la cantidad de partículas. De esta manera se ven diferentes resultados, representando las distintas precisiones que se pueden llegar a conseguir, ver Figura (6.1).

El tamaño de las regiones seleccionadas para cada vídeo se corresponde con:

- Vídeo 1, el tamaño de imagen recortada (región seleccionada) es de 250×250, permitiendo estudiar únicamente la zona donde hay más partículas (centro-arriba), no obstante, debido a esta escasez es difícil ver o seguir un patrón de partículas claro.

(a) Vídeo 1, 30 FPS, $\Delta t = 0{,}3sec.$

(b) Vídeo 2, 25 FPS, $\Delta t = 0{,}2sec.$

Figura 6.1: Captura (a), vídeo 1, con menos cantidad de partículas. Captura (b), vídeo 2, con mayor cantidad de partículas. Fuente [Autor].

- Vídeo 2, el tamaño es de la zona es de 900×600, permitiendo estudiar el comportamiento de las partículas (por tanto del flujo) que rodea al obstáculo. Este vídeo se ha obtenido sin iluminación láser, es decir, se ha grabado directamente sobre las partículas con un fondo negro aprovechando el contraste de colores. Así pues se le da una utilidad más al software desarrollado.

6.1.1. Selección de 5 ventanas.

Así pues, se realizan distintas pruebas, comenzando por escoger un reducido número de ventanas, **cinco** para ser exactos.

Análisis del vídeo 1.

En primer lugar, Figura (6.2), se muestra el vídeo con menos partículas.

Cabe destacar que todos los campos escalares tienen la siguiente magnitud:

- Rojo (Velocidad alta).
- Amarillo (velocidad media).
- Verde (Velocidad 0).
- Azul claro (velocidad media en sentido contrario).
- Azul oscuro (velocidad alta en sentido contrario).

Como era deducible, los resultados que se observan no son precisamente los más adecuados, ya que al verse tan poca cantidad de ventanas y por tanto un un contraste elevado en los colores de cada ventana, los resultados son complicados de interpretar y no sirven para mucho. Al margen de la cantidad de ventanas, el hecho de haber tan poca cantidad de partículas (3 o 4 por ventana) no permite distinguir correctamente el patrón, o al menos con la precisión de este algoritmo.

Es pues que la escasa cantidad de partículas unida a la escasa cantidad de áreas de interrogación, no son suficientes para determinar unos resultados coherentes y en los que se observe claramente el patrón, a pesar de que devuelva una velocidad de flujo real, en torno a $0{,}5\ m/s$. No obstante, este vídeo no muestra una distribución uniforme de las partículas, por tanto habrá zonas con análisis más reales que otras.

(a) Campo vectorial (b) Campo escalar

Figura 6.2: Resultados vídeo 1, 5 ventanas, con menos cantidad de partículas. Fuente [Autor].

Análisis del vídeo 2.

La Figura (6.3) muestra los resultados del vídeo con una mayor cantidad de partículas, dividiendo en 5 áreas de interrogación la zona seleccionada. A priori, en este vídeo sí que hay cantidad de partículas suficientes para ver el patrón a estudiar y en este caso en cada área de interrogación se ven una gran cantidad de partículas, entre 30 y 35.

(a) Campo vectorial (b) Campo escalar

Figura 6.3: Resultados vídeo 2, 5 ventanas, con mayor cantidad de partículas. Fuente [Autor].

De esta manera, se puede ver una interpretación más o menos real de los resultados, la velocidad en la parte del borde trasero del cilindro es mayor (campo escalar) y el campo de vectores sigue con bastante claridad el obstáculo (cilindro), ver figura (6.3).

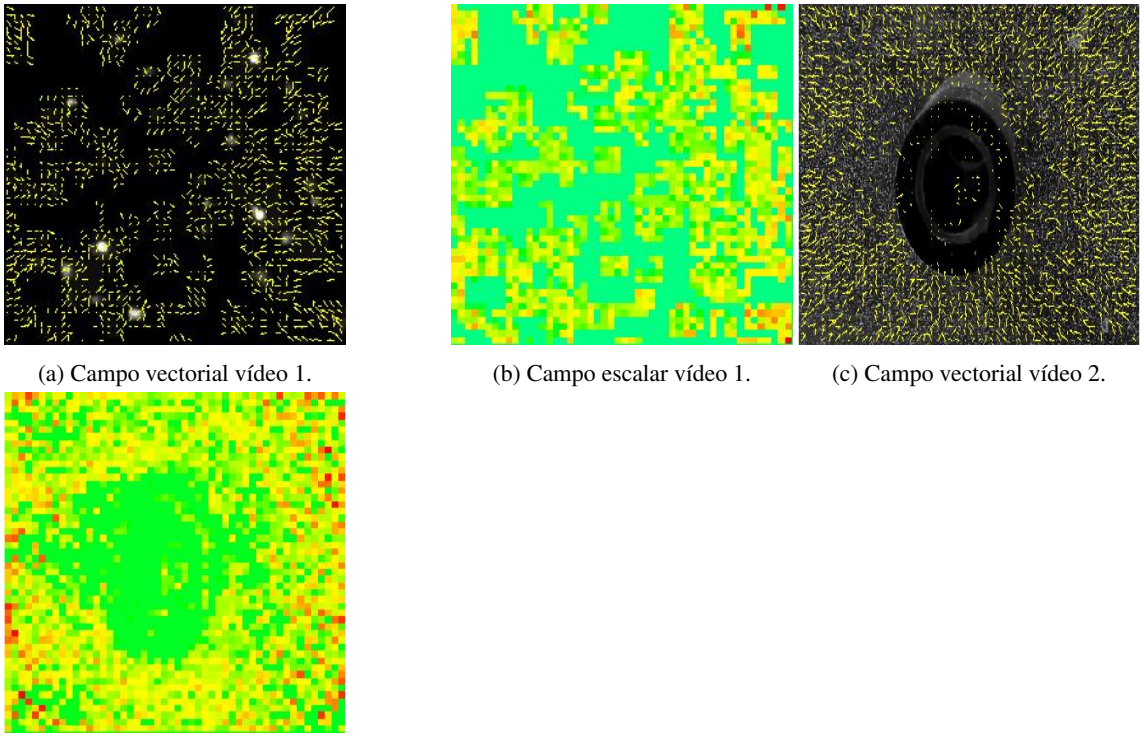
Las velocidades recopiladas con estos datos son del orden de $0{,}03 - 0{,}07\ m/s$. Es pues que los resultados con este número de ventanas y esta cantidad de partículas son bastante buenos pero mejorables.

6.1.2. Selección de 50 ventanas.

En la Figura (6.4) se muestran los resultados, que devuelve el algoritmo, de la selección de un tamaño desmesurado de ventanas para ambos vídeos, 50 en concreto.

Las velocidades que nos ofrecen estos resultados no se corresponden con la realidad de los vídeos, por lo que no muestran coherencia, ya que valores como $0{,}8\ m/s$ para el vídeo 1 y de $4{,}62\ m/s$ para el vídeo 2 no son los adecuados o los reales para lo que se observa en el vídeo seleccionado.

Por lo tanto, escoger muchas áreas de interrogación no aumenta la precisión ni la fiabilidad del experimento, al contrario, pues se ha de distinguir perfectamente el patrón de partículas, incluyendo al menos



(a) Campo vectorial vídeo 1. (b) Campo escalar vídeo 1. (c) Campo vectorial vídeo 2.

(d) Campo escalar vídeo 2.

Figura 6.4: Resultados vídeos 1 y 2, con 50 áreas de interrogación. Fuente [Autor].

5 partículas por región. En este caso no ocurre esto, con lo cual se rechaza este método, sin importar qué cantidad de partículas haya.

6.1.3. Selección de 20 ventanas.

Continuando con el estudio de la cantidad de ventanas y de partículas, se seleccionan 20 ventanas.

Análisis del vídeo 1.

(a) Campo vectorial (b) Campo escalar

Figura 6.5: Resultados vídeo 1, 20 ventanas, con menor cantidad de partículas. Fuente [Autor].

En este caso, el resultado de velocidades para este número de ventanas (20) es más coherente para este vídeo, entre $0{,}5 - 0{,}7\ m/s$. No obstante, no se observa un patrón claro que marque la dirección del flujo.

Análisis del vídeo 2.

(a) Campo vectorial (b) Campo escalar

Figura 6.6: Resultados vídeo 2, 20 ventanas, con mayor cantidad de partículas. Fuente [Autor].

Para este tipo de vídeo (con muchas partícula), 20 ventanas siguen siendo demasiadas. Pues el algoritmo es incapaz de seguir la trayectoria real de las partículas, devolviendo unos valores de velocidad bastante irreales ($3 - 4\ m/s$).

6.1.4. Selección de 10 ventanas.

Por último se seleccionan 10 ventanas, siendo el tamaño intermedio que mejor resultados es capaz de ofrecer, pues concuerdan los valores de velocidad con la realidad y para ambos vídeos (más el segundo que el primero) es capaz de mostrar el comportamiento del flujo con bastante semejanza a la realidad, mediante el campo de vectores.

De esta manera se muestran y analizan los resultados de cada vídeo.

Análisis del vídeo 1.

Como ya se ha visto en todas las configuraciones anteriores, en el vídeo 1 hay poca cantidad de partículas. A causa de esto, no se observa un patrón claro, pues muchas partículas se desplazan sin seguir fielmente al flujo (van hacia arriba, abajo y hacia atrás).

(a) Campo vectorial (b) Campo escalar

Figura 6.7: Resultados vídeo 1, 10 ventanas, con menor cantidad de partículas. Fuente [Autor].

Viendo el campo de vectores de la Figura (6.7), se contempla claramente la aleatoriedad del movimiento que siguen estas partículas. Así pues, se tendría que mejorar el filtro de ruido o seleccionar una zona más adecuada para mostrar resultados coherentes. Aunque también hay que añadir que tanto en el campo escalar como vectorial se observa un comportamiento bastante real, devolviendo una velocidad de $0{,}5 - 0{,}6\ m/s$ (Reflejando más o meno el comportamiento real).

Análisis del vídeo 2.

Simplemente con ver el campo vectorial, ya se puede apreciar que los resultados van a ser mejores y de mayor calidad y realismo. Pues una velocidad de $0{,}05 - 0{,}1\ m/s$ corresponde con al velocidad del flujo del vídeo seleccionado.

Viendo la Figura (6.8), si se observa el campo escalar, se comprueba que antes del obstáculo las partículas se mueven a una velocidad uniforme y siguiendo un flujo laminar $0{,}5 - 0{,}6\ m/s$. Una vez llegan al obstáculo, las partículas se aceleran en la parte superior e inferior llegando a valores de $0{,}1\ m/s$, dejando sin movimiento la parte trasera del obstáculo. Todo esto se justifica viendo el seguimiento que marca el campo vectorial.

(a) Campo vectorial

(b) Campo escalar

Figura 6.8: Resultados vídeo 2, 10 ventanas, con mayor cantidad de partículas. Fuente [Autor].

Por lo tanto, se puede afirmar tras comprobarlo experimentalmente, que es importante el número de ventanas seleccionadas, pues puede influir sustancialmente en los campos de velocidad que muestran los resultados. Por otro lado, la cantidad de partículas es imprescindible para la obtención de resultados coherentes, pues se podrá ver mucho mejor el patrón de movimiento del flujo. A la vista están los resultados obtenidos para el vídeo 1 (menos partículas) y el vídeo 2 (más partículas).

6.2. Interpretación de los resultados.

Comprendiendo y aplicando lo anterior, sabiendo que puede ser distinto para cada vídeo, se pueden obtener grandes resultados. En la Figura (6.9), se aprecia que la magnitud y la dirección de la flechas del campo vectorial es del mismo orden y equiespaciadas, corroborando que se ha seleccionado un patrón adecuado y con velocidades de partículas sino iguales, muy semejantes entre ellas.

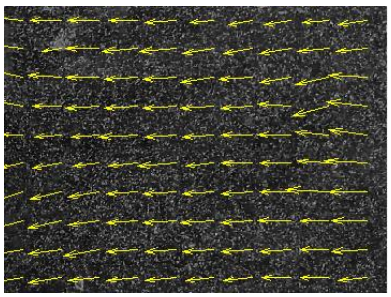


Figura 6.9: Campo de vectores del vídeo 2 (antes de llegar al obstáculo), flujo laminar rectilíneo. Fuente [Autor].

Además, como observa en la Figura (6.8) hay un vacío de vectores en la parte central de la imagen, esto corresponde con el obstáculo introducido, y que por lo tanto permite interpretar el comportamiento de las partículas cuando se encuentran con esto.

Es posible que algunas partículas se desplacen en el sentido opuesto al general de todas, como se observa en la Figura (6.7), esto puede ser debido a obstáculos o a turbulencias en el flujo. Por esto último, en los experimentos realizados en el laboratorio, se ha intentado conseguir un régimen de flujo laminar y coherente, sin muchas alteraciones. No obstante, que escasas partículas se desvíen no va a influir en gran consideración al análisis correcto de los resultados que es el objetivo principal de esta parte.

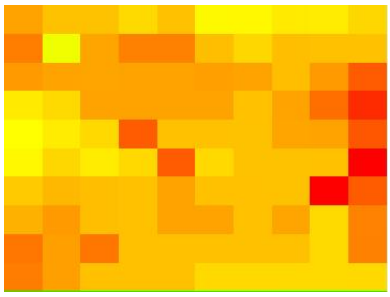


Figura 6.10: Campo escalar del vídeo 2 (antes de llegar al obstáculo), flujo laminar rectilíneo. Fuente [Autor].

De esta manera y aplicando todo lo anterior, es posible conseguir una imagen que refleje de la manera más precisa posible todo lo estimado para cualquier vídeo a analizar. En este caso, la Figura (6.10) muestra un rango de velocidades de $0{,}3$ a $0{,}6 m/s$, mostrando el módulo de la velocidad y comparando las tonalidades de colores bastante distintas entre los dos ejes del sistema de referencia.

