

Una herramienta integral para el procesamiento de Velocimetría por Imágenes de Partículas aplicado a problemas de fluidodinámica

por

Juan Sebastian Barja y Leandro Fiaschetti

Trabajo final de la carrera de grado de
Ingeniería de Sistemas de la
Universidad del Centro de la Provincia de Buenos Aires



Director: Dr. Gustavo Boroni

Co-Director: Ing. Javier Dottori

Tandil, Argentina, Noviembre de 2015

Agradecimientos

Queremos agradecer a todas aquellas personas que han influido directamente tanto en la realización de esta tesis como en la formación como profesionales en esta rama. A Gustavo Boroni y Javier Dottori, nuestros director y codirector respectivamente, que nos han apoyado y brindado todo su tiempo y conocimiento permitiéndonos realizar este trabajo de la mejor manera posible. A todo el instituto PLADEMA, el cual nos ha brindado el lugar para realizar las pruebas necesarias contribuyendo a la finalización de esta tesis. A Juan Gomba y Ramiro Mansilla, quienes han contribuido en la realización de los experimentos para el cálculo de resultados. A toda nuestras familias incluyendo novias y amigos, quienes han sabido apoyarnos en todo momento y han transitado esta etapa final con nosotros.

Sebastián y Leandro

Índice general

Agradecimientos	2
1. Introducción	11
1.1. Antecedentes	11
1.2. Motivación	13
1.3. Objetivos	14
1.4. Organización del Trabajo	15
2. Velocimetría por Imágenes de Partículas	17
2.1. Introducción	17
2.2. Clasificación	18
2.3. Componentes	19
2.3.1. Partículas para PIV	20
2.3.2. Láser	21
2.3.3. Hardware de grabación	22
2.4. Procesamiento de imágenes	23
2.4.1. Velocimetría por seguimiento de partículas (PTV)	25
2.4.2. PIV basado en correlación	26
2.4.3. Pre-Procesamiento	34
2.4.4. Post-Procesamiento	36
2.4.5. Procesamiento en secuencias de imágenes	39
3. Relevamiento de Herramientas PIV	41
3.1. Pruebas de Herramientas PIV	41
3.2. Herramientas PIV relevadas	42

3.2.1. ImageJ + Plugin PIV	42
3.2.2. Mpiv	44
3.2.3. MatPIV	45
3.2.4. JPIV	47
3.2.5. Comparación de las herramientas PIV Analizadas	48
4. Diseño de capa de Abstracción para PIV	50
4.1. Alcance de la capa de abstracción	50
4.1.1. Interfaz	51
4.2. Arquitectura interna de la capa	56
4.2.1. Funcionamiento general basado en <i>Pipes and Filters</i>	56
4.2.2. Diseño de la arquitectura interna de la capa	62
5. Instanciación	74
5.1. Creación de un nuevo filtro	74
5.2. Instanciación de un nuevo filtro	78
5.3. Consideraciones adicionales	81
6. Análisis de resultados	83
6.1. Caso de estudio 1: Desplazamiento manual de una imagen	83
6.1.1. Procesamiento PIV	84
6.1.2. Procesamiento PIV con Pre-procesamiento	88
6.1.3. Procesamiento PIV con Pre y Post-procesamiento	89
6.1.4. Comparativa de los resultados	91
6.2. Caso de estudio 2: Repositorio de imágenes de prueba	93
6.3. Caso de estudio 3: Problema de la cavidad cúbica	94
6.4. Caso de estudio 4: Cálculo PIV para un canal poiseuille	101
6.4.1. Configuración experimental	101
6.4.2. Resultados experimentales	106
6.4.3. Inconvenientes presentados	108
Conclusiones	109
Trabajos futuros	111

Índice de figuras

1.1.	Comportamiento dinámico de partículas.	11
1.2.	Partículas iluminadas por un plano láser.	13
2.1.	Procedimiento PIV.	18
2.2.	Configuración de componentes PIV.	19
2.3.	Plano de luz formado por un láser a través de los lentes.	22
2.4.	(a) Imagen con baja densidad de partículas. (b) Imagen con densidad media de partículas. (c) Imagen con alta densidad de partículas. . . .	24
2.5.	Dificultades presentes ante la determinación del desplazamiento de las partículas.	25
2.6.	Cálculo del desplazamiento a partir de la correlación de una ventana de interrogación	27
2.7.	Ventana de interrogación de una imagen a partir de la técnica simple-fotograma/doble-pulso y la función de autocorrelación para una ventana de interrogación particular.	29
2.8.	Imágenes basada en la técnica simple-fotograma/simple-pulso y función de correlación cruzada.	29
2.9.	Ventanas de interrogación.	30
2.10.	Plano de correlación cruzada para un tamaño de ventana de 4x4. . . .	31
2.11.	Mapa de vectores de desplazamiento a partir del método de correlación cruzada.	33
2.12.	Mapa de vectores con baja resolución contra otro de mayor resolución. . . .	33
2.13.	Filtros de pre-procesamiento.	35

2.14. (a) Gráfico de Vectores. (b) Gráfico de Magnitud. (c) Gráfico de Vorticidad. (d) Gráfico de Streamlines	37
2.15. Vecindad de 8 vectores.	38
2.16. (a) Mapa de vectores con vectores inválidos. (b) Mapa de vectores tras la eliminación de vectores inválidos. (c) Mapa de vectores tras el reemplazo de vectores inválidos.	39
3.1. Pantalla de configuración del Plugin PIV y resultados obtenidos.	43
3.2. Pantalla de configuración de Mpiv y resultados obtenidos.	45
3.3. Diferentes visualizaciones de la herramienta MatPIV.	46
3.4. Interfaz gráfica de JPIV.	48
4.1. Diagrama de C&C de una aplicación comunicada con herramientas mediante la capa de abstracción.	51
4.2. Esquema general de división del problema en Filtros.	52
4.3. Diagrama de clases UML de Filtros y Elementos Procesables.	54
4.4. Arquitectura basada en Pipes and Filters.	57
4.5. Estado inicial para un ejemplo de procesamiento con Pipes and Filters.	57
4.6. Primer ciclo del escenario de ejemplo.	58
4.7. Segundo ciclo del escenario de ejemplo.	58
4.8. Tercer ciclo del escenario de ejemplo.	59
4.9. últimos ciclos del escenario de ejemplo.	59
4.10. Estado inicial para un ejemplo con múltiples filtros concurrentes.	61
4.11. Primer ciclo del escenario de ejemplo con múltiples filtros concurrentes.	61
4.12. Segundo ciclo del escenario de ejemplo con múltiples filtros concurrentes.	62
4.13. Ciclo final del escenario de ejemplo con múltiples filtros concurrentes.	62
4.14. Diagrama de secuencia de interacción entre procesos.	64
4.15. Diagrama de clases UML de Procesos, Seleccionadores y Buffers.	66
4.16. Diagrama de actividades de procesamiento.	68
4.17. Diagrama de clases de procesadores.	69
4.18. Estructura de caché implementada.	71
4.19. Diagrama de clases de Caché.	73
5.1. Diagrama de clases de patrón Adapter.	75

5.2. Resultado de aplicar el filtro Find Maxima sobre una imagen.	76
5.3. Diagrama de Clase PluginFilterManager.	79
5.4. Diagrama de actividades de la carga de filtros.	80
5.5. Ejemplo de estructura de paquetes.	81
6.1. Imágenes utilizadas para el Caso de estudio 1.	84
6.2. Selección de imágenes utilizando la herramienta.	85
6.3. Selección y configuración de un filtro de procesamiento PIV.	85
6.4. Selección y configuración de un filtro de Visualización.	86
6.5. Ejecutar PIV utilizando la herramienta.	87
6.6. Resultados de los filtro de visualización tras la aplicación del procesamiento PIV.	87
6.7. Selección y configuración de un filtro de Pre-procesamiento.	88
6.8. Previsualización de la etapa de pre-procesamiento.	89
6.9. Resultado de la aplicación del filtro HighPass sobre una de las imágenes.	89
6.10. Selección y configuración de un filtro de Post-procesamiento.	90
6.11. Gráfico comparativo de los resultados para el Caso de estudio 1. . . .	91
6.12. Comparativa de performance entre las funcionalidades ejecutadas sobre la capa de abstracción y sobre la herramienta original.	94
6.13. Configuración de la cavidad cúbica.	95
6.14. Imágenes experimentales de la cavidad cúbica.	96
6.15. Comparación entre el perfil de velocidad horizontal obtenido numéricamente y el resultado obtenido con la herramienta propuesta a partir de un filtro el procesamiento PIV.	97
6.16. Comparación entre el perfil de velocidad horizontal obtenido numéricamente y el resultado obtenido con la herramienta propuesta a partir de un filtro de pre-procesamiento CLAHE y un filtro de procesamiento PIV.	98
6.17. Comparación entre el perfil de velocidad horizontal obtenido numéricamente y el resultado obtenido con la herramienta propuesta a partir de un filtro de pre-procesamiento CLAHE y un filtro de procesamiento PIV.	99
6.18. Streamlines correspondientes al caso de estudio 3.	100

6.19. Perfil de velocidad parabólico de un flujo poiseuille.	101
6.20. Motor de bomba de desagote utilizado para el experimento.	102
6.21. Mangueras de entrada y salida del fluido conectadas al canal.	102
6.22. Filtro utilizado para laminarizar el flujo.	103
6.23. Configuración final del canal.	103
6.24. Cámara utilizada para la obtención de las imágenes.	104
6.25. Láser generando el plano de luz sobre el canal.	104
6.26. Microesferas de vidrio utilizadas como partículas trazadoras.	105
6.27. Imagen del plano de luz obtenida por la cámara de enfoque perpendicular al plano.	105
6.28. Perfiles de velocidad correspondientes a los diferentes instantes de tiempo.	107
6.29. Comparación de los perfiles de velocidad.	107

Índice de cuadros

3.1. Comparación de las herramientas relevadas en cuanto a sus funcionalidades.	49
4.1. Representación del Mapa de Vectores.	56
5.1. Estuctura del archivo de configuración.	79
5.2. Ejemplo de archivo de configuración.	81
6.1. Tabla comparativa de los resultados obtenidos en cada prueba.	92
6.2. Cuadro comparativo de los resultados para el Caso de estudio 3.	99

Capítulo 1

Introducción

En este capítulo se presentan los antecedentes de la técnica PIV (Particle Image Velocimetry) junto con los aspectos que impulsan el desarrollo de este trabajo final. Además, se plantean los objetivos a cumplir y se describe un esquema de la organización del trabajo realizado.

1.1. Antecedentes

El procesamiento digital de imágenes ha mejorado la manera en que los fluidos y sus características pueden ser estudiados. A través de este procesamiento se pueden obtener de forma automática, rápida y precisa, características importantes del fluido en cuestión, como por ejemplo densidad, temperatura, velocidad y estructuras que se forman cuando el mismo se encuentra en movimiento.

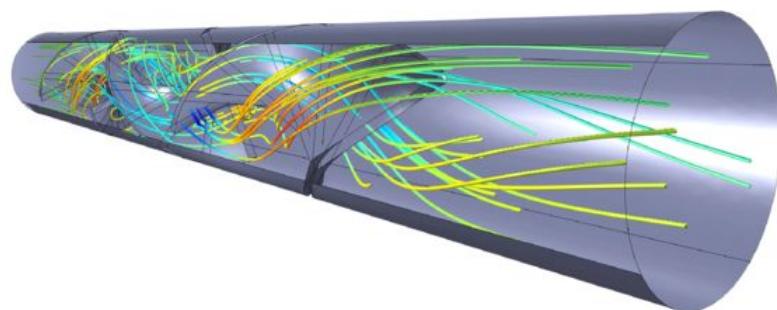


Figura 1.1: Comportamiento dinámico de partículas.

La posibilidad de determinar la velocidad de un flujo de fluidos es un problema importante en diversas áreas de la ingeniería, la industria y la ciencia. Existen diferentes técnicas para medir velocidades de fluidos [26], donde se emplean diversos componentes o dispositivos que se introducen en el fluido con el objetivo de calcular la velocidad del flujo en un punto para un intervalo de tiempo. Sin embargo, uno de los métodos más utilizados para este fin es el conocido como velocimetría de partículas (PIV), el cual es considerado como un método efectivo, instantáneo y no intrusivo.

Algunas de las primeras medidas de velocidad cuantitativas en los campos de fluidos se obtuvieron utilizando tubos de Pitot [10]. Posteriormente, la introducción de los anemómetros de filamento por hilo caliente en la década de 1920 fue un avance significativo, sobre todo en términos de la miniaturización de la sonda, respuesta de frecuencia, y la capacidad de medir múltiples componentes de la velocidad simultáneamente. Sin embargo, estas dos técnicas requieren la inserción de una sonda física que puede alterar el propio flujo. La invención del láser en la década de 1960 condujo al desarrollo del anemómetro láser Doppler que utiliza una sonda de láser para permitir mediciones de la velocidad no intrusivas. Si bien permitió medir velocidades con alta precisión y despreciable perturbación del medio, las mismas no dejaban de ser puntuales. La posibilidad de realizar mediciones globales de velocidad condujo rápidamente a que las técnicas basadas en imágenes de partículas sean muy ponderadas en el ámbito de la mecánica de fluidos [24].

En 1984, R. J. Adrián propone una técnica en la cual se ilumina un conjunto de partículas sembradas en un fluido utilizando un abanico de luz [1]. El plano formado por la intersección del abanico de luz y las partículas individuales situadas dentro del fluido, al ser fotografiado dentro de intervalos de tiempo pequeños, podía ser utilizado para encontrar la velocidad y dirección de estas partículas. Este método tomó el nombre de velocimetría de imagen de partículas (PIV) para distinguirlo de la velocimetría de punto láser (LSV) [21].

La técnica de PIV propone que el flujo del fluido puede ser observado mediante el agregado de partículas trazadoras. Si se obtienen dos imágenes del fluido con partículas en un corto lapso de tiempo de separación entre ellas, es posible deducir su velocidad en el campo de flujo. La base teórica de PIV fue presentada por Adrián

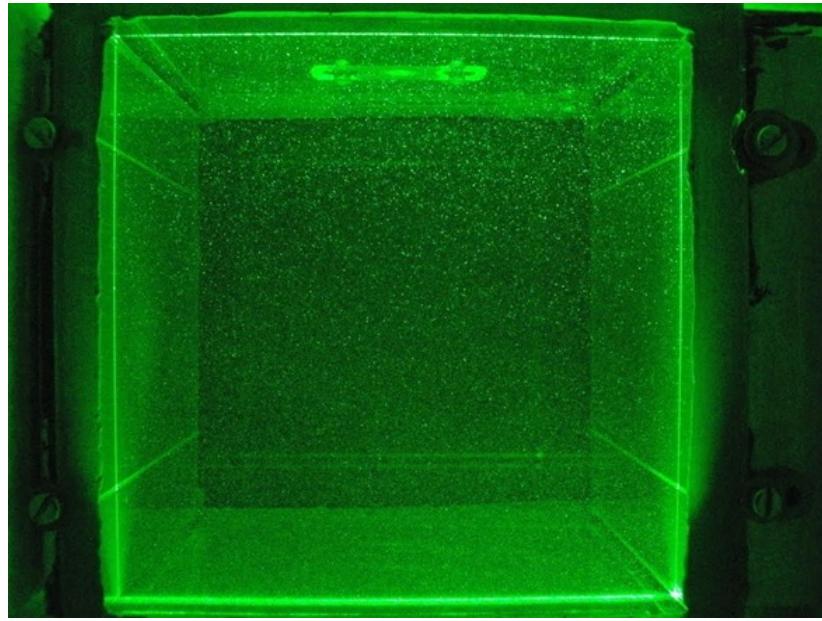


Figura 1.2: Partículas iluminadas por un plano láser.

en 1988, en donde describe el valor esperado de la función de auto-correlación para una imagen PIV continua de doble exposición [3].