Por otro lado, se puede distinguir la velocidad en los ejes x e y. De esta manera, observando la Figura (6.11) se observa que al ser un flujo laminar rectilíneo sobre el eje x, prácticamente todo el valor del módulo de la velocidad está en el eje x, siendo casi nulo el movimiento en el eje y.

(a) Campo escalar eje x

(b) Campo escalar eje y.

Figura 6.11: Resultados vídeo 2, 10 ventanas, con mayor cantidad de partículas. Fuente [Autor].

Donde la velocidad en x es azul porque se desplaza en sentido contrario al eje de referencia, pero el módulo de esta componente tendrá el mismo valor.

Es importante saber que este Software unido a la grabación de los vídeos, no es lo mismo que lo que se puede llegar a obtener en un laboratorio PIV con programas especializados en este tipo de funcionamiento. Pero los resultados obtenidos y la sencillez de la interfaz, permiten que los resultados obtenidos sean lo suficientemente buenos para el uso que va a tener en el ámbito académico, o al menos, esa es la intención.

6.3. Ejemplos con otros vídeos.

En esta sección se van a mostrar varios ejemplos de los resultados obtenidos con algunos vídeos y capturas de una calidad considerable que han sido descargados de Internet. El objetivo consiste en realizar probaturas con el programa y observar como la interfaz devuelve resultados cada vez mejores. Todo esto para posteriormente conseguir, con los vídeos definitivos del laboratorio, los mejores resultados posibles.

6.3.1. Ejemplo 1.

Este vídeo muestra un fluido viscoso con bastantes partículas, acto seguido deja caer una esfera pesada para que se sumerja en el flujo. La región analizada tiene un tamaño de 200×200 píxeles. En la Figura (6.12) se muestran los resultados.

(a) Campo vectorial ejemplo 1.

(b) Campo escalar ejemplo 1.

Figura 6.12: Resultados vídeo ejemplo 1, 8 ventanas, 30 FPS. Fuente [Autor].

El comportamiento que se observa de las partículas es el esperado, pues únicamente se observará variación de velocidad de las partículas que van entrando en contacto, en este caso se estudia un intervalo de tiempo de $0{,}2\ segundos$. Las velocidades a las que van las partículas que se encuentran en los bordes de la esfera son del orden de $0{,}6 - 0{,}9\ m/s$.

6.3.2. Ejemplo 2.

En este caso se analiza un flujo rectilíneo bastante rápido, lo que genera turbulencias durante su recorrido en la zona superior. En la Figura (6.13) se observa esa tendencia recta con una velocidad de $0{,}3 m/s$ en una región de 300×300 píxeles.

(a) Campo vectorial ejemplo 2.

(b) Campo escalar ejemplo 2.

Figura 6.13: Resultados vídeo ejemplo 2, 5 ventanas, 20 FPS. Fuente [Autor].

6.3.3. Ejemplo 3.

Por último, se emplea la opción de la interfaz de tomar dos fotografías obtenidas online y de alta calidad. Estas dos capturas muestran un pequeño movimiento de partículas en 2D y en espiral. En este caso, al ser las imágenes de alta calidad, el patrón de capta fácilmente, reflejando un aumento de la velocidad en las zonas centrales. Para ello se selecciona una región de 600×400 píxeles.

(a) Campo vectorial ejemplo 3.

(b) Campo escalar ejemplo 3.

Figura 6.14: Resultados capturas ejemplo 3, 10 ventanas. Fuente [Autor].

En la Figura (6.14) se muestra perfectamente el comportamiento mencionado, el movimiento es bastante rápido, así que en un intervalo de $0{,}5\ segundos$, la velocidad es de $0{,}1\ m/s$.

6.4. Resultados definitivos.

En esta sección se van a analizar los últimos vídeos y por tanto los definitivos, que se han obtenido con la configuración del laboratorio, corroborando e introduciendo todos los parámetros y mejoras posibles que se han indicado con ejemplos en este proyecto. Para analizar los resultados es importante diferenciar si tiene o no obstáculo (cilindro inmerso en el fluido), de esta manera se podrá interpretar el comportamiento del flujo mediante el análisis de las partículas, finalidad de la técnica PIV.

6.4.1. Flujo laminar sin obstáculo.

En primer lugar, se analiza el comportamiento del fluido de en un extremo del cajón de metacrilato sin un obstáculo, coincidiendo con la zona opuesta a la manguera propulsora, de esta manera se consigue el flujo laminar continuo deseado para analizarlo, ver Figura (6.15).

(a) Campo vectorial.

(b) Campo escalar.

Figura 6.15: Resultados vídeo definitivo sin obstáculo (cilindro). Fuente [Autor].

Los resultados obtenidos son bastante coherentes, pues para una ventana seleccionada de 300×300 píxeles (región con muchas partículas y que siguen un comportamiento homogéneo), dividida el 5 áreas de interrogación y con un incremento de tiempo de $10ms$, la velocidad del flujo laminar indica que es del orden del $0{,}1\ a\ 0{,}3\ m/s$. Estos datos muestran el comportamiento real del fluido, indicado por las partículas.

Como se observa, hay vectores que van hacia arriba, esto es debido a que el flujo no es perfectamente recto. Se debe mencionar que al ser un cajón corto y propulsado por una bomba, es normal que hay algunas irregularidades y las partículas se desordenen levemente. Esto se podría solucionar con una mayor cantidad de partículas y un cajón más largo, de manera que estudiando la parte del final se obtenga un flujo laminar casi perfecto.

Cabe destacar que para la obtención de estos resultados, de bastante calidad, se ha empleado la máxima potencia del láser, 80 %. No obstante, a modo de prueba se han realizado grabaciones y se han obtenido resultados con una potencia al 20 y al 50 %, ver Figura (6.16).

(a) Láser al 80 %. (b) Láser al 50 %. (c) Láser al 20 %.

Figura 6.16: Grabaciones con distinta potencia. Fuente [Autor].

Así pues, se justifica el hecho de que la potencia del láser tiene que ser lo suficientemente alta para que las partículas reflejen la luz necesaria para que puedan ser estudiadas. Por supuesto, contando con la pérdida que produce el paso por la lente que provoca el plano del haz.

6.4.2. Flujo laminar con obstáculo.

En segundo lugar, se analiza el comportamiento del fluido introduciendo el cilindro que hará de obstáculo, para llevarlo a cabo, se han grabado a distintas frecuencias (200 y 100 Hz). Ver figuras (6.17) y (6.19).

(a) Campo vectorial. (b) Campo escalar.

Figura 6.17: Resultados vídeo definitivo con obstáculo (cilindro) a 200 Hz. Fuente [Autor].

En este caso, aunque menos preciso que el anterior (sin obstáculo), también se obtienen unos resultados concisos. La ventana seleccionada es del tamaño de 450×500, dividida en 10 áreas de interrogación y con un incremento de tiempo de $20ms$.

La irregularidad mostrada por los vectores se puede justificar mediante la colocación del láser. Como se ha descrito en el capítulo anterior el láser ilumina desde arriba, por lo tanto si colocamos un obstáculo en medio no dejará pasar la luz a las zonas inferiores y laterales, dejando un vacío que dificulta el seguimiento correcto del movimiento de las partículas.

No obstante, si únicamente se selecciona una pequeña región por encima del obstáculo (ventana de 300×300 y Δt de 10 ms), se puede observar un comportamiento coherente del fluido, justificando que las partículas se desvían con dicho obstáculo. Ver Figura (6.18).

(a) Campo vectorial. (b) Campo escalar.

Figura 6.18: Resultados vídeo definitivo con obstáculo, seleccionando región superior del cilindro, a 200 Hz. Fuente [Autor].

La velocidad que devuelve la interfaz está bastante dispersada, pues antes de llegar al cilindro por la parte superior e inferior, la velocidad del flujo laminar es del orden de $0{,}3m/s$. En la parte trasera del cilindro, coincidiendo con la zona donde se produce el aumento de velocidades en régimen turbulento, la magnitud de las velocidades asciende hasta el orden del $12{,}5m/s$. De esta manera, más o menos, se refleja un comportamiento real del fluido mediante el análisis de las partículas.

En el caso de 100 Hz de frecuencia, se ha aprovechado el incremento de pulsos durante la grabación, $10ms$ cada $90ms$ de reproducción. De esta manera los resultados analizados con un Δt de $10ms$ son los siguientes.

(a) Campo vectorial. (b) Campo escalar.

Figura 6.19: Resultados vídeo definitivo con obstáculo (cilindro) a 100 Hz. Fuente [Autor].

Así pues, con láser pulsado de 100 Hz se tienen resultados similares a 200 Hz.

Capítulo 7

Mejoras y usos avanzados de la técnica

Posteriormente al analizar y conocer a fondo la técnica PIV, se necesitan varias condiciones tanto espaciales como temporales para la obtención de unos resultados claros y concisos. Por lo tanto, principalmente se pueden extraer las siguientes limitaciones:

- El tiempo de retraso entre los impulsos del láser debe ser lo suficientemente largo como para capturar el desplazamiento de las partículas correctamente y suficientemente corto para que las partículas tengan una componente de velocidad fuera del plano que se salga del plano de luz utilizado.
- Con el uso de láseres de alta potencia, se puede reducir el tamaño de las partículas trazadoras. La exactitud de las mediciones de PIV mejorará drásticamente a medida que las partículas sigan el flujo en un tamaño de muestra más reducido.
- El tamaño del área de interrogación debe ser pequeño para que no haya un gradiente de velocidad significativo, lo que puede provocar que sea más complicado la obtención de resultados.

Utilizando estas limitaciones, existen varias alternativas y mejoras de la técnica PIV.

7.1. Stereo PIV (3D PIV)

Generalmente, se miden dos componentes de velocidad utilizando la técnica PIV estándar (2D PIV). Así pues, usando un enfoque estereoscópico, se pueden medir y registrar todas los componentes de las tres velocidades correspondientes a los tres ejes del espacio tridimensional, dando como resultado vectores de velocidad 3D instantáneos para todo el área [19].

La técnica PIV de tres componentes (3D PIV) se basa en el mismo principio fundamental que la vista del ojo humano: visión estéreo.

Nuestros ojos ven imágenes ligeramente diferentes del mundo que nos rodea, y comparando estas imágenes, el cerebro es capaz de hacer una interpretación tridimensional. Con un solo ojo, es perfectamente capaz de reconocer el movimiento hacia arriba, hacia abajo o hacia los lados, pero es difícil de juzgar las distancias y el movimiento cercano o lejos del observador. En esta técnica, las cámaras juegan el papel de los 'ojos'. La determinación más precisa del desplazamiento fuera del plano (es decir, la velocidad) se logra cuando hay 90^{o} entre las dos cámaras. La cámara y la lente están alineadas con el plano de luz mediante un ángulo.

Se produce una distorsión de los campos de imagen. Por lo tanto, el ángulo se mantiene tan pequeño que el desenfoque es aceptable. Para superar el problema de enfoque en el desplazamiento angular, se debe satisfacer la condición de Scheimpflug (el plano de la lente, el plano del objeto y el plano de la imagen para cada cámara tienen que intersectar en una línea común). El efecto secundario de usar este método es una fuerte distorsión de la perspectiva. La distorsión de la perspectiva se evita utilizando una función de correlación de segundo orden. La función de asignación es la matriz que transforma el sistema de coordenadas

de píxeles en sistemas de coordenadas reales.

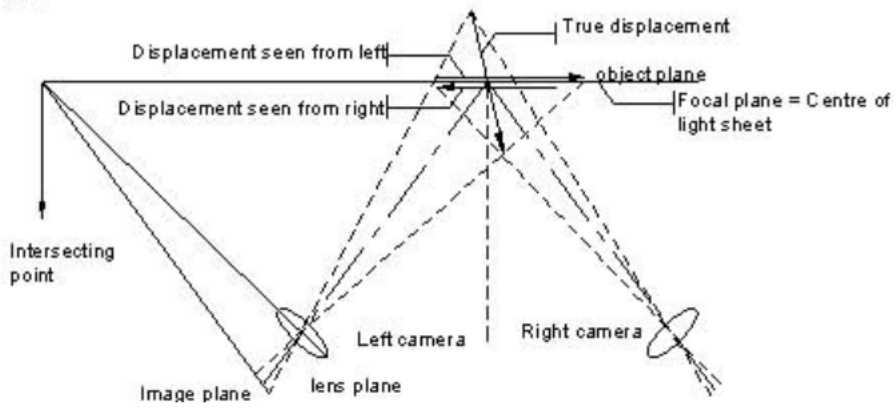


Figura 7.1: Configuración técnica 3D PIV. Fuente [19].

7.1.1. Condición de Scheimpflug

En esta técnica, el plano de la imagen, el plano del eje de la lente y el plano del objeto (plano de luz) se encuentran en el mismo punto, como se muestra en la Figura (7.1). Esta configuración se puede lograr montando la lente de la cámara con un desplazamiento angular, como en la configuración de la cámara estereoscópica de desplazamiento angular y luego inclinando el cuerpo de la cámara (con respecto a la lente). El enfoque correcto y ángulos de inclinación se logran cuando todas las partículas dentro del campo de visión de la cámara están en buen enfoque. La actualización en línea de las imágenes capturadas por la cámara hace que este proceso de configuración sea sencillo.

La disposición de Scheimpflug, ver Figura (7.6), permite mantener el plano de mejor enfoque en el plano de la lámina de luz tanto para los flujos de aire como de agua mientras la cámara visualiza la lámina de luz desde el ángulo que está fuera del eje. La configuración de Scheimpflug introduce una distorsión en perspectiva a las imágenes que hacen que un rectángulo en el plano de la lámina de luz, que será representado como un trapecio en el sensor de imagen.

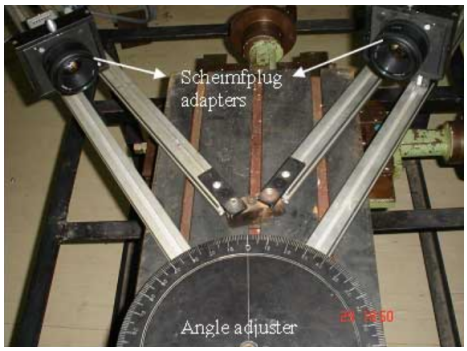


Figura 7.2: Vista fotográfica de la disposición del desplazamiento para el experimento estéreo PIV. Fuente [19].