Más adelante, la teoría fue generalizada para incluir múltiples exposiciones [2] y análisis de la correlación cruzada [17]. Posteriormente, la misma fue ampliada por Westerweel para incluir imágenes digitales a PIV [30]. Hoy en día PIV es una técnica ampliamente reconocida y utilizada, sobre todo con el advenimiento de las cámaras digitales [32].

1.2. Motivación

En la actualidad, existe una amplia gama de herramientas de software basadas en esta técnica. Dichas herramientas forman una potente alternativa de medición en dinámica de fluidos, capaz de determinar sin grandes errores un campo de velocidad distribuido en líquidos, gases y flujos.

La aplicación de PIV para el estudio del campo de velocidades involucra tres etapas bien identificadas. En primer lugar, un pre-procesamiento de imágenes, cuyo objetivo es mejorar la calidad de las imágenes previo a la etapa de procesamiento. En

esta etapa se pueden incluir diversos filtros, como por ejemplo filtros de eliminación de ruido y aplicación de máscaras. En segundo lugar, el procesamiento PIV, a través del cual se obtiene la información del campo de velocidades de las partículas a partir de las imágenes. Este procesamiento puede ser realizado mediante diferentes algoritmos (correlación cruzada y autocorrelación), y múltiples modos de correlación como por ejemplo convolución, transformada rápida de Fourier (FFT) y diferencia de mínimos cuadrados. Finalmente, la etapa de post-procesamiento, mediante la cual se eliminan y corrigen los vectores inválidos a partir de la aplicación de filtros, como por ejemplo, Test de mediana normalizada, Test de media dinámica, entre otros [26].

A partir de un análisis general de las herramientas de PIV más utilizadas [16] [19] [20] [23] [22], se pudo determinar que las mismas presentan diferencias significativas. Dichas diferencias fueron tanto en la utilización de funcionalidades específicas como en la instancia de configuraciones para las distintas etapas de procesamiento.

Por otra parte cada una de estas herramientas posee pros y contras dependiendo de la naturaleza del caso de estudio. Esto obliga a los usuarios a determinar la herramienta utilizar en cada caso. Entonces, se desprende naturalmente que la integración de todas las funcionalidades y etapas de procesamiento en un único sistema facilitaría a los usuarios el análisis de comportamiento del fluido. Sumado a esto, en la mayoría de los softwares analizados también se pudo observar la dificultad de incorporar nuevas funcionalidades o etapas de procesamiento, por ejemplo nuevos filtros de eliminación ruidos.

De todo lo anterior surge la idea de diseñar y construir una herramienta robusta, escalable, extensible y usable, que permita integrar fácilmente diferentes arquitecturas de software sobre la base de funcionalidades preexistentes. Esto facilitará al usuario la utilización de una configuración de procesamiento (etapas y métodos) adaptable a sus necesidades.

1.3. Objetivos

El objetivo general es desarrollar una herramienta de software que incluya una capa abstracción para integrar distintas herramientas de PIV basadas en software libre, y una arquitectura de clases que permita incorporar nuevos métodos en las

diferentes etapas de procesamiento.

En primer lugar, se propone analizar las utilidades y restricciones que ofrecen las diferentes herramientas PIV de software libre existentes, con el objetivo de seleccionar un subconjunto de ellas que permitan, en cooperación, representar la mayoría de las funcionalidades presentes en el conjunto total de herramientas analizadas. Una vez seleccionadas dichas herramientas en base a diferentes criterios (extensibilidad, funcionalidad provista, capacidad de configuración, entre otros), se realizará una capa de abstracción de software que posibilite utilizar todas las características que las mismas ofrecen, buscando la mayor transparencia para el usuario. Estas características debieran poder incorporarse sin necesidad de recompilar la herramienta, para aumentar la extensibilidad de la misma.

Posteriormente, y teniendo en cuenta que la mayoría de las herramientas hasta ahora analizadas no incluyen la etapa de pre-procesamiento, se brindarán facilidades para incorporar algoritmos que permitan mejorar la calidad de las imágenes adquiridas y así obtener mejores resultados en la etapa de procesamiento.

Por otro lado, muchas de las herramientas analizadas no aprovechan las capacidades de computo en paralelo presentes en hardware actuales. Por esto también se desea que la herramienta provea paralelismo entre los diferentes procesos que se utilicen.

Por último, se realizarán validaciones sobre problemas clásicos dentro de la simulación de fluidos, priorizando aquellos que tienen soluciones analíticas o numéricas conocidas.

Para finalizar se propone la construcción y configuración de un sistema de PIV completo (canal, láser, cámara, entrada, salida), sobre el cual se realizará un experimento aplicando las diferentes etapas de procesamiento mencionadas a partir de la herramienta propuesta.

1.4. Organización del Trabajo

El trabajo se encuentra organizado en capítulos. En el Capítulo 2 se hace una reseña de la técnica de velocimetría por imágenes de partículas (PIV), donde se presentan los principios básicos y detalles de implementación. El Capítulo 3 presenta

análisis comparativo de las principales herramientas de PIV de software libre en el mercado, haciendo énfasis en las diferentes funcionalidades que cada una de ellas ofrece, su extensibilidad y capacidad de configuración. En el Capítulo 4 se propone una arquitectura que permite fácilmente extender e incorporar nuevas funcionalidades, posibilitando la combinación de diversos filtros provistos por las distintas herramientas de manera transparente. El Capítulo 5 describe el proceso mediante el cual un nuevo filtro puede ser implementado e incorporado a la herramienta. Finalmente, en los Capítulos 6 y 7 se presentan cuatro casos de estudio y las conclusiones del trabajo.

Capítulo 2

Velocimetría por Imágenes de Partículas

En este capítulo se presenta una introducción a la técnica de velocimetría por imágenes de partículas (PIV) basadas en plano láser, con el objetivo de comprender sus principios básicos e implementación. Para ello, se describe los componentes necesarios para obtener las imágenes a analizar, el pre-procesamiento de las mismas, los algoritmos PIV para el cálculo de los campos de velocidad, y la etapa de post-procesamiento que permite mejorar los resultados obtenidos.

2.1. Introducción

La velocimetría por imágenes de partículas (PIV) es una técnica no invasiva que permite obtener, mediante el procesamiento de imágenes digitales, el campo de velocidades de un fluido en movimiento. Las herramientas basadas en esta técnica son una potente alternativa de medición de la dinámica de fluidos, capaz de medir sin grandes errores un campo de velocidad distribuido en líquidos, gases y flujo multifase[26].

Las técnicas de PIV basadas en plano láser consisten en agregar pequeñas partículas trazadoras al fluido en estudio, las cuales se asumen que se mueven con la velocidad y dirección del mismo. Si bien requiere de dominios transparentes o semi-transparentes para permitir la visibilidad de las partículas al momento de capturar

las imágenes, su aplicación ha encontrado mucha aceptación en el estudio de fluidos, permitiendo por ejemplo analizar turbulencias de manera apropiada.

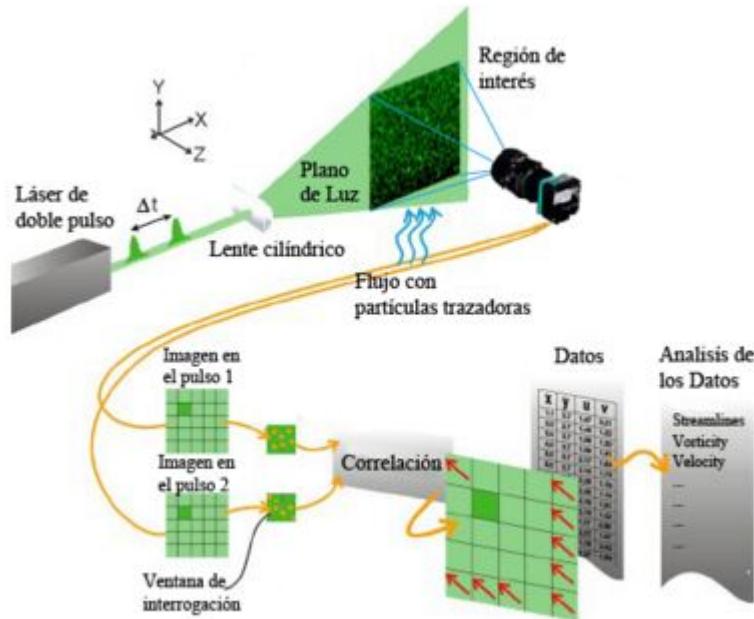


Figura 2.1: Procedimiento PIV.

2.2. Clasificación

De acuerdo a una clasificación propuesta por [14], un sistema de medición puede clasificarse según la terna (k, l, m) , donde $k = 1, 2, 3$ indica el número de componentes de velocidad que se mide, $l = 0, 1, 2, 3$ señala la cantidad de dimensiones espaciales del dominio de medición, y $m = 0, 1$ indica si el registro temporal de la medición es instantáneo o continuo. Por ejemplo, un sistema de medición puntual puede llegar a la categoría $(3, 0, 1)$ en el mejor de los casos. Por otra parte, el sistema PIV más difundido alcanza la clasificación $(2, 2, 0)$, permitiendo obtener medidas de dos componentes de velocidad en un plano para tiempos discretos (también conocido como PIV 2D-2C). La técnica PIV por holografía [12] perteneciente a la categoría $(3, 3, 0)$, utiliza la interferencia de la luz coherente dispersada por una partícula y un haz de referencia, para codificar la información de la amplitud y la fase de

la luz incidente dispersa en un plano del sensor. Desafortunadamente, las películas holográficas requieren un proceso de revelado que conlleva la mayor parte del tiempo de procesamiento de los datos, haciendo de esta técnica muy costosa. Otro caso es el PIV estereoscópico [18] (3, 2, 1), el cual utiliza dos cámaras con ángulos de visión separados para extraer el desplazamiento del eje z, donde ambas cámaras se encuentran enfocadas en el mismo punto en el flujo; y el PIV traslativo [24], que permite obtener información de velocidad 3D a partir del análisis de múltiples planos paralelos (3, 2, 0).

2.3. Componentes

PIV requiere cuatro componentes básicos (Figura 2.2) (1) una sección de prueba ópticamente transparente que contiene el flujo sembrado con partículas trazadoras; (2) una fuente de luz (láser) para iluminar la región de interés (plano o volumen); (3) el hardware de grabación que comprende una cámara CCD (*charge coupled device*), película, o placas holográficas; (4) y una computadora con el software adecuado para procesar las imágenes grabadas y extraer la información de la velocidad a partir de la posición de las partículas trazadoras.

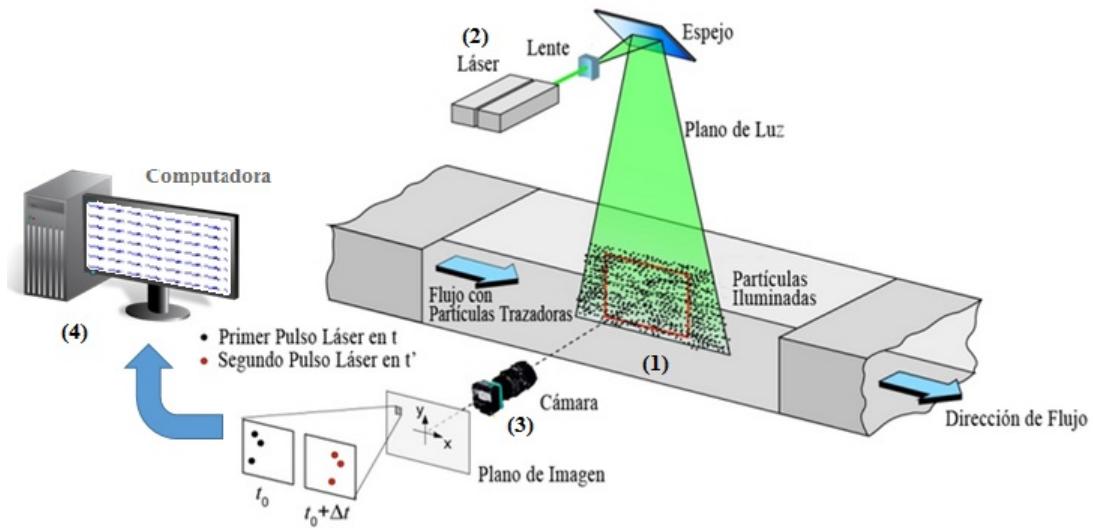


Figura 2.2: Configuración de componentes PIV.

La región del fluido donde se desea obtener el campo de velocidades es iluminada por el plano láser. La cámara digital, colocada de forma perpendicular al plano, permite obtener dos imágenes en un corto período de tiempo. Una vez que se tienen las imágenes, de ser necesario, las mismas pueden ser mejoradas para optimizar su procesamiento y eliminar defectos experimentales. Luego un algoritmo de PIV se aplica sobre las mismas para obtener el campo de velocidades correspondiente, y finalmente, se aplican técnicas de post-procesamiento para corregir errores acarreados durante el procesamiento.

2.3.1. Partículas para PIV

La técnica de PIV es utilizada para determinar la velocidad de un fluido en forma indirecta. En este sentido, la velocidad es calculada mediante el procesamiento de imágenes del fluido con partículas, que con anterioridad se insertan en el fluido. Las partículas son llamadas “partículas trazadoras”, y su movimiento deberá ser muy parecido al del fluido. De esta manera la velocidad de una partícula será similar a la velocidad del fluido en cada punto.

Las partículas trazadoras deben satisfacer dos requerimientos:

1. Deben seguir adecuadamente el flujo sin excesivo resbalamiento (slip).
2. Deben ser eficientes dispersores de la luz láser.

El primer requerimiento está relacionado con la densidad y forma de las partículas, convenientemente esféricas y de densidad similar o igual a la del fluido. El segundo impacta en la iluminación laser y el hardware de grabación.