7.1.2. Calibración

Para realizar la calibración y ajuste de la cámara se utiliza el siguiente procedimiento, ver Figura (7.3):

- Se graban dos imágenes de un objetivo de calibración.
- Se introducen marcadores contenidas en el objetivo de calibración en posiciones conocidas.

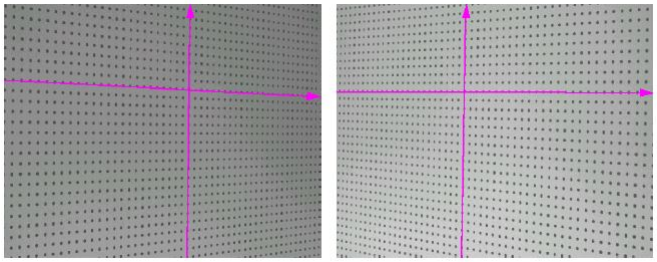


Figura 7.3: Calibración de la cámara PIV. Fuente [Autor].

- Por tanto, se comparan las posiciones conocidas de los marcadores con el marcador correspondiente a las posiciones de cada imagen de la cámara. Siendo estos los parámetros del modelo necesarios para dar el mejor ajuste posible.

7.1.3. 3D PIV sprays

En este caso se estudia un campo de flujo instantáneo generado por la propagación de un spray Diesel en dos atmósferas con distintas densidades [22].

De esta manera se estudia:

- La entrada de aire y la formación de la mezcla.
- Los vórtices y el desarrollo del spray.

El campo del fluido cercano al spray Diésel no evaporado, así como la superficie del spray usando un láser Nd-YAG y un PIV.

Un estudio de los mecanismos de mezclado de fuel con el aire de alrededor ayuda a entender los mecanismos de formación del spray, así como la atomización del fuel y las características de entrada del aire, ver Figura (7.5).

En la Figura (7.4) se observan los resultados de un inyector multipunto (MPI), dos milisegundos después del SOI (inicio de la inyección). Los colores de fondo representan la velocidad en la dirección del eje z, mientras que la dirección positiva apunta hacia el plano de la imagen.

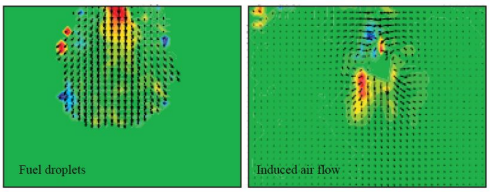


Figura 7.4: Ejemplo de campo vectorial y escalar de un spray. Fuente [Autor].

Las velocidades en el interior son más altas que en los bordes de los conos de rociado, principalmente debido a los diferentes tamaños de gotitas. Las gotitas más grandes conservan su impulso inicial durante un período de tiempo más largo, mientras que las gotitas más pequeñas son desaceleradas y desplazadas a los lados de los conos de pulverización, lejos de las gotitas más grandes por el flujo de aire inducido.

El spray de un inyector multipunto (MPI) se estudió por medio de PIV estereoscópico, ver Figura (7.5).

La distribución global de las gotitas de combustible y el flujo de aire inducido se registraron utilizando partículas fluorescentes para visualizar el flujo de aire. Además de dos cámaras CCD de doble obturador y óptica Scheimpflug. Para el plano de luz láser se emplean dos lentes cilíndricas con un espesor de 2 mm.

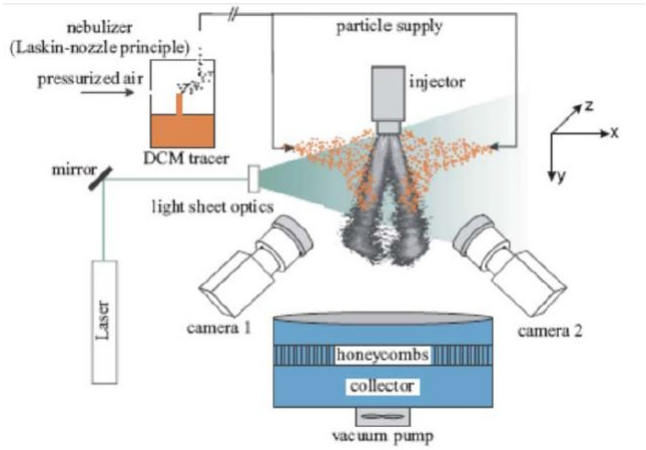


Figura 7.5: Instalación del estudio del chorro de un inyector. Fuente [22].

7.2. Micro PIV

La velocimetría de imágenes de partículas de resolución micrométrica (Micro PIV) es una herramienta para medir el perfil de velocidad a través de un plano en un dispositivo microfluídico. Debido a las pequeñas dimensiones del campo en el flujo de micro canales, es imposible utilizar sistemas PIV convencionales para obtener dos planos ortogonales que tengan acceso óptico al campo de flujo. En su lugar, los micro sistemas PIV utilizan una técnica de iluminación de volumen donde la fuente de luz y el campo de visión se introducen a través de la misma óptica de un microscopio, ver Figura (7.6). Con este enfoque el plano focal se mueve hacia abajo a través del campo de flujo para asignar todo el volumen [20].

La siembra de partículas es uno de los elementos más importantes para no tener éxito en los resultados que devuelve Micro PIV, por una serie de razones. En primer lugar, las partículas de siembra proporcionan una señal fluorescente fuerte. Segundo, las longitudes de onda de excitación y emisión de partículas de siembra, son compatibles con el resto del sistema óptico, que está diseñado para maximizar la relación señal/ruido. Por último, los diámetros de hasta $100nm$ están disponibles con el fin de hacer frente al continuo aumento de los altos requisitos de resolución espacial en la comunidad microfluídica.

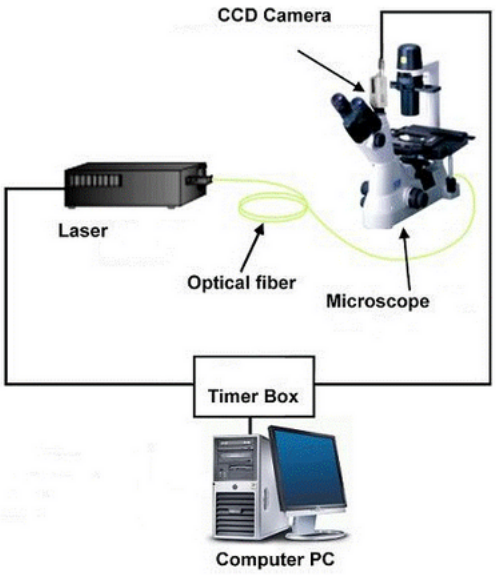


Figura 7.6: Sistema Micro PIV. Fuente [20].

7.3. Aplicaciones en motores

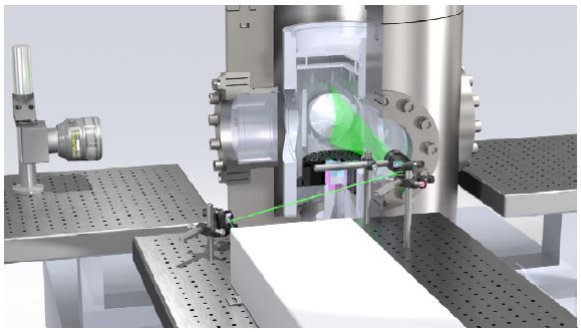


Figura 7.7: Instalación para el estudio del cilindro de un motor de 2T. Fuente [Autor].

Una de las aplicaciones importantes del PIV, consiste en el estudio y la observación del movimiento de un chorro de inyección a alta Presión y Temperatura.

Existen proyectos realizados en un "motor maqueta", ver Figura (7.7), siendo un motor grande de 2 tiempos. De manera que mediante PIV, se pueda obtener el campo de velocidades instantáneo, ver Figura (7.8).

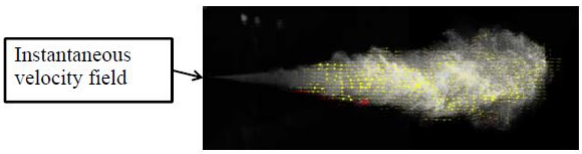


Figura 7.8: Campo de vectores del chorro del motor. Fuente [Autor].

El problema de la instalación es que ciclo a ciclo hay una dispersión elevada, por lo tanto para mayor precisión la técnica que se usa es LIF [1]. Por lo tanto, PIV es más preciso y se utiliza mejor en túneles de viento, aunque en cuestiones aerodinámicas sea una técnica básica.

[1] Fluorescencia Inducida por Láser.

Capítulo 8

Conclusiones

El estudio de fluidos mediante dos imágenes consecutivas o capturas de un vídeo (PIV) está sujeto a errores y puede producir resultados no fiables que pueden variar en exceso si la relación del tamaño de las áreas de interrogación con el número de partículas visibles no es la adecuada. Así pues, el objetivo de este proyecto consiste en obtener mediante la instalación del laboratorio y la interfaz desarrollada varios vídeos de calidad y precisos, acompañados a su vez de sus campos de velocidad correspondientes, ofreciendo unos resultados reales y coherentes.

Por lo tanto, PIV es una técnica que es capaz de obtener resultados relevantes y que sea sencilla de comprender. De esta manera permite conocer de una manera más dinámica el mundo de la mecánica de fluidos, consiguiendo fomentar el interés de esta rama de la física por parte del alumno.

La interfaz creada en el proyecto muestra en gran medida las posibilidades que ofrece MATLAB. Permitiendo desarrollar con claridad y sencillez una ventana multifunción que reproduzca un vídeo, consiga dos capturas del mismo con un intervalo de tiempo predeterminado y devuelva los campos de velocidad de dichas imágenes como resultados.

Finalmente, ya que este trabajo tiene una finalidad académica se puede afirmar que existe muchísimo margen de mejora. Pudiendo aumentar la precisión de la interfaz desarrollada mediante métodos también descritos en esta memoria y la precisión de la instalación del laboratorio para obtener vídeos mejores y por tanto resultados de mayor calidad. No obstante, los objetivos principales del proyecto, como se muestra en cada uno de los capítulos, se han logrado consistentemente.

Capítulo 9

Presupuesto

ELEMENTOS MATERIALES	Coste en €	Coste del % de uso exclusivo del proyecto en €
1. Licencia Matlab	2000 €	5 % 100 €
2. Cajón de metacrilato	-	100 % 120 €
3. Agua destilada	-	100 % 6 €
4. Bomba Compact 5000+, Marca EHEIM	-	100 % 112 €
5. Partículas trazadoras, Marca Etalon research (1 bote)	-	100 % 500 €
6. Láser ACCULASE PWM 650 nm y 5 mW	-	100 % 124 €
7. Cámara de alta velocidad, Marca MIKROTRON Eo Sens MC	6000 €	10 % 600 €
8. Lona negra para tapar el cajón	-	100 % 5 €
PRESUPUESTO TOTAL MATERIAL	-	1567€

MANO DE OBRA	Coste por hora	Horas	Coste en €
Implementación y desarrollo de la interfaz gráfica	25€/hora	80 horas	2000 €
Configuración y pruebas de la instalación	15€/hora	25 horas	375 €
Análisis y recopilación de los resultados	20€/hora	20 horas	400 €
Búsqueda de información y redacción de la memoria	15 €/hora	60 horas	900€
PRESUPUESTO TOTAL MANO DE OBRA	-	-	3675€

COSTE TOTAL DEL PROYECTO	4932 €

Bibliografía

[1] JONATHAN GOLDSMITH, *Particle Image Velocimetry and it's Applications to Granular Media, 10, 2012*.

[2] R. J. ADRIAN, *Twenty years of particle image velocimetry*, **Experiments in Fluids, 1984.**.

[3] S. WERELEY J. KOMPENHANS M. REL, C. WILLERT, *Particle Image Velocimetry a Practical Guide, 2007*.

[4] MOHSEN JAHANMIRI, *Particle Image Velocimetry: Fundamentals and It's Applications*, **Göteborg, Sweden, 2011**.

[5] http://www.dantecdynamics.com/seeding-materials. Feb 2017.

[6] PPT, **Tracer Particles and Seeding for PIV**. Mar 2017.

[7] M.RAFFEL, C.WILLERT, AND J.KOMPEHANS, *PIV: A Practical Guide. Springer*, **1st edition edition, 2002**.

[8] http://www.dantecdynamics.com/volumetric-illumination-optics. Mar 2017.

[9] OLLE TÖRNBLOM, *Introduction course in particle image velocimetry*, **March 31, 2004**.

[10] http://www.dantecdynamics.com/time-resolved-piv. Mar 2017.

[11] KATARÍNA RATKOVSKÁ, *Particle Image Velocimetry Publication was supported by project: „Budování excelentního vědeckého týmu pro experimentální a numerické modelování v mechanice tekutin a termodynamice" Project registration number: CZ*, **Univerzitní 22, 306 14, Pilsen, 2009**.

[12] ROB J .M. BASTIAANS, *Cross-correlation PIV; theory, implementation and accuracy*, **Department of Mechanical Engineering, Section of Internal Combustion Engines, Eindhoven University of Technology, 2004**.

[13] JOSÉ MIGUEL IRIARTE MUÑOZ, *Velocimetría PIV en tiempo real basada en lógica programable FPGA*, **Junio de 2008**.

[14] T.ZHANG AND S.W.VAN SCIVER, *Use of the particle image velocimetry technique to study the propagation of second sound shock in superfluid helium. Physics of Fluids*, **6:99—102, 2004**.

[15] DIEGO ORLANDO BARRAGÁN GUERRERO, *Manual de Interfaz Gráfica de Usuario en Matlab, 2009* .

[16] DAVID GARCÍA PÉREZ, *MANEJO BÁSICO DE IMÁGENES CON MATLAB*, **PRÁCTICA 1 Grupo de Visión Artificial, 2012**.

[17] J. WESTERWEEL & C. POELMA OF TECHNICAL UNIVERSITY OF DELFT ADAPTED BY K. KIGER, *Burgers Program For Fluid Dynamics Turbulence School*, **College Park, Maryland, May 24-27**.

[18] LEWIS HOWARD, *The Investigation of Surface Flow using Particle Image Velocimetry, 2013*.

[19] http://www.onera.fr/en/daap/piv?page=2, Jul 2017.

[20] www.lavision.com/en/products/flowmaster/micro-piv/index.php, Jul 2017.

[21] DE HAETIG J, *Introductory on particle behavior ISL / AGRAD* , **taller sobre anemometría láser (Institute Saint Louis) informe R 117/76, 1976**.

[22] B. RICHTER, G. ROTTENKOLBER, M. HEHLE, K. DULLENKOPF, AND S. WITTIG, **ILASS-Europe 2001**.

[23] ANTONIO JOSÉ PÉREZ VIDAL, **Desarrollo de un sistema PIV (velocímetro por imagen de partículas) didáctico, 2016**

Capítulo 10

Anexos

10.1. Anexo 1 (Explicación transformadas de Fourier)

10.1.1. Transformadas de Fourier

La Transformada de Fourier se puede llevar a cabo generalmente en una, dos o más dimensiones. La importancia de la Transformada de Fourier en 1D es que la función es representada por cada función periódica 1D como una suma exponencial compleja de un número de medios, como la suma del seno y coseno, indicadas en las ecuaciones de Euler.

$$e^{i\varphi} = cos\varphi + i.sin\varphi \tag{10.1}$$

En la práctica, la Transformada de Fourier se utiliza para expresar la señal dependiente del tiempo utilizando las funciones armónicas seno y coseno. También puede interpretarse como la transformación del dominio del tiempo de transferencia de una señal en la frecuencia. La Transformada de Fourier 2D proporciona una función de distribución de la suma del seno y del coseno como una Transformada de Fourier 1D. En este caso la función es de dos variables, sean (x, y). Esta función no puede representar una imagen digital que capture la cámara con PIV.

Transformada Fourier 1D:

$$F(\omega) = \int_{-\infty}^{+\infty} f(t)e^{-i\omega t}dt \tag{10.2}$$

Transformada inversa de Fourier 1D:

$$f(\omega) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} f(t)e^{-i\omega t}d\omega \tag{10.3}$$

Transformada Fourier 2D:

$$F(u, v) = \int\int_{-\infty}^{+\infty} f(x, y)e^{-i(xu,yv)}dxdy \tag{10.4}$$

Transformada inversa de Fourier 2D:

$$F(x, y) = \frac{1}{4\pi^2}\int_{-\infty}^{+\infty} F(u, v)e^{-i(xu,yv)}dudv \tag{10.5}$$

10.1.2. Transformada discreta de Fourier

La ecuación que define Fourier necesita la comprensión de la expresión matemática de la señal o del espectro de un intervalo finito. El problema es cómo determinar el rango de las muestras de la señal o la señal del espectro de la muestra. Para este propósito, se utiliza el método numérico Transformada discreta de Fourier. La señal de entrada se ve como una secuencia $f(n)$ con elementos

$n = 0, 1, \ldots, n = N - 1$. Consecuentemente, $f(k)$ es el espectro de Fourier de la señal $f(n)$.

Expresiones matemáticas de la Transformada discreta de Fourier y la Transformada discreta de Fourier inversa:

$$F(k) = \sum_{n=0}^{N-1} f(n) e^{\frac{-2\pi i k n}{N}} \qquad f(n) = \frac{1}{N} \sum_{n=0}^{N-1} F(k) e^{\frac{-2\pi i k n}{N}} \tag{10.6}$$

10.1.3. Método Wienerova - Teorema de Khinchin

Método Wienerova - Teorema de Khinchin es un segundo método de aplicación de una Transformada de Fourier. El método se utiliza para calcular la autocorrelación, y además la correlación cruzada. Para obtener una grabación se utiliza como la primera Transformada de Fourier. En consecuencia, está determinada por la raíz absoluta de la función compleja del espectro de potencia. La función de correlación se obtiene entonces aplicando una Transformada discreta de Fourier inversa.

10.1.4. Transformada rápida de Fourier

La Transformada rápida de Fourier es un algoritmo eficiente para calcular la Transformada discreta de Fourier y su inversa. Mientras que el cálculo de la Transformada discreta de Fourier de la La fórmula tiene que ser $N2$ aritmética para calcular la FFT es suficiente para operaciones aritméticas de $N.logN$.

Para los cálculos numéricos extensos actuales la Transformada rápida de Fourier es indispensable. Sin embargo, el principio del algoritmo FFT, un algoritmo artificial, provoca el aumento del ruido en la correlación, particularmente en los bordes del rango de evaluación. Para eliminar el ruido de la ventana de correlación se utilizan una función de filtro en el dominio de la frecuencia.

10.2. Anexo 2 (Código Matlab)

10.2.1. Primera interfaz: cargarvideopiv.m

```
function varargout = cargarvideopiv(varargin)
%CARGARVIDEOPIV MATLAB code for cargarvideopiv.fig
%      CARGARVIDEOPIV, by itself, creates a new CARGARVIDEOPIV or raises the existing
%      singleton*.
%
%      H = CARGARVIDEOPIV returns the handle to a new CARGARVIDEOPIV or the handle to
%      the existing singleton*.
%
%      CARGARVIDEOPIV('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in CARGARVIDEOPIV.M with the given input arguments.
%
%      CARGARVIDEOPIV('Property','Value',...) creates a new CARGARVIDEOPIV or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before cargarvideopiv_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to cargarvideopiv_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help cargarvideopiv

% Last Modified by GUIDE v2.5 17-Mar-2017 13:55:45

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @cargarvideopiv_OpeningFcn, ...
                   'gui_OutputFcn',  @cargarvideopiv_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT


%--- Executes just before cargarvideopiv is made visible.
function cargarvideopiv_OpeningFcn(hObject, eventdata, handles,
    varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to cargarvideopiv (see VARARGIN)


lista = VideoReader.getFileFormats();
ext = {lista.Extension};
lista_ext1 = '';

for i=1:length(ext)-1
        lista_ext1 = [lista_ext1,'*.',ext{i},';'];
        lista_ext2 = strrep(lista_ext1,';',',');


end


handles.datos.lista_ext1 = [lista_ext1,'*.',ext{end}];
handles.datos.lista_ext2 = strrep(lista_ext1,';',',');


%Empezamos el temporizador

handles.hTimer = timer('ExecutionMode','fixedRate','Period',1);

set(handles.figureVideo,'CurrentAxes',handles.ejesVideo)

axis off

%colocamos la imagen a los pushbutton

[a,map]=imread('itunes2.png');
[r,c,d]=size(a);
x=ceil(r/100);
y=ceil(c/100);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.botonPlayPause,'CData',g);

[a,map]=imread('next.jpg');
[r,c,d]=size(a);
x=ceil(r/100);
y=ceil(c/50);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.Next,'CData',g);


% Choose default command line output for cargarvideopiv
```

```
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);


%UIWAIT makes cargarvideopiv wait for user response (see UIRESUME)
% uiwait(handles.figureVideo);

%--- Outputs from this function are returned to the command line.
function varargout = cargarvideopiv_OutputFcn(hObject, eventdata,
    handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


%--- Executes on button press in cargarvideo.
function cargarvideo_Callback(hObject, eventdata, handles)
% hObject    handle to cargarvideo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[FileName, PathName] = uigetfile({handles.datos.lista_ext1,['Ficheros
    de video aceptables (',handles.datos.lista_ext2,')'];'*.*', 'Todos
     los ficheros (*.*)'},'Elige el video de la secuencia a procesar');

if ~isequal(FileName,0)
    set(handles.figureVideo,'CurrentAxes',handles.ejesVideo);
    fichero = [PathName,FileName];

    try
        objVideo = VideoReader(fichero);
    catch
        errordlg('Error al leer el archivo de video','Correlacion');
        return;
    end

    enable_objects(handles);

    global cont temp rep fot_grab;
    cont = 1;
    temp = 0;
    rep = 0;
    fot_grab = [0 0];

    handles.video.numFotogramas = objVideo.NumberOfFrames;
    handles.video.duracion = objVideo.Duration; err
    handles.video.resolucion = [objVideo.Width objVideo.Height];
    handles.video.ratioFotogramas = objVideo.FrameRate;

     % Valores maximos