Si bien lo ideal es encontrar partículas que al ser insertadas en el fluido sigan el movimiento del mismo sin modificar su flujo, esta característica es difícil de llevar a cabo, por lo que en la práctica se utilizan aquellas partículas que siguen el flujo sin perturbarlo demasiado. Es por este motivo, que las partículas y el fluido deben tener aproximadamente la misma densidad, evitando que las mismas floten o se hundan en el mismo.

También es necesario que las partículas dispersen bien la luz del láser, ya que de lo contrario éstas no serían captadas apropiadamente por la cámara. En general la luz dispersada por las partículas está en función de su índice de refracción y del

índice del fluido que las rodea, así como del tamaño y forma de las partículas. La luz dispersada también depende del ángulo de observación, el cual generalmente es 90° respecto de la luz incidente, por ser una de las posiciones con mayor dispersión de luz.

Por otra parte, el tamaño y cantidad de las partículas es un factor a tener en cuenta. Las más grandes son mejores reflectoras de luz; sin embargo las pequeñas se adaptan mejor al movimiento del fluido, sin provocar distorsiones en el mismo. Así, si se utilizan muchas partículas, el flujo del fluido se verá modificado, mientras que un número reducido de partículas puede provocar la carencia de información para obtener los campos de velocidad del fluido en todos los puntos.

De este modo, es importante una buena elección del tipo de partículas en cada experimento, dependiendo del fluido y tipo de flujo que se desee estudiar.

2.3.2. Láser

La técnica de PIV requiere la generación de un plano de luz para iluminar las partículas trazadoras. Debido a la capacidad que tienen los rayos láser para generar un plano de luz por medio de lentes, estos son los más utilizados en la técnica de PIV.

Los tipos de láseres aplicados a la técnica de PIV pueden ser continuos o pulsados, siendo estos últimos los más utilizados ya que por su corta duración de luz en cada pulso, permiten que una partícula (incluso a muy altas velocidades) aparezca “congelada” en la imagen, con niveles altos de intensidad. De todas formas, a partir de los láseres continuos se pueden crear pulsos de luz haciendo cortes al haz, o empleando un arreglo de espejos que deriven la luz al rotar (produciendo a la vez un plano láser por barrido).

Entre los láseres continuos es común encontrar los de Helio-Neón y en mayor medida los de iones de argón (por ser más potentes principalmente) con potencias de algunos vatios. Entre los láseres pulsados son muy utilizados los de Nd+: YAG, que pueden producir pulsos de unos 100 mJ (con potencias de megavatios en cada pulso) con repeticiones de decenas de Hz.

La elección del láser depende de la velocidad del fluido que se quiere estudiar y el tamaño de la región a iluminar. Esta última característica es importante, ya

que no todos los dispositivos tienen la misma potencia para garantizar que el plano generado sea del tamaño del área a estudiar, y al mismo tiempo que la iluminación del área sea uniforme. Además, la fuente de luz debe de ser suficientemente potente para que, sin importar su tamaño, las partículas sean iluminadas de modo tal que reflejen la cantidad de luz necesaria para que sean visibles y puedan ser capturadas por el sensor de la cámara.

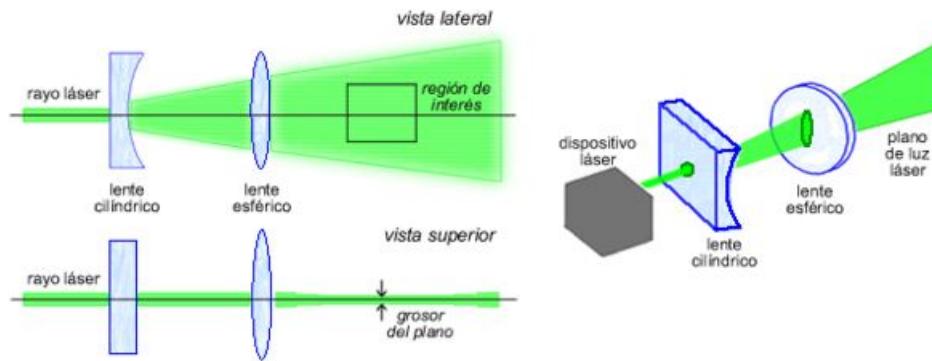


Figura 2.3: Plano de luz formado por un láser a través de los lentes.

2.3.3. Hardware de grabación

Para la captura de las imágenes en PIV es necesario de un hardware que realice dicha tarea. Aunque es posible capturar las imágenes mediante otros tipos de dispositivos, los sensores CCD son actualmente la elección más popular debido a que presentan grandes facilidades experimentales frente a otras elecciones posibles, donde el proceso de digitalización requiere más pasos.

En PIV la resolución de las imágenes es un factor a considerar, ya que si las partículas son muy pequeñas se requiere de cámaras de alta resolución para poder ser capturadas. Sin embargo, las imágenes de alta resolución ocupan mayor espacio de almacenamiento e incrementan el tiempo de procesamiento.

Otra característica a tener en cuenta es el intervalo de tiempo que debe existir entre cada par de imágenes tomadas, debido a que el mismo depende de la velocidad del fluido. Si la velocidad es muy alta y el intervalo entre las imágenes es muy

grande, no será posible determinar el campo de velocidad del flujo, ya que no habrá partículas en común entre las dos imágenes tomadas consecutivamente. Sin embargo, un intervalo muy pequeño podría dar lugar a que las partículas no presenten un desplazamiento significativo.

2.4. Procesamiento de imágenes

Las imágenes obtenidas a través de la cámara digital son la representación bidimensional del fluido cuyas partículas fueron iluminadas por el plano láser (Figura 2.2). Estas imágenes están formadas por un conjunto de valores digitales llamados pixeles, los cuales son almacenados en una matriz $f(x, y)$ de tamaño $M \times N$ cuyos valores representan la intensidad o nivel de grises en ese punto de la imagen. El intervalo de valores que puede tener $f(x, y)$ es $[0, L]$. Donde $f(x, y) = 0$ se considera negro y $f(x, y) = L$ es el blanco en la escala y generalmente es 255. Todos los valores intermedios son tonos de gris entre el blanco y el negro.

Si bien la técnica básica de PIV consiste en el procesamiento de las imágenes con el objetivo de obtener el campo de velocidades de un fluido, es importante incluir dos etapas adicionales: el pre-procesamiento basado en la aplicación de filtros para mejorar las imágenes de entrada, y el post-procesamiento que hace uso de un algoritmo de validación y sustitución para corregir los vectores resultantes.

El cálculo del campo de velocidad está basado en la relación

$$\text{velocidad} = K \frac{\text{desplazamiento}}{\text{tiempo}}, \quad (2.1)$$

siendo K la constante de conversión que da la relación entre píxeles en la imagen y unidades de longitud en el experimento. Dado que K , *tiempo* y el tiempo son valores conocidos, fijados en la configuración inicial del experimento, sólo es necesario calcular el *desplazamiento* a partir del movimiento de las partículas en el fluido en la dirección x e y para determinar las velocidades del campo.

Un factor a tener en cuenta es la densidad de partículas en la imagen, ya que de ello dependerá el método de procesamiento a utilizar. En la Figura 2.4 se observan los tres tipos de densidades de partículas en imágenes.

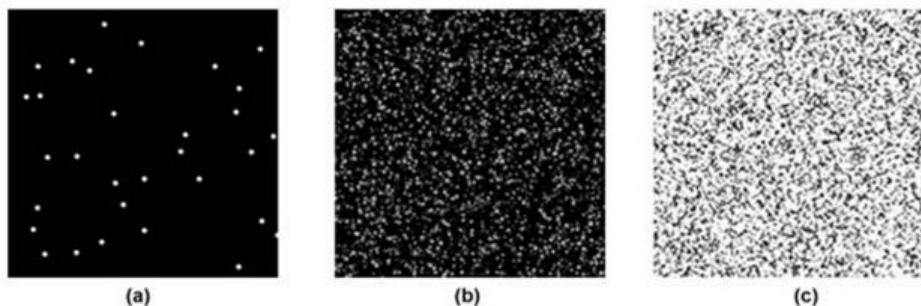


Figura 2.4: (a) Imagen con baja densidad de partículas. (b) Imagen con densidad media de partículas. (c) Imagen con alta densidad de partículas.

Si la separación de partículas es considerablemente mayor a su desplazamiento (Figura 2.4a), se considera que la densidad de partículas es pequeña. Caso contrario, si el promedio de separación entre las partículas es menor a su desplazamiento (Figura 2.4c) , se considera que la densidad de partículas es alta.

Para el caso de imágenes con baja densidad de partículas es conveniente utilizar el método “Particle Tracking Velocimetry” o “PTV” (Velocimetría por seguimiento de partículas), donde cada partícula es detectada de forma individual, permitiendo determinar su desplazamiento y velocidad local. Sin embargo, la baja densidad puede producir la falta información para determinar la velocidad en sectores donde no haya partículas presentes. PTV tiene un porcentaje de error alto en imágenes con ruido, y se debe a que el ruido es confundido con partículas durante el procesamiento de la imagen.

Las imágenes con densidad media son procesadas generalmente con la técnica convencional de PIV, ya que a simple vista no es posible distinguir una misma partícula en dos imágenes. Este método determina el desplazamiento de pequeños grupos de partículas a partir de una evaluación estadística. Al considerar un promedio del desplazamiento, la técnica de PIV es más tolerante al ruido que PTV.

Por último, las imágenes con alta densidad deben utilizar el método “Láser Speckle Velocimetry” o “LSV” (Velocimetría de Manchas por Láser), debido principalmente a la imposibilidad de identificar partículas en forma individual. Este método realiza una aproximación estadística relevando el desplazamiento de las manchas formadas por partículas. La principal desventaja es que la densidad de partículas tiende a afectar la dinámica original del fluido.

2.4.1. Velocimetría por seguimiento de partículas (PTV)

Este tipo de procesamiento consiste en iluminar las partículas dos veces en un corto lapso de tiempo, para capturar en un mismo fotograma las partículas en ambos instantes (t_0 y t_1) a través un modo de grabación llamado *simple-fotograma/doble-pulso*. La distancia entre las partículas se utiliza para estimar la velocidad local de las mismas en el plano. Cabe aclarar que para realizar este procesamiento se requiere de un trabajo sofisticado que consiste en asociar coordenadas a la imagen de cada partícula, diferenciarlas del ruido, y asociar de a pares los puntos que corresponden a la misma partícula [5].

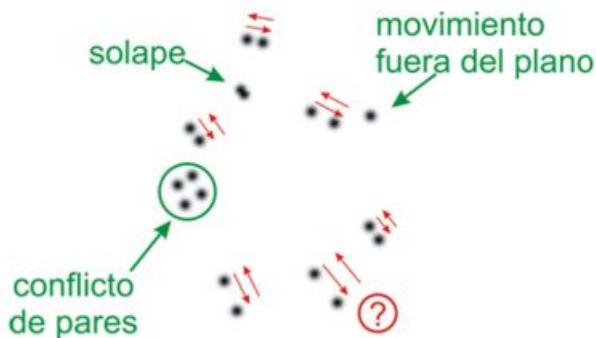


Figura 2.5: Dificultades presentes ante la determinación del desplazamiento de las partículas.

Para reducir el ruido generalmente se eliminan valores por debajo de un umbral, y se buscan los centros de los puntos brillantes. Sin embargo, existen otras dificultades que surgen durante este tipo de procesamiento, entre las que podemos nombrar el solapamiento, el movimiento fuera del plano y los conflictos de pares.

El *solapamiento* sucede cuando las partículas no se encuentran lo suficientemente separadas en la imagen, de modo que imposibilite determinar si se trata de una partícula o varias. Esto puede deberse a dos factores; la densidad de partículas no es suficientemente baja, con lo cual dos partículas se superponen, o el intervalo de tiempo con el que son capturadas las imágenes es muy pequeño, ocasionando que el desplazamiento de una partícula no sea notorio en la imagen.

El *movimiento fuera del plano* ocurre cuando la partícula queda fuera del plano láser en una de las dos capturas, imposibilitando encontrar su par en la imagen.

Otra dificultad que se puede presentar es el *conflicto de pares*, donde un grupo de partículas se encuentran distribuidas de forma tal que es imposible determinar pares de partículas. Aunque eventualmente esta situación podría darse en condiciones normales, al igual que en el primer caso, suele deberse a que la densidad de partículas no es lo suficientemente baja.

Por último, el hecho que las partículas en ambos instantes de tiempo se encuentran en un mismo fotograma dificulta la determinación del sentido de su movimiento, aunque este problema puede ser resuelto si se conoce con anterioridad la dirección del flujo.

Como se mencionó anteriormente, en general PTV funciona mejor cuando las concentraciones de partículas son bajas. Sin embargo, dado que los resultados siempre se presentan en lugares donde se encontraron pares de partículas, puede generarse un plano de medición no-uniforme.

Pese a que en la actualidad no es una técnica muy utilizada, PTV fue muy útil en los comienzos de PIV. En 1985, como bien describe Adrians a su computadora DEC PDP 11/23 en [4], el poder de almacenamiento y cómputo era muy bajo, y PTV era la solución más práctica en ese entonces.

2.4.2. PIV basado en correlación

La correlación es el método de análisis de imágenes de PIV más extendido, ya sea sobre una sola imagen con doble pulso (autocorrelación), o sobre dos imágenes aplicando un solo pulso por cada una (correlación cruzada).

A diferencia de PTV, las técnicas basadas en correlación no requieren buscar la pareja de cada partícula individual. En lugar de determinar el desplazamiento de partículas individuales, este método determina el desplazamiento medio de grupos de partículas contenidas en pequeñas regiones conocidas como ventanas de interrogación. Dichas ventanas suelen ser cuadradas y se distribuyen como una malla uniforme sobre la imagen de PIV del flujo. La imagen de PIV es dividida en estas regiones, y la función de correlación se calcula secuencialmente sobre todas las ventanas (Figura 2.6).

Para cada ventana se calcula la correlación con diferentes desplazamientos. Luego, el vector de desplazamiento es calculado a partir de la distancia entre el centro de

esta función y alguno de sus máximos. Más adelante en esta sección se detallan diferentes aspectos al tener en cuenta en la elección del mismo, dependiendo de la correlación aplicada.

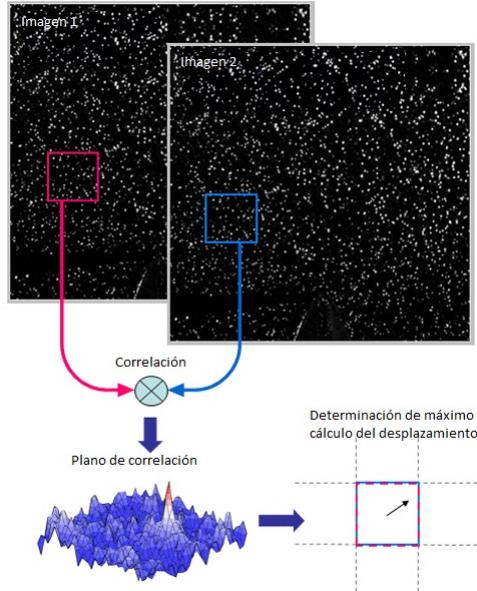


Figura 2.6: Cálculo del desplazamiento a partir de la correlación de una ventana de interrogación

2.4.2.1. Autocorrelación

La autocorrelación es un caso particular de correlación, ya que la imagen se correlaciona consigo misma, y se aplica cuando las dos imágenes consecutivas de las partículas se graban en un mismo fotograma. Esto puede generarse dejando el obturador de la cámara abierto durante dos pulsos láser consecutivos, que es un modo de grabación llamado *simple-fotograma/doble-pulso* [24].