```

```
153         [handles.datos.min_max, handles.datos.seg_max, handles.datos.
                mil_max] = MinSegMil(handles.video.numFotogramas,handles.video
                .ratioFotogramas);
154
155         % Actualizar textos
156
157         set(handles.labelTitulo,'String',FileName);
158         set(handles.labelDuracion,'String',sprintf('%d'' %4g"',floor(
                handles.video.duracion/60),handles.video.duracion-floor(handles
                .video.duracion/60)*60));
159         set(handles.labelFPS,'String',sprintf('%4g ',handles.video.
                ratioFotogramas));
160
161         % Textos de posicion
162
163         set(handles.minutos,'String','00');
164         set(handles.segundos,'String','00');
165         set(handles.milesimas,'String','000');
166          set(handles.textoFot,'String','1');
167
168         handles.video.videoFotogramas = read(objVideo);
169         dT = round((1/handles.video.ratioFotogramas)*1000)/1000; % Tiempo
                entre fotogramas
170         set(handles.hTimer,'StartDelay',dT,'Period',dT);
171         fotograma = handles.video.videoFotogramas(:,:,:,cont);
172         set(handles.slidervideo,'Min',1,'Max',handles.video.numFotogramas,'
                Value',1,'SliderStep',1/(handles.video.numFotogramas-1)*ones
                (1,2));
173         handles.hImage = image(fotograma);
174         axis off;
175         axis image;
176
177         set(handles.hTimer,'TimerFcn',{@timerCallback,handles});
178
179 end
180
181 guidata(hObject, handles);
182
183 function timerCallback(objTimer,~,handles)
184         global cont;
185         cont = cont + 1;
186         set(handles.hImage,'CData',handles.video.videoFotogramas(:,:,:,cont
                ));
187         [Min, Seg, Mil] = MinSegMil(cont,handles.video.ratioFotogramas);
188         set(handles.minutos,'String',sprintf('%03d',Min));
189         set(handles.segundos,'String',sprintf('%02d',Seg));
190         set(handles.milesimas,'String',sprintf('%03d',Mil));
191          set(handles.textoFot,'String',sprintf('%d',cont));
192         set(handles.slidervideo,'Value',cont);
193         if cont == handles.video.numFotogramas
194             global rep;
195             if rep
196                 cont = 0;
197             else
198                 global temp;
199                 temp = 0;
200                 stop(objTimer);
```

```
201             enable_reproduccion(handles,'on');
202         end
203     end
204
205 function [M,S,m] = MinSegMil(num,ratio)
206     seg = (num-1)/ratio;
207     M = floor(seg/60);
208     S = floor(seg - 60*M);
209     m = floor((seg - M*60 - S)*1000);
210
211 function minutos_Callback(hObject, eventdata, handles)
212 % hObject    handle to minutos (see GCBO)
213 % eventdata  reserved - to be defined in a future version of MATLAB
214 % handles    structure with handles and user data (see GUIDATA)
215
216 % Hints: get(hObject,'String') returns contents of minutos as text
217 %        str2double(get(hObject,'String')) returns contents of minutos
    as a double
218  global cont;
219     prov = str2double(get(hObject,'String'));
220     if prov < 0
221         prov = 0;
222     elseif prov > handles.datos.min_max
223         prov = handles.datos.min_max;
224     end
225     set(hObject,'String',sprintf('%03d',prov));
226     Min = str2double(get(handles.minutos,'String'));
227     Seg = str2double(get(handles.segundos,'String'));
228     Mil = str2double(get(handles.milesimas,'String'));
229     cont = Fotograma(Min,Seg,Mil,handles.video.ratioFotogramas);
230     set(handles.slidervideo,'Value',cont);
231    set(handles.textoFot,'String',num2str(cont));
232     set(handles.hImage,'CData',handles.video.videoFotogramas(:,:,:,cont
        ));
233
234 %--- Executes during object creation, after setting all properties.
235
236 function minutos_CreateFcn(hObject, eventdata, handles)
237 % hObject    handle to minutos (see GCBO)
238 % eventdata  reserved - to be defined in a future version of MATLAB
239 % handles    empty - handles not created until after all CreateFcns
    called
240
241 % Hint: edit controls usually have a white axes1 on Windows.
242 %       See ISPC and COMPUTER.
243 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
244     set(hObject,'BackgroundColor','white');
245 end
246
247 function milesimas_Callback(hObject, eventdata, handles)
248 % hObject    handle to milesimas (see GCBO)
249 % eventdata  reserved - to be defined in a future version of MATLAB
250 % handles    structure with handles and user data (see GUIDATA)
251
252 % Hints: get(hObject,'String') returns contents of milesimas as text
```

```
253 %        str2double(get(hObject,'String')) returns contents of
        milesimas as a double
254 global cont;
255     Min = str2double(get(handles.minutos,'String'));
256     Seg = str2double(get(handles.segundos,'String'));
257     Mil = str2double(get(handles.milesimas,'String'));
258     if Mil < 0
259         Mil = 0;
260     elseif Min == handles.datos.min_max && Seg == handles.datos.seg_max
            && Mil > handles.datos.mil_max
261         Mil = handles.datos.mil_max;
262     elseif Mil > 999
263         Mil = 999;
264     end
265     set(hObject,'String',sprintf('%03d',Mil));
266
267     cont = Fotograma(Min,Seg,Mil,handles.video.ratioFotogramas);
268     set(handles.slidervideo,'Value',cont);
269      set(handles.textoFot,'String',num2str(cont));
270     set(handles.hImage,'CData',handles.video.videoFotogramas(:,:,:,cont
            ));
271
272 %--- Executes during object creation, after setting all properties.
273 function milesimas_CreateFcn(hObject, eventdata, handles)
274 % hObject    handle to milesimas (see GCBO)
275 % eventdata  reserved - to be defined in a future version of MATLAB
276 % handles    empty - handles not created until after all CreateFcns
        called
277
278 % Hint: edit controls usually have a white axes1 on Windows.
279 %       See ISPC and COMPUTER.
280 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
        defaultUicontrolBackgroundColor'))
281     set(hObject,'BackgroundColor','white');
282 end
283
284 function segundos_Callback(hObject, eventdata, handles)
285 % hObject    handle to segundos (see GCBO)
286 % eventdata  reserved - to be defined in a future version of MATLAB
287 % handles    structure with handles and user data (see GUIDATA)
288
289 % Hints: get(hObject,'String') returns contents of segundos as text
290 %        str2double(get(hObject,'String')) returns contents of segundos
         as a double
291   global cont;
292     Min = str2double(get(handles.minutos,'String'));
293     Seg = str2double(get(handles.segundos,'String'));
294     Mil = str2double(get(handles.milesimas,'String'));
295     if Seg < 0
296         Seg = 0;
297     elseif Min == handles.datos.min_max && Seg > handles.datos.seg_max
298         Seg = handles.datos.seg_max;
299     elseif Seg > 59
300         Seg = 59;
301     end
302     set(hObject,'String',sprintf('%02d',Seg));
303
```

```
304     cont = Fotograma(Min,Seg,Mil,handles.video.ratioFotogramas);
305     set(handles.slidervideo,'Value',cont);
306     set(handles.textoFot,'String',num2str(cont));
307     set(handles.hImage,'CData',handles.video.videoFotogramas(:,:,:,cont
            ));
308
309 %--- Executes during object creation, after setting all properties.
310 function segundos_CreateFcn(hObject, eventdata, handles)
311 % hObject    handle to segundos (see GCBO)
312 % eventdata  reserved - to be defined in a future version of MATLAB
313 % handles    empty - handles not created until after all CreateFcns
        called
314
315 % Hint: edit controls usually have a white axes1 on Windows.
316 %       See ISPC and COMPUTER.
317 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
318     set(hObject,'BackgroundColor','white');
319 end
320
321 %--- Executes on slider movement.
322
323 function slidervideo_Callback(hObject, eventdata, handles)
324 % hObject    handle to slidervideo (see GCBO)
325 % eventdata  reserved - to be defined in a future version of MATLAB
326 % handles    structure with handles and user data (see GUIDATA)
327
328 % Hints: get(hObject,'Value') returns position of slider
329 %        get(hObject,'Min') and get(hObject,'Max') to determine range
        of slider
330  global cont;
331     cont = round(get(hObject,'Value'));
332     set(handles.hImage,'CData',handles.video.videoFotogramas(:,:,:,cont
            ));
333     [Min, Seg, Mil] = MinSegMil(cont,handles.video.ratioFotogramas);
334
335     set(handles.minutos,'String',sprintf('%03d',Min));
336     set(handles.segundos,'String',sprintf('%02d',Seg));
337     set(handles.milesimas,'String',sprintf('%03d',Mil));
338 set(handles.textoFot,'String',sprintf('%d',cont));
339 %--- Executes during object creation, after setting all properties.
340
341 function slidervideo_CreateFcn(hObject, eventdata, handles)
342
343 % hObject    handle to slidervideo (see GCBO)
344 % eventdata  reserved - to be defined in a future version of MATLAB
345 % handles    empty - handles not created until after all CreateFcns
        called
346
347 % Hint: slider controls usually have a light gray axes1.
348 if isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
349     set(hObject,'BackgroundColor',[.9 .9 .9]);
350 end
351 %--- Executes on button press in botonPlayPause.
352 function botonPlayPause_Callback(hObject, eventdata, handles)
353 % hObject    handle to botonPlayPause (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global cont temp
    if temp == 0
        start(handles.hTimer);
        enable_reproduccion(handles,'off');
        if handles.video.numFotogramas == cont
            cont = 1;
            enable_reproduccion(handles,'on');
        end
    else
        enable_reproduccion(handles,'on');
        stop(handles.hTimer);
    end
    temp = ~temp;

%--- Executes on button press in botonFot1.

function botonFot1_Callback(hObject, eventdata, handles)
% hObject    handle to botonFot1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global cont;
  set(handles.labelFot1,'String',sprintf('Fot1: %d',cont));
    global fot_grab
    fot_grab(1) = cont;
    if prod(fot_grab)
        set(handles.Next,'enable','on');
        set(handles.labelDeltaT,'String',sprintf('Incremento de t: %d
            ms',round((fot_grab(2)-fot_grab(1))/handles.video.
            ratioFotogramas*1000)));
    end

    rgb=getimage(handles.ejesVideo);
    if isempty(rgb),return,end

    fileTypes=supportedImageTypes;
[f,p]=uiputfile(fileTypes);
if f==0,return,end
fName=fullfile(p,f);
imwrite(rgb,fName);
msgbox(['Imagen guardada en ' fName]);
    function fileTypes=supportedImageTypes
        fileTypes={'*.jpg','JPEG(*.jpg)';'*.tif','TIFF(*.tif)';'*.bmp',
            'Bitmap(*.bmp)';'*.*','All files (*.*)'};

%--- Executes on button press in botonFot2.
function botonFot2_Callback(hObject, eventdata, handles)
% hObject    handle to botonFot2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global cont;

    set(handles.labelFot2,'String',sprintf('Fot2: %d',cont));
    global fot_grab
    fot_grab(2) = cont;
```

```
    if prod(fot_grab)
        set(handles.Next,'enable','on');
        set(handles.labelDeltaT,'String',sprintf('Incremento de T: %d ms
            ',round((fot_grab(2)-fot_grab(1))/handles.video.
            ratioFotogramas*1000)));
    end

rgb=getimage(handles.ejesVideo);
if isempty(rgb),return,end
fileTypes=supportedImageTypes;
[f,p]=uiputfile(fileTypes);
if f==0,return,end
fName=fullfile(p,f);
imwrite(rgb,fName);
msgbox(['Imagen guardada en ' fName]);

        %
```

```
function enable_objects(handles)
    set(handles.minutos,'enable','on');
    set(handles.segundos,'enable','on');
    set(handles.milesimas,'enable','on');
    set(handles.textoFot,'enable','on');
    set(handles.botonPlayPause,'enable','on');
    set(handles.botonFot1,'enable','on');
    set(handles.botonFot2,'enable','on');
    set(handles.Next,'enable','on');

function enable_reproduccion(handles,valor)
    set(handles.minutos,'enable',valor);
    set(handles.segundos,'enable',valor);
    set(handles.milesimas,'enable',valor);
    set(handles.textoFot,'enable',valor);
    set(handles.cargarvideo,'enable',valor);

%--- Executes during object creation, after setting all properties.

%--- Executes on button press in Next.
function Next_Callback(hObject, eventdata, handles)
% hObject    handle to Next (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 global fot_grab

pruebaestudioimagen(handles.video.videoFotogramas(:,:,:,fot_grab(1)),
    handles.video.videoFotogramas(:,:,:,fot_grab(2)),(fot_grab(2)-
    fot_grab(1)),handles.video.ratioFotogramas);

function textoFot_Callback(hObject, eventdata, handles)
% hObject    handle to textoFot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of textoFot as text
```

```
458 %        str2double(get(hObject,'String')) returns contents of textoFot
        as a double
459 global cont;
460     cont = round(str2double(get(hObject,'String')));
461     if cont < 1
462         cont = 1;
463     elseif cont > handles.video.numFotogramas
464         cont = handles.video.numFotogramas;
465     end
466     set(handles.slidervideo,'Value',cont);
467 
468     [Min, Seg, Mil] = MinSegMil(cont,handles.video.ratioFotogramas);
469     set(handles.minutos,'String',sprintf('%03d',Min));
470     set(handles.segundos,'String',sprintf('%02d',Seg));
471     set(handles.milesimas,'String',sprintf('%03d',Mil));
472 
473 %--- Executes during object creation, after setting all properties.
474 function textoFot_CreateFcn(hObject, eventdata, handles)
475 % hObject    handle to textoFot (see GCBO)
476 % eventdata  reserved - to be defined in a future version of MATLAB
477 % handles    empty - handles not created until after all CreateFcns
        called
478 
479 % Hint: edit controls usually have a white background on Windows.
480 %       See ISPC and COMPUTER.
481 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
        defaultUicontrolBackgroundColor'))
482     set(hObject,'BackgroundColor','white');
483 end
484 
485 %--- Executes on button press in pushbutton7.
486 function pushbutton7_Callback(hObject, eventdata, handles)
487 % hObject    handle to pushbutton7 (see GCBO)
488 % eventdata  reserved - to be defined in a future version of MATLAB
489 % handles    structure with handles and user data (see GUIDATA)
490 dosimagenes
```