Para un patrón de intensidades $I(i, j)$ (el fotograma) se define la función de autocorrelación $C(x, y)$ donde (i, j) son las coordenadas de cada pixel dentro de la ventana (vi) y (x, y) el desplazamiento respecto del centro de la ventana de interrogación como:

$$C(x, y) = \int \int I(i, j)I(i + x, j + y)di dj, \quad (2.2)$$

donde las integrales se realizan sobre todo el dominio dentro de vi .

La integral de área se convierte en una doble sumatoria discreta cuando la definición se traslada al dominio digital (discreto):

$$\phi_{auto}(x, y) = \sum_{i=0}^N \sum_{j=0}^M I(i, j)I(i + x, j + y), \quad (2.3)$$

siendo N y M el tamaño de la ventana en cada dimensión.

Denotando la operación de correlación mediante el símbolo \odot y la operación de convolución por \otimes . Las mismas se relacionan por:

$$C(x, y) = I(i, j) \odot I(i, j) = I(i, j) \otimes I(-i, -j), \quad (2.4)$$

donde $F[\cdot]$ es la Transformada Rápida de Fourier bidimensional (FFT-2D). Por el teorema de convolución:

$$F[C(x, y)] = F[I(i, j) \odot I(i, j)] = F[I(i, j)]F^*[I(i, j)] = |F[I(i, j)]|^2 \quad (2.5)$$

$$\Rightarrow C(x, y) = F^{-1}[|F[I(i, j)]|^2]. \quad (2.6)$$

Esta función tiene la posibilidad de calcularse para la imagen digital de forma más eficiente si posee las dimensiones apropiadas. Se requiere que sea cuadrada y sus lados una potencia de 2.

Como se observa en la Figura 2.7, la función de autocorrelación para una ventana de interrogación cuenta con tres picos destacados. El máximo en la posición (0,0) y dos picos secundarios de altura muy similar, situados simétricamente en la posición (S_x, S_y) y ($-S_x, -S_y$). El máximo central corresponde a correlacionar un desplazamiento nulo de los patrones en esa ventana, mientras que los máximos secundarios son utilizados para determinar la magnitud y dirección del movimiento de las partículas, pero no así de la dirección de las mismas. Sin embargo, esta desventaja puede ser solucionada si se conoce el sentido preferencial del flujo o estrategias adicionales [24]. Adicionalmente, este método impide determinar desplazamientos nulos o muy pequeños, ya que el pico central tiene en general el diámetro de la imagen de una partícula.

Debido a estos problemas, en general se evita hacer PIV por autocorrelación, uti-

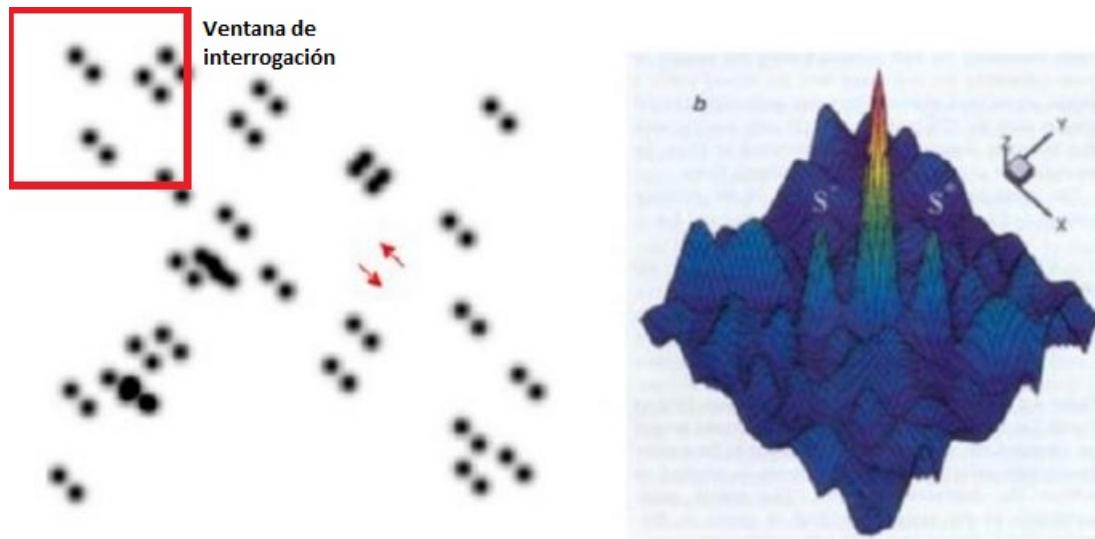


Figura 2.7: Ventana de interrogación de una imagen a partir de la técnica simple-fotograma/doble-pulso y la función de autocorrelación para una ventana de interrogación particular.

lizando preferentemente correlación cruzada el cual posee ciertas ventajas las cuales serán expuestas en la siguiente sección.

2.4.2.2. Correlación Cruzada

Este tipo de correlación se establece entre dos fotogramas, los cuales corresponden cada uno a un pulso de láser en un intervalo de tiempo corto. Así, cada fotograma se corresponde con la imagen de partículas iluminadas en cada pulso de láser.

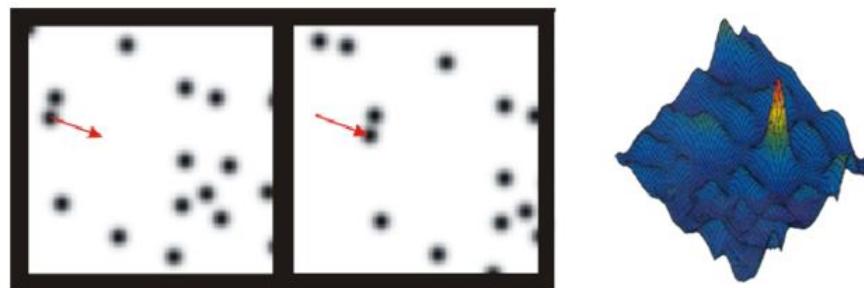


Figura 2.8: Imágenes basada en la técnica simple-fotograma/simple-pulso y función de correlación cruzada.

Contar con dos imágenes secuenciales del movimiento de las partículas permite que al correlacionar ambos fotogramas se obtenga un único pico máximo. Este determina el desplazamiento de las partículas, eliminando las ambigüedades de sentido que presentaba el algoritmo de autocorrelación.

Como ya se ha mencionado, cada imagen se divide en regiones o subimágenes llamadas “ventanas de interrogación”. Se asume que las partículas dentro de estas ventanas se mueven de manera uniforme.

El algoritmo consiste en obtener la correlación de cada una de las ventanas de interrogación de una imagen con su par en la segunda imagen para diferentes desplazamientos. De este modo, a partir del pico máximo de correlación entre los desplazamientos, es posible obtener el de las partículas en dicha ventana.