10.2.2. Segunda interfaz: pruebaestudioimagen.m

```
1  function varargout = pruebaestudioimagen(varargin)
2  % PRUEBAESTUDIOIMAGEN MATLAB code for pruebaestudioimagen.fig
3  %      PRUEBAESTUDIOIMAGEN, by itself, creates a new
       PRUEBAESTUDIOIMAGEN or raises the existing
4  %      singleton*.
5  %
6  %      H = PRUEBAESTUDIOIMAGEN returns the handle to a new
       PRUEBAESTUDIOIMAGEN or the handle to
7  %      the existing singleton*.
8  %
9  %      PRUEBAESTUDIOIMAGEN('CALLBACK',hObject,eventData,handles,...)
       calls the local
10 %      function named CALLBACK in PRUEBAESTUDIOIMAGEN.M with the given
       input arguments.
11 %
12 %      PRUEBAESTUDIOIMAGEN('Property','Value',...) creates a new
       PRUEBAESTUDIOIMAGEN or raises the
```

```
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before pruebaestudioimagen_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to pruebaestudioimagen_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help pruebaestudioimagen

% Last Modified by GUIDE v2.5 17-Mar-2017 15:53:13

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @pruebaestudioimagen_OpeningFcn, ...
                   'gui_OutputFcn',  @pruebaestudioimagen_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


%--- Executes just before pruebaestudioimagen is made visible.
function pruebaestudioimagen_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to pruebaestudioimagen (see VARARGIN)
 set(handles.figureEstudioimagenes,'CurrentAxes',handles.axes1);
    handles.hImage01 = varargin{1};
    handles.hImage1=image(varargin{1});
   [m,n,p]=size(handles.hImage01);
set(handles.numancho1,'String',n);
set(handles.numalto1,'String',m);
```

```
60      axis off;
61      axis image;
62       set(handles.figureEstudioimagenes,'CurrentAxes',handles.axes2);
63      handles.hImage02 = varargin{2};
64     handles.hImage2=image(varargin{2});
65      axis off;
66      axis image;
67
68      handles.datos.delta_fot = varargin{3};
69      handles.datos.fps = varargin{4};
70      handles.datos.color1 = false;
71      handles.datos.color2 = false;
72
73      set(handles.textoFPS,'String',num2str(handles.datos.fps));
74      set(handles.deltaT,'String',num2str(sprintf('%d s',(handles.datos.
            delta_fot)/handles.datos.fps)));
75
76      handles.datos.vista_completa_X1 = get(handles.axes1,'XLim');
77      handles.datos.vista_completa_Y1 = get(handles.axes1,'YLim');
78      handles.datos.vista_completa_X2 = get(handles.axes2,'XLim');
79      handles.datos.vista_completa_Y2 = get(handles.axes2,'YLim');
80
81
82  [a,map]=imread('next.jpg');
83  [r,c,d]=size(a);
84  x=ceil(r/100);
85  y=ceil(c/50);
86  g=a(1:x:end,1:y:end,:);
87  g(g==255)=5.5*255;
88  set(handles.next,'CData',g);
89
90  % Choose default command line output for pruebaestudioimagen
91  handles.output = hObject;
92
93  % Update handles structure
94  guidata(hObject, handles);
95
96  % UIWAIT makes pruebaestudioimagen wait for user response (see UIRESUME
        )
97  % uiwait(handles.figureEstudioimagenes);
98
99  %--- Outputs from this function are returned to the command line.
100 function varargout = pruebaestudioimagen_OutputFcn(hObject, eventdata,
        handles)
101 % varargout  cell array for returning output args (see VARARGOUT);
102 % hObject    handle to figure
103 % eventdata  reserved - to be defined in a future version of MATLAB
104 % handles    structure with handles and user data (see GUIDATA)
105
106 % Get default command line output from handles structure
107 varargout{1} = handles.output;
108
109 %--- Executes on button press in blanconegro1.
110 function blanconegro1_Callback(hObject, eventdata, handles)
111 % hObject    handle to blanconegro1 (see GCBO)
112 % eventdata  reserved - to be defined in a future version of MATLAB
113 % handles    structure with handles and user data (see GUIDATA)
```

```
imagen=getimage(handles.axes1);
 if size(imagen,3)==3
                    grises = rgb2gray(imagen);
 else
                    grises = imagen;
 end

axes(handles.axes1);
axis off;
imshow(grises),guidata(hObject,handles);

%--- Executes on button press in blanconegro2.
function blanconegro2_Callback(hObject, eventdata, handles)
% hObject    handle to blanconegro2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
imagen=getimage(handles.axes2);

 if size(imagen,3)==3
                    grises = rgb2gray(imagen);
 else
                    grises = imagen;
 end

axes(handles.axes2);
axis off;
imshow(grises),guidata(hObject,handles);

function xmin1_Callback(hObject, eventdata, handles)
% hObject    handle to xmin1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of xmin1 as text
%        str2double(get(hObject,'String')) returns contents of xmin1 as
    a double
Val=get(hObject,'String');
NewVal=str2double(Val);
handles.xmin1=NewVal;
guidata(hObject,handles);

%--- Executes during object creation, after setting all properties.
function xmin1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to xmin1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
   called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function ymin1_Callback(hObject, eventdata, handles)
% hObject    handle to ymin1 (see GCBO)
```

```
168 %eventdata  reserved - to be defined in a future version of MATLAB
169 %handles    structure with handles and user data (see GUIDATA)
170
171 %Hints: get(hObject,'String') returns contents of ymin1 as text
172 %        str2double(get(hObject,'String')) returns contents of ymin1 as
        a double
173 Val=get(hObject,'String');
174 NewVal=str2double(Val);
175 handles.ymin1=NewVal;
176 guidata(hObject,handles);
177
178 %--- Executes during object creation, after setting all properties.
179 function ymin1_CreateFcn(hObject, eventdata, handles)
180 %hObject    handle to ymin1 (see GCBO)
181 %eventdata  reserved - to be defined in a future version of MATLAB
182 %handles    empty - handles not created until after all CreateFcns
        called
183
184 %Hint: edit controls usually have a white background on Windows.
185 %      See ISPC and COMPUTER.
186 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
        defaultUicontrolBackgroundColor'))
187     set(hObject,'BackgroundColor','white');
188 end
189
190 function anchura1_Callback(hObject, eventdata, handles)
191 %hObject    handle to anchura1 (see GCBO)
192 %eventdata  reserved - to be defined in a future version of MATLAB
193 %handles    structure with handles and user data (see GUIDATA)
194
195 %Hints: get(hObject,'String') returns contents of anchura1 as text
196 %        str2double(get(hObject,'String')) returns contents of anchura1
        as a double
197 Val=get(hObject,'String');
198 NewVal=str2double(Val);
199 handles.anchura1=NewVal;
200 guidata(hObject,handles);
201
202 %--- Executes during object creation, after setting all properties.
203 function anchura1_CreateFcn(hObject, eventdata, handles)
204 %hObject    handle to anchura1 (see GCBO)
205 %eventdata  reserved - to be defined in a future version of MATLAB
206 %handles    empty - handles not created until after all CreateFcns
        called
207
208 %Hint: edit controls usually have a white background on Windows.
209 %      See ISPC and COMPUTER.
210 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
        defaultUicontrolBackgroundColor'))
211     set(hObject,'BackgroundColor','white');
212 end
213
214 function altura1_Callback(hObject, eventdata, handles)
215 %hObject    handle to altura1 (see GCBO)
216 %eventdata  reserved - to be defined in a future version of MATLAB
217 %handles    structure with handles and user data (see GUIDATA)
218
```

```
219 % Hints: get(hObject,'String') returns contents of altura1 as text
220 %        str2double(get(hObject,'String')) returns contents of altura1
        as a double
221 Val=get(hObject,'String');
222 NewVal=str2double(Val);
223 handles.altura1=NewVal;
224 guidata(hObject,handles);
225
226 %--- Executes during object creation, after setting all properties.
227 function altura1_CreateFcn(hObject, eventdata, handles)
228 % hObject    handle to altura1 (see GCBO)
229 % eventdata  reserved - to be defined in a future version of MATLAB
230 % handles    empty - handles not created until after all CreateFcns
        called
231
232 % Hint: edit controls usually have a white background on Windows.
233 %       See ISPC and COMPUTER.
234 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
        defaultUicontrolBackgroundColor'))
235     set(hObject,'BackgroundColor','white');
236 end
237
238 %--- Executes on button press in recorte1.
239 function recorte1_Callback(hObject, eventdata, handles)
240 % hObject    handle to recorte1 (see GCBO)
241 % eventdata  reserved - to be defined in a future version of MATLAB
242 % handles    structure with handles and user data (see GUIDATA)
243 xm=handles.xmin1;
244 ym=handles.ymin1;
245 anch=handles.anchura1;
246 alt=handles.altura1;
247
248 set(handles.figureEstudioimagenes,'CurrentAxes',handles.axes3);
249 imagen=getimage(handles.axes1);
250 imshow(imagen);
251 handles.h1rect=rectangle('Position',[xm ym anch alt],'LineWidth',2,'
        EdgeColor','w');
252
253 set(handles.numancho2,'String',anch);
254 set(handles.numalto2,'String',alt);
255
256 guidata(hObject,handles);
257
258 %--- Executes on button press in recorte2.
259 function recorte2_Callback(hObject, eventdata, handles)
260 % hObject    handle to recorte2 (see GCBO)
261 % eventdata  reserved - to be defined in a future version of MATLAB
262 % handles    structure with handles and user data (see GUIDATA)
263 xm=handles.xmin1;
264 ym=handles.ymin1;
265 anch=handles.anchura1;
266 alt=handles.altura1;
267
268 set(handles.figureEstudioimagenes,'CurrentAxes',handles.axes4);
269 imagen=getimage(handles.axes2);
270 imshow(imagen);
```

```
handles.h2rect=rectangle('Position',[xm ym anch alt],'LineWidth',2,'
    EdgeColor','w');


guidata(hObject,handles);

%--- Executes on button press in outruido1.
function outruido1_Callback(hObject, eventdata, handles)
% hObject    handle to outruido1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
imagen1=getimage(handles.axes1);
imagen12=rgb2gray(imagen1);
imagen13=imnoise(imagen12,'salt & pepper',0.1);

f_media=medfilt2(imagen13);

axes(handles.axes1);
axis off;
imshow((f_media)),guidata(hObject,handles);
%explicar el tipo de filtro.

%--- Executes on button press in outruido2.
function outruido2_Callback(hObject, eventdata, handles)
% hObject    handle to outruido2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
imagen1=getimage(handles.axes2);
imagen12=rgb2gray(imagen1);
imagen13=imnoise(imagen12,'salt & pepper',0.1);


f_media=medfilt2(imagen13);

axes(handles.axes2);
axis off;
imshow(uint8(f_media)),guidata(hObject,handles);

%--- Executes during object creation, after setting all properties.

%--- Executes on button press in next.
function next_Callback(hObject, eventdata, handles)
% hObject    handle to next (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

rect1=get(handles.h1rect,'Position');
rect1_act=~isequal(rect1,[0 0 1 1]);
rect2=get(handles.h2rect,'Position');
rect2_act=~isequal(rect1,[0 0 1 1]);

patron=getimage(handles.axes1);
imagen=getimage(handles.axes2);
px2mm = 1;
tam1 = size(patron);
tam2 = size(imagen);
fps = str2double(get(handles.textoFPS,'String'));
```

```
num=str2double(get(handles.NumVentana,'String'));

r1=floor(rect1);
r2=floor(rect2);
offset_patron=[0 0];
offset_imagen=[0 0];
if rect1_act
    patron=patron(r1(2):r1(2)+r1(4),r1(1):r1(1)+r1(3));
    offset_patron=[r1(1) r1(2)];

end
if rect2_act
    imagen=imagen(r2(2):r2(2)+r2(4),r2(1):r2(1)+r2(3));
    offset_imagen=[r2(1) r2(2)];
end
if rect1_act
                    ancho = floor(rect1(3)/num);
                      alto = floor(rect1(4)/num);
else

                      ancho = floor(tam1(2)/num);
                      alto = floor(tam1(1)/num);
end

[U, V, X, Y] = corr(patron,imagen,[ancho, alto],offset_patron,
    offset_imagen);

Resultados(patron,X,Y,U,V,ancho,alto,px2mm/1000/(handles.datos.
    delta_fot/fps));

function textoTamPixel_Callback(hObject, eventdata, handles)
% hObject    handle to textoTamPixel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of textoTamPixel as
    text
%        str2double(get(hObject,'String')) returns contents of
    textoTamPixel as a double

%--- Executes during object creation, after setting all properties.
function textoTamPixel_CreateFcn(hObject, eventdata, handles)
% hObject    handle to textoTamPixel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
    called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function NumVentana_Callback(hObject, eventdata, handles)
% hObject    handle to NumVentana (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of NumVentana as text
%        str2double(get(hObject,'String')) returns contents of NumVentana as a double

%--- Executes during object creation, after setting all properties.
function NumVentana_CreateFcn(hObject, eventdata, handles)
% hObject    handle to NumVentana (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

10.2.3. Tercera interfaz: Resultados.m

```
function varargout = Resultados(varargin)
%RESULTADOS MATLAB code for Resultados.fig
%      RESULTADOS, by itself, creates a new RESULTADOS or raises the existing
%      singleton*.
%
%      H = RESULTADOS returns the handle to a new RESULTADOS or the handle to
%      the existing singleton*.
%
%      RESULTADOS('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in RESULTADOS.M with the given input arguments.
%
%      RESULTADOS('Property','Value',...) creates a new RESULTADOS or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Resultados_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Resultados_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Resultados

% Last Modified by GUIDE v2.5 17-Mar-2017 15:50:28
```

```
26
27 %Begin initialization code - DO NOT EDIT
28 gui_Singleton = 1;
29 gui_State = struct('gui_Name',       mfilename, ...
30                   'gui_Singleton',  gui_Singleton, ...
31                   'gui_OpeningFcn', @Resultados_OpeningFcn, ...
32                   'gui_OutputFcn',  @Resultados_OutputFcn, ...
33                   'gui_LayoutFcn',  [] , ...
34                   'gui_Callback',   []);
35 if nargin && ischar(varargin{1})
36     gui_State.gui_Callback = str2func(varargin{1});
37 end
38
39 if nargout
40     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
41 else
42     gui_mainfcn(gui_State, varargin{:});
43 end
44 %End initialization code - DO NOT EDIT
45
46
47 %--- Executes just before Resultados is made visible.
48 function Resultados_OpeningFcn(hObject, eventdata, handles, varargin)
49
50     handles.datos.patron = varargin{1};
51     handles.datos.X = varargin{2};
52     handles.datos.Y = varargin{3};
53     handles.datos.U = varargin{4};
54     handles.datos.V = varargin{5};
55     handles.datos.ancho = varargin{6};
56     handles.datos.alto  = varargin{7};
57     cte = varargin{8};
58
59     tam_patron = size(handles.datos.patron);
60
61     %Calcular los campos escalares
62     ancho = handles.datos.ancho;
63     alto  = handles.datos.alto;
64
65     U_tot = zeros(tam_patron(1),tam_patron(2));
66     V_tot = zeros(tam_patron(1),tam_patron(2));
67
68     numY = floor(tam_patron(2)/ancho);
69     numX = floor(tam_patron(1)/alto);
70
71     for i=1:numY
72         for j=1:numX
73             U_tot(alto*(j-1)+1:alto*j,ancho*(i-1)+1:ancho*i) = handles.
                datos.U(j,i)*ones(alto,ancho)*cte;
74             V_tot(alto*(j-1)+1:alto*j,ancho*(i-1)+1:ancho*i) = handles.
                datos.V(j,i)*ones(alto,ancho)*cte;
75         end
76     end
77     Mag = sqrt(U_tot.^2 + V_tot.^2);
78
79     %Tomar la posicion central de los ejes y la minima anchura, altura
80
```

```
81      pos = get(handles.ejesPatron,'Position');
82      ancho_max = pos(3);
83      alto_max = pos(4);
84      pos_central = [pos(1)+pos(3)/2, pos(2)+pos(4)/2];
85
86      prop_grafica = ancho_max/alto_max; % Proporcion
87      prop_patron  = tam_patron(2)/tam_patron(1);
88
89      if prop_patron > prop_grafica %Domina ancho -> Bajar alto
90          pos(4) = ancho_max/prop_patron;
91          pos(2) = pos_central(2) - pos(4)/2;
92      else %Domina alto -> Bajar ancho
93          pos(3) = alto_max*prop_patron;
94          pos(1) = pos_central(1) - pos(3)/2;
95      end
96
97      handles.hImagen = imshow(handles.datos.patron,'Parent',handles.
            ejesPatron);
98      set(handles.ejesPatron,'XLim',[0.5 tam_patron(2)+0.45],'YLim',[0.5
            tam_patron(1)+0.45],'Position',pos,'box','on','tickdir','out','
            nextplot','add','ydir','reverse','XTick',[],'YTick',[],'
            XLimMode','manual','YLimMode','manual');
99      colormap(gray);
100     handles.hVect = quiver(handles.datos.X,handles.datos.Y,handles.
            datos.U,handles.datos.V,'Parent',handles.ejesPatron,'Color','y'
            ,'LineWidth',1.3,'AutoScaleFactor',1);
101     maximo = max(max(Mag));
102     minimo = min([min(min(U_tot)), min(min(V_tot))]);
103     handles.datos.V_color = escala_colores((Mag    - minimo)./(maximo -
            minimo));
104     handles.datos.Vx_color  = escala_colores((U_tot  - minimo)./(maximo
            - minimo));
105     handles.datos.Vy_color  = escala_colores((V_tot  - minimo)./(maximo
            - minimo));
106
107     %Definir escala colores
108     marcas{1} = sprintf('%.2f',minimo);
109     marcas{2} = sprintf('%.2f',minimo + (maximo-minimo)/4);
110     marcas{3} = sprintf('%.2f',(maximo + minimo)/2);
111     marcas{4} = sprintf('%.2f',maximo - (maximo-minimo)/4);
112     marcas{5} = sprintf('%.2f',maximo);
113     set(handles.ejesColores,'YLim',[0 1],'XLim',[0 1],'nextplot','add',
            'YTick',[0 0.25 0.5 0.75 1],'YTickLabel',marcas,'XTick',[],'
            YAxisLocation','right')
114     n_colores = 100;
115     alto = 1/n_colores;
116     for i=1:n_colores
117         t = alto*(i-1) + alto/2;
118         patch([0 1 1 0],[alto*(i-1) alto*(i-1) alto*i alto*i],
                escala_colores(t),'Parent',handles.ejesColores,'LineStyle',
                'none');
119     end
120     set(hObject,'CurrentAxes',handles.ejesPatron);
121
122     handles.datos.Mag = Mag;
123     handles.datos.U_tot = U_tot;
124     handles.datos.V_tot = V_tot;
```

```
125
126 % Choose default command line output for Resultados
127 handles.output = hObject;
128
129 % Update handles structure
130 guidata(hObject, handles);
131
132 % UIWAIT makes Resultados wait for user response (see UIRESUME)
133 % uiwait(handles.figureResultados);
134
135 %--- Outputs from this function are returned to the command line.
136 function varargout = Resultados_OutputFcn(hObject, eventdata, handles)
137 % varargout  cell array for returning output args (see VARARGOUT);
138 % hObject    handle to figure
139 % eventdata  reserved - to be defined in a future version of MATLAB
140 % handles    structure with handles and user data (see GUIDATA)
141
142 % Get default command line output from handles structure
143 varargout{1} = handles.output;
144
145 %--- Executes on button press in checkMostrarCampo.
146 function checkMostrarCampo_Callback(hObject, eventdata, handles)
147 if get(hObject,'Value')
148     set(handles.panelCampo,'visible','on')
149     if get(handles.radioEscalar,'Value')
150         set(handles.hImagen,'visible','on');
151         set(handles.panelCampoEscalar,'visible','on');
152         set(handles.checkMostrarImagen,'enable','off');
153         Vx = get(handles.radioVx,'value');
154         Vy = get(handles.radioVy,'value');
155         V  = get(handles.radioV,'value');
156         switch(1)
157             case Vx
158                 set(handles.hImagen,'CData',handles.datos.Vx_color);
159             case Vy
160                 set(handles.hImagen,'CData',handles.datos.Vy_color);
161             case V
162                 set(handles.hImagen,'CData',handles.datos.V_color);
163         end
164     else
165         set(handles.hVect,'visible','on');
166         if get(handles.checkMostrarImagen,'value')
167             set(handles.hImagen,'CData',handles.datos.patron);
168             set(handles.hImagen,'visible','on');
169         else
170             set(handles.hImagen,'visible','off');
171         end
172     end
173 else
174     set(handles.checkMostrarImagen,'enable','on');
175     if get(handles.checkMostrarImagen,'value')
176         set(handles.hImagen,'CData',handles.datos.patron);
177         set(handles.hImagen,'visible','on');
178     else
179         set(handles.hImagen,'visible','off');
180     end
181     set(handles.panelCampo,'visible','off');
```

```
182     set(handles.panelCampoEscalar,'visible','off');
183     set(handles.hVect,'visible','off');
184
185 end
186
187
188 function checkMostrarImagen_Callback(hObject, eventdata, handles)
189 if get(hObject,'Value')
190     set(handles.hImagen,'visible','on');
191 else
192     set(handles.hImagen,'visible','off');
193 end
194
195 %--- Executes when selected object is changed in panelCampo.
196 function panelCampo_SelectionChangeFcn(hObject, eventdata, handles)
197 % hObject    handle to the selected object in panelCampo
198 % eventdata  structure with the following fields (see UIBUTTONGROUP)
199 %       EventName: string 'SelectionChanged' (read only)
200 %       OldValue: handle of the previously selected object or empty if
    none was selected
201 %       NewValue: handle of the currently selected object
202 % handles    structure with handles and user data (see GUIDATA)
203 if isequal(eventdata.NewValue,handles.radioVectorial)
204     set(handles.hVect,'visible','on');
205     set(handles.checkMostrarImagen,'enable','on');
206     set(handles.panelCampoEscalar,'visible','off');
207     set(handles.hImagen,'CData',handles.datos.patron);
208     if get(handles.checkMostrarImagen,'value');
209         set(handles.hImagen,'Visible','on');
210     else
211         set(handles.hImagen,'Visible','off');
212     end
213 else
214     set(handles.hVect,'visible','off');
215     set(handles.checkMostrarImagen,'enable','off');
216     set(handles.panelCampoEscalar,'visible','on');
217     Vx = get(handles.radioVx,'value');
218     Vy = get(handles.radioVy,'value');
219     V  = get(handles.radioV,'value');
220     switch(1)
221         case Vx
222             set(handles.hImagen,'CData',handles.datos.Vx_color);
223         case Vy
224             set(handles.hImagen,'CData',handles.datos.Vy_color);
225         case V
226             set(handles.hImagen,'CData',handles.datos.V_color);
227     end
228     set(handles.hImagen,'visible','on');
229 end
230
231 %--- Executes when selected object is changed in panelCampoEscalar.
232 function panelCampoEscalar_SelectionChangeFcn(hObject, eventdata,
    handles)
233 % hObject    handle to the selected object in panelCampoEscalar
234 % eventdata  structure with the following fields (see UIBUTTONGROUP)
235 %       EventName: string 'SelectionChanged' (read only)
```

```
%       OldValue: handle of the previously selected object or empty if none was selected
%       NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)
switch(eventdata.NewValue)
    case handles.radioVx
        set(handles.hImagen,'CData',handles.datos.Vx_color);
    case handles.radioVy
        set(handles.hImagen,'CData',handles.datos.Vy_color);
    case handles.radioV
        set(handles.hImagen,'CData',handles.datos.V_color);
end

%--- Executes on mouse motion over figure - except title and menu.
function figureResultados_WindowButtonMotionFcn(hObject, eventdata, handles)
        handles.datos.pos_raton = get(hObject,'CurrentPoint'); %La posicion del ratio en la figura
        axes_area = get(handles.ejesPatron,'Position'); %El area que abarca los ejes
        if (handles.datos.pos_raton(1,1) < (axes_area(1) + axes_area(3))) && ...
                (handles.datos.pos_raton(1,1) > (axes_area(1))) && ...
                (handles.datos.pos_raton(1,2) < (axes_area(2) + axes_area(4))) && ...
                (handles.datos.pos_raton(1,2) > (axes_area(2))) %En ejes

            pos_raton = get(handles.ejesPatron,'CurrentPoint'); %Posicion del ratio en la grafica
            j = round(pos_raton(1,1));
            i = round(pos_raton(1,2));
            if i > size(handles.datos.Mag,1)
                i = i -1;
            end
            if j > size(handles.datos.Mag,2)
                j = j -1;
            end
            set(handles.labelVelocidad,'String',sprintf('V = %.3f m/s\nVx = %.3f m/s\nVy = %.3f m/s',handles.datos.Mag(i,j),handles.datos.U_tot(i,j),handles.datos.V_tot(i,j)));
        else
            set(handles.labelVelocidad,'String','');
        end

function botonVistaCompleta_Callback(hObject, eventdata, handles)
    figure('Numbertitle','off','Name',sprintf('Campo de velocidades - Ventana de %d x %d pixels',handles.datos.ancho,handles.datos.alto),'Color','w');
    ejesPantCompl = axes;
    imshow(get(handles.hImagen,'CData'));
    if get(handles.radioVectorial,'Value')
        hold on;
        copyobj(handles.hVect,ejesPantCompl);
    end
    set(0,'CurrentFigure',handles.figureResultados);

```

```
281     %--- Executes on button press in guardarimagen.
282 function guardarimagen_Callback(hObject, eventdata, handles)
283 % hObject    handle to guardarimagen (see GCBO)
284 % eventdata  reserved - to be defined in a future version of MATLAB
285 % handles    structure with handles and user data (see GUIDATA)
286 rgb=getframe(handles.ejesPatron);
287     if isempty(rgb)
288         return
289     end
290 
291 fileTypes=supportedImageTypes;
292 [f,p]=uiputfile(fileTypes);
293 
294     if f==0
295         return
296     end
297 
298 fName=fullfile(p,f);
299 imwrite(rgb.cdata,fName);
300 msgbox(['Imagen guardada en ' fName]);
301 
302 function fileTypes=supportedImageTypes
303 
304 fileTypes={'*.jpg','JPEG(*.jpg)';'*.tif','TIFF(*.tif)';'*.bmp','Bitmap
    (*.bmp)';'*.*','All files (*.*)'};
305 
306 %--- Executes on button press in creagif.
307 function creagif_Callback(hObject, eventdata, handles)
308 % hObject    handle to creagif (see GCBO)
309 % eventdata  reserved - to be defined in a future version of MATLAB
310 % handles    structure with handles and user data (see GUIDATA)
311 for i=1:3
312     a=strcat('gif',num2str(i), '.jpg');
313     img=imread(a);
314     b(i)=im2frame(img);
315 end
316 movie(b,10,2) %reproduce b, 5 veces, 10 cuadrados por segundo
317 set(handles.figureResultados,'CurrentAxes',handles.ejesPatron)
318 axis off;
```

10.3. Anexo 3 (Guía del usuario)

10.3.1. Introducción

Objetivos

- Proporcionar un entendimiento de la técnica PIV ('Velocimetría de Imágenes de Partículas') de una manera sencilla y dinámica.
- Conocer y comprobar las características principales (resolución espacial, temporal, rango dinámico...)
- Verificar de una manera aproximada la velocidad y dirección de un fluido.

Funcionamiento

El método PIV permite medir la velocidad de un fluido en una región 2D iluminada, dicha región se forma por el corte de una hoja de luz láser en el flujo.

Es necesario que el flujo contenga partículas de dopado, pues se va registrar el movimiento de dichas partículas utilizando técnicas de múltiple exposición, es decir, se toman fotos consecutivas de lo que se va viendo para trazar la trayectoria. Si sabemos el desplazamiento de un grupo de partículas (vector desplazamiento) y el tiempo que ha pasado entre las fotos, sabremos la velocidad (vector velocidad).

Una vez tenemos las fotos (2 por ejemplo), se procesan y mediante algoritmos de correlación determinamos dónde se corresponde esa región del espacio con ese fotograma y a partir de ahí se obtiene el vector velocidad.

Dependiendo del tamaño de la partícula, se dispersará más o menos luz, suponemos como tamaño característico 10 μm para partículas grandes.

¿Cómo se procesan los datos?

Se busca un cierto patrón, el cual se busca en la 2ª imagen, esto se realiza definiendo la ventana sobre la que vamos a trabajar (región de interrogación). Si vemos todas las partículas desplazadas en una dirección, el patrón se habrá movido en bloque. Por lo tanto se puede afirmar que el vector velocidad va a ser el espacio que se haya desplazado ese patrón dividido por el tiempo entre los 2 disparos.

¿Qué es la función de correlación?

La mejor forma de realizar el cálculo (identificar los patrones) es utilizando la función de correlación cruzada.

Si tenemos una función $g_1(x, y)$ (cualquiera), definimos la correlación como la multiplicación entre $g_1(x, y)$ y otra función $g_2(x, y)$ desplazada una distancia que decidamos, se suman los valores y obtenemos un dato.

$$\Phi = g_1(x,y)g_2(x,y) = \sum_{i=-M}^{M} \sum_{j=-N}^{N} g_1(i,j)g_2(i+x, j+y) \tag{10.7}$$

para $x = 0, \pm 1, \pm 2 \ldots \pm N - 1 \quad y = 0, \pm 1, \pm 2 \ldots \pm M - 1$, donde $g_1(x, y)$ es la subimagen correspondiente a la primera zona de interrogación, $g_2(x, y)$ es la 2ª zona, M es el ancho y N la altura en píxels de la zona de interrogación.

Para valores de desplazamiento en los que las imágenes de las partículas estén alineadas unas con otras, el valor de será máximo.

Consejos a tener en cuenta cuando se utiliza la correlación:

- Normalmente se desprecia la contribución del ruido, pero no siempre es cierto, en algunos casos puede ser importante y aparecen muchos puntitos adicionales en la foto. Por ese motivo el programa es capaz de aplicar un filtro Gaussiano para reducir el ruido de aquellas fotos que presenten un alto nivel del mismo.
- Estamos asumiendo que el patrón se mueve en bloque, pero hemos de tener en cuenta que cuando hablamos de flujo es mentira. Cada partícula tiene su trayectoria.
- Las partículas deben tener un tamaño de entre 3 y 5 píxeles de mi imagen, es decir, que cuando tomemos la foto, se vea que la partícula ocupa entre 3 y 5 píxeles.
- Es importante que en nuestra región de interrogación se vean al menos de entre 5 y 15 partículas.
- Para unos resultados coherentes debemos asegurarnos de que le desplazamiento máximo del patrón a buscar en la 2º foto sea como máximo del 25 % de la región de interrogación.

10.3.2. Descripción de la interfaz

1ª ventana: cargarvideopiv.m

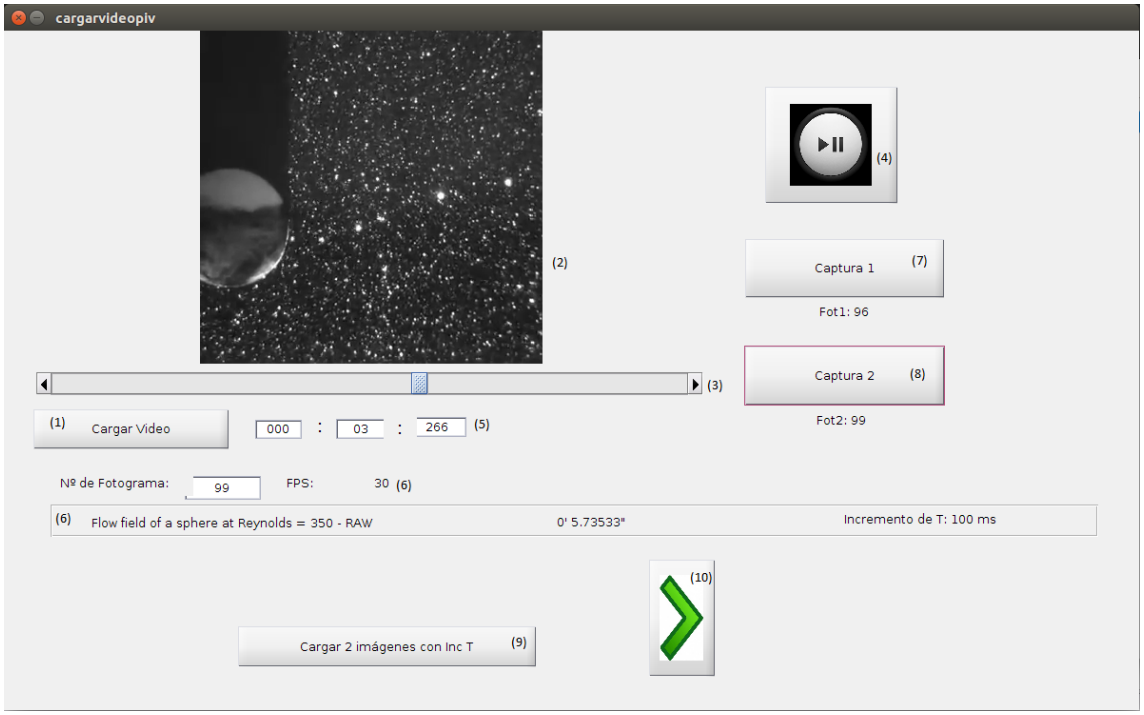


Figura 10.1: Interfaz Cargarvideopiv.m

- **(1)** Cargar video: Se pulsa para cargar arhivos .avi.
- **(2)** Ventana del vídeo: Donde vemos la reproducción del vídeo.
- **(3)** Barra Slider: Arrastrando podemos ver el instante que deseamos.
- **(4)** Play Pause: Comienza la reproducción y la detiene en cualquier instante.
- **(5)** Temporizador: Minutos : Segundos : Milésimas.

- **(6)** Panel información: Nombre y formato del vídeo, Duración total y Δt (ms) entre fotogramas. Arriba se indian los FPS (fotogramas por segundo) del vídeo.
- **(7)** Captura 1: Captura y archiva la primera foto en cualquier formato, india el fotograma correspondiente.
- **(8)** Captura 2: Captura y archiva la segunda foto en cualquier formato y habilita el Δt, india el fotograma correspondiente.
- **(9)** Cargar 2 imágenes con Δt: Abre el archivo dosimágenes.m para realizar el estudio con 2 imágenes que ya tenemos.
- **(10)** Siguiente ventana: Abre el archivo pruebaestudioimgen.m y continúa el proceso con nuestras 2 capturas.

2ª ventana: pruebaestudioimagen.m

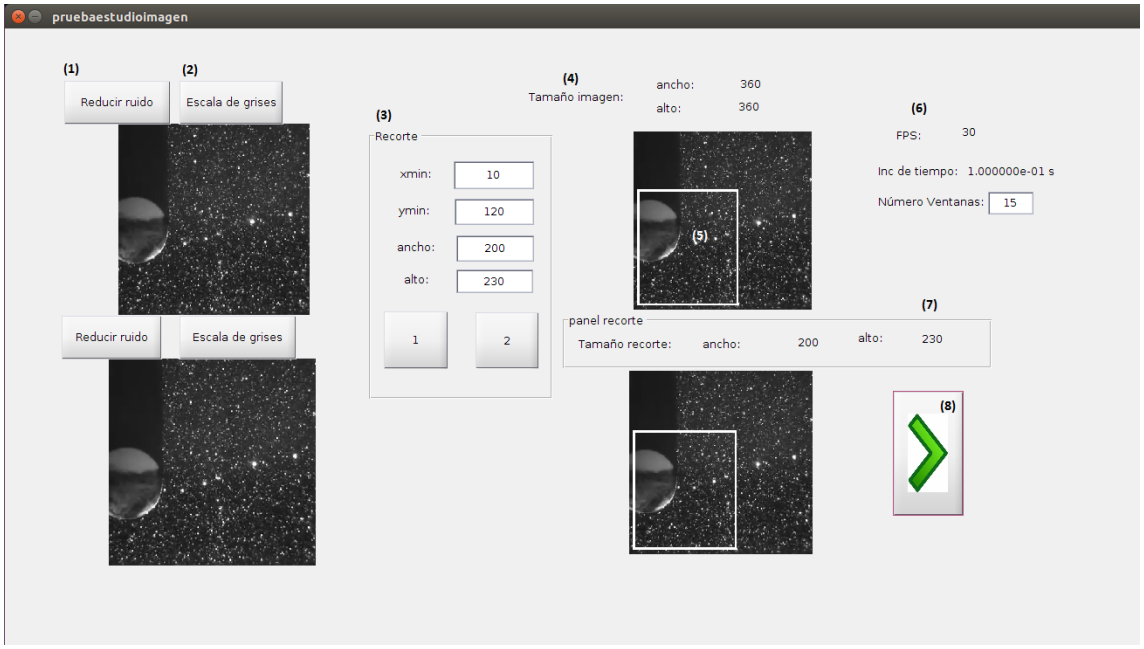


Figura 10.2: Interfaz Pruebaestudioimagen.m

- **(1)** Reducir ruido: (opcional) Aplica filtro Gaussiano para reducir el ruido de la imagen (si lo hubiera).
- **(2)** Blano y negro: (opiconal) Pasa nuestra imagen a escala de grises (si estuviera a color).
- **(3)** Panel recorte: Indicamos Posición (x, y) y tamaño $ancho - alto$ de nuestro rectángulo recortado.
- **(4)** Tamaño imágenes: India el tamaño real de la imagen en píxeles.
- **(5)** Rectángulo recorte: Muestra donde hemos ubicado el rectángulo.
- **(6)** Información: FPS , Δt (s) y selección del nº de ventanas en las que se dividirá el rectángulo.
- **(7)** Tamaño rectángulo recorte: Indica el tamaño real del recorte en píxeles.
- **(8)** Siguiente ventana (correlación): Abre el archivo Resultados.m después de realizar la correlación y observamos los resultados.

3ª ventana: Resultados.m

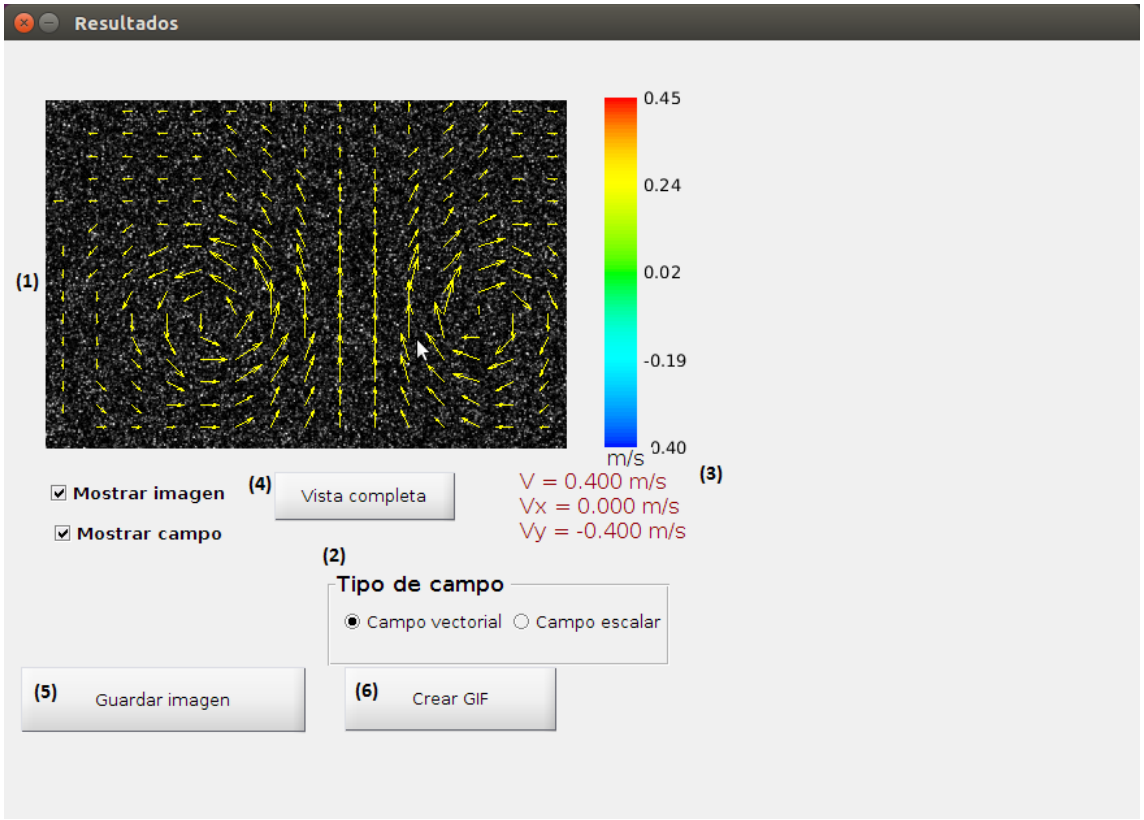


Figura 10.3: Interfaz Resultados.m

- **(1)** Ventana resultados: Es la ventana donde observamos el campo de vectores o el campo escalar.
- **(2)** Selección del tipo de campo: Campo escalar o vectorial.
- **(3)** Panel velocidades: Panel que muestra los valores de $V(total)(m/s)$, $Vx(m/s)$ y $Vy(m/s)$ de la zona de la ventana resultados donde pasamos por encima.
- **(4)** Vista completa: Reproduce cualquier ventana imagen en otra ventana aparte.
- **(5)** Guardar la imagen de resultados: Salva la imagen en cualquier formato y con el nombre que deseemos.
- **(6)** Crear Gif: Lee 3 imagenes con nombre gif(i). Siendo $i = 1, 2, 3$. Y las reproduce viendo la evolción de cualquier tipo de campo.

10.3.3. Paso a paso

Primera parte:

En primer lugar abrimos la carpeta donde se encuentran todos los archivos (Software). Una vez abierta localizamos los Scripts de Matlab: **cargarvideopiv.m, pruebaestudioimagenes.m, Resultados.m**. De los cuales únicamente abriremos directamente desde la carpeta el archivo **cargarvdideopiv.m**. Click dos veces en el archivo y se abrirá la ventana correspondiente a la figura (1) y comienza el proceso:

- Pulsamos el botón 'cargar vídeo' y nos aparecerá una ventana en la cual tendremos que seleccionar el vídeo que queremos analizar. Una vez cargue tendremos encima del 'Slider' la ventana donde se mostrará.

- Pulsamos 'Play Pause' y el vídeo empezará a correr, entonces esperamos a observar un grupo de partículas con una cierta tendencia y pausamos. Acto seguido obtenemos la primera imagen con 'Captura 1' y volvemos a darle al 'Play Pause', esperamos un breve instante de tiempo y pausamos, capturando ahora la segunda imagen con 'Captura 2'. Si no hay suficiente precisión para pausar, podemos editar lo que avanzamos con la barra del Slider y fijándonos en el número de fotogramas.
- Una vez hayamos pulsado 'Captura 2' nos aparecerá el $\Delta t\,(m/s)$.
- Cada vez que pulsemos los botones 'Captura' tendremos que guardar las imágenes en el formato que deseemos.
- Si lo que queremos es calcular las velocidades que nos ofrecen 2 imágenes, pulsamos 'Cargar 2 imágenes con Δt'.
- Ya sea 'Cargar 2 imágenes con Δt' o pulsando el botón 'Siguiente' (flechita verde) nos aparecerá automáticamente la siguiente interfaz **pruebaestudioimagenes.m**.

Segunda parte:

A continuación veremos automáticamente en esta segunda interfaz como nos aparecen las 2 imágenes que hemos seleccionado anteriormente y continuamos el proceso:

- En el caso de que nuestras imágenes tengan ruido, pulsaremos 'Quitar ruido' y se aplicará el filtro Gaussiano.
- En el caso de que nuestras imágenes estén en color, pulsaremos 'Escala de grises' y las imágenes se verán en blanco y negro, aportando mayor contraste en la visualización de las partículas.
- Una vez tratada la imagen, nos vamos al 'Panel recorte', donde escribiremos los datos: **xmin, ymin, ancho, alto** significando lo que se observa en la figura (5).

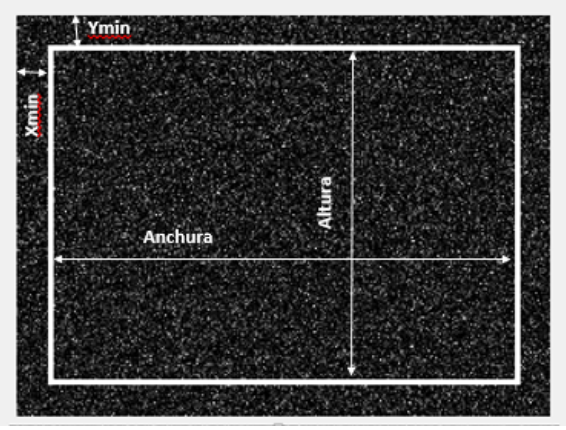


Figura 10.4: Selección del rectángulo

- Una vez escritos los datos anteriores pulsamos '1' y '2' respectivamente para observar el rectángulo que hemos seleccionado de la imagen y obteniendo nuestro patrón. Estos 2 rectángulos los veremos en la parte derecha de la interfaz.
- Para continuar pulsamos el botón 'Siguiente' (flechita verde) y se realizará la correlación cruzada. Abriéndose automáticamente la interfaz **Resultados.m**.

Tercera parte:

Para terminar utilizaremos la última ventana, en la cual nos aparecerá la ventana de resultados (campo de vectores, velocidades...) y terminamos el proceso:

- Primero nos detenemos a observar la ventana del campo de vectores que nos aparece. Si deslizamos el puntero del ratón por encima de la ventana veremos los datos de velocidades en cada región de la imagen.
- Si queremos ver estas velocidades en una magnitud escalar, seleccionamos 'Campo escalar' y nos fijamos en los valores de la barra de colores a la derecha de la imagen.
- Para ver una cierta evolución, podemos guardar la imagen de cualquiera de los Campos llamándole gif(i), para $i = 1, 2, 3$. Una vez hayamos guardados 3 imágenes (correspondientes a 3 cálculos), pulsamos 'Crear Gif' y veremos en la ventana de resultados nuestra imagen en movimiento.

Nota: para realizar más de un cálculo, es necesario cerrar las interfaces **Resultados.m** y **pruebaestudioimagen.m**.

10.3.4. Conclusiones

Una vez seguido el proceso e investigado todas las opciones que tiene la interfaz, podemos establecer una comparativa de la precisión y de las velocidades de distintos vídeos e imágenes. De esta manera comprobaremos la capacidad de nuestro programa.

Cabe destacar que cuanta más calidad de imagen o de vídeo, obtendremos una mayor calidad en el campo de vectores y en el escalar. De esta manera el cálculo de las velocidades será más preciso también.

Es por esto que debemos emplear, si es posible, formatos de vídeo más pesados, aunque tengan más capacidad y el software corra más lento.