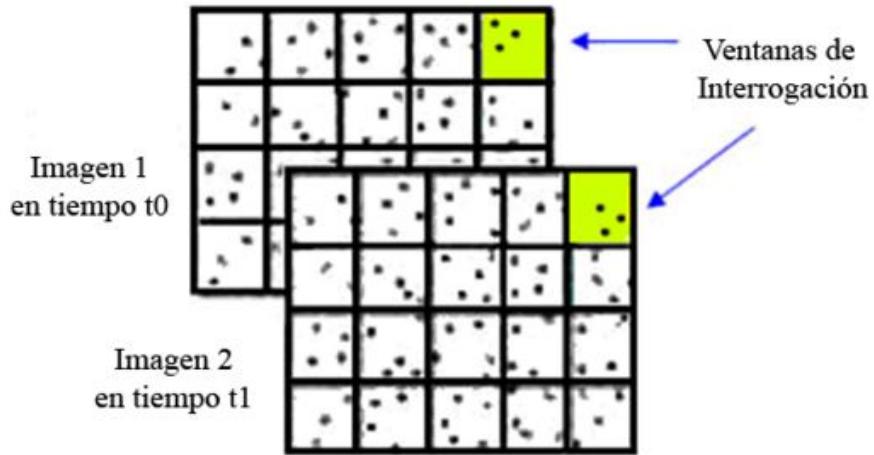


Figura 2.9: Ventanas de interrogación.

La función de correlación cruzada en un punto (x, y) entre dos ventanas de interrogación se define como:

$$C(x, y) = f(i, j) \odot g(i, j) = \sum_{i=0}^N \sum_{j=0}^M f(i, j)g(i + x, j + y), \quad (2.7)$$

con $i = 0, \pm 1, \pm 2, \dots, \pm N - 1$, $j = 0, \pm 1, \pm 2, \dots, \pm M - 1$, siendo $f(i, j)$ la ventana de interrogación correspondiente a la primera imagen y $g(i, j)$ la ventana de interrogación correspondiente a la segunda imagen, ambas de $M \times N$ pixeles. El proceso consiste

en desplazar una ventana de interrogación sobre la otra, sumando los productos de los valores en donde haya solapamiento en ambas ventanas. Cada valor en cada punto es guardado en una matriz llamada “matriz o plano de correlación cruzada”. La coordenada en donde este se almacena está dada por el desplazamiento de la segunda subimagen sobre la primera.

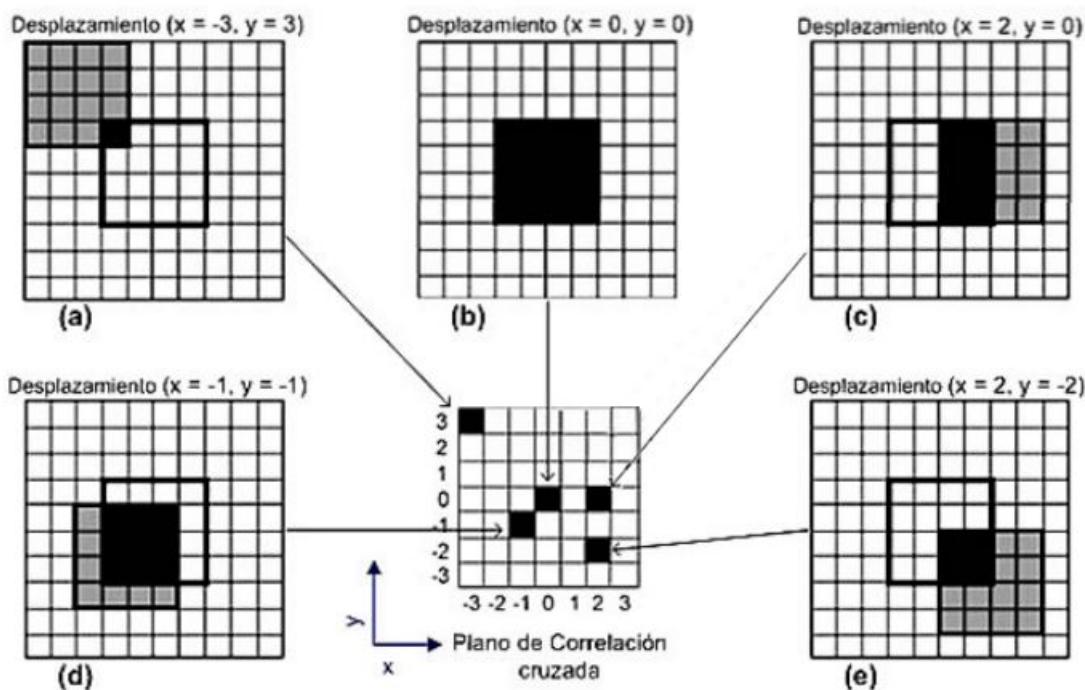


Figura 2.10: Plano de correlación cruzada para un tamaño de ventana de 4x4.

En la Figura 2.10 se observa un ejemplo del cálculo del plano de correlación cruzada para un tamaño de ventana de 4x4. En la Figura 2.10(a) se muestra un desplazamiento de -3 en x y 3 en y quedando solo un pixel de la subimagen solapado (pixel de color negro). En cambio, se puede ver en la Figura 2.10(b) que el solapamiento es total, producto de un desplazamiento nulo. Cada uno de los resultados obtenidos a partir de la suma de los productos de los pixeles solapados es almacenado en el plano de correlación cruzada en las coordenadas correspondientes al desplazamiento aplicado para el cálculo. De esta manera el tamaño de dicha matriz es de $2M - 1 \times 2N - 1$

Cada uno de los máximos obtenidos en el plano de correlación cruzada corres-

ponden a los solapamientos de mayor similitud de una imagen sobre la otra, donde el máximo absoluto representa el desplazamiento promedio de las partículas en esa ventana de interrogación. De esta forma, las coordenadas del mismo permiten determinar el desplazamiento en píxeles que tuvieron las partículas. A diferencia del método de autocorrelación, en la correlación cruzada solo se obtiene un pico máximo. Debido a que las partículas se encuentran en imágenes separadas, este pico duplica la señal respecto de los picos obtenidos a través de la autocorrelación donde las partículas de ambos instantes de tiempo se encuentran en una única imagen.

Este método de cálculo de correlación cruzada tiene un costo computacional elevado, con lo cual suele ser imposible de aplicar en imágenes de alta resolución. En esos casos, se aplica la transformada de Fourier la cual permite reducir la complejidad computacional. El teorema de correlación cruzada establece que:

$$F[C(x, y)] = F[f(i, j) \odot g(i, j)] = F[f(i, j)]F^*[g(i, j)] = A(u, v)B^*(u, v), \quad (2.8)$$

siendo A la transformada de Fourier de la subimagen f y B^* la conjugación de la transformada de Fourier de la subimagen g . De esta forma, tenemos que:

$$\Rightarrow C(x, y) = F^{-1}[A(u, v)B^*(u, v)] \quad (2.9)$$

Una vez aplicado el proceso de correlación cruzada sobre todas las ventanas de interrogación se obtiene como resultado una matriz que indica el desplazamiento medio de las partículas para cada ventana de interrogación. Conociendo además el intervalo de tiempo entre la captura de ambas imágenes y el factor de escala utilizado, es posible determinar la velocidad local del fluido en cada ventana, conformando el campo de velocidades de un fluido. Generalmente dichas velocidades son expresadas de forma vectorial. El tamaño de ventana determina el número de ventanas en que la imagen puede ser dividida, y por consiguiente, el número total de vectores de velocidad resultantes. Entre más pequeña sea la zona de interrogación, más vectores se obtendrán.

Si bien las ventanas de interrogación pequeñas aumentan la resolución y el detalle del fluido, pueden afectar negativamente la validez de los resultados obtenidos.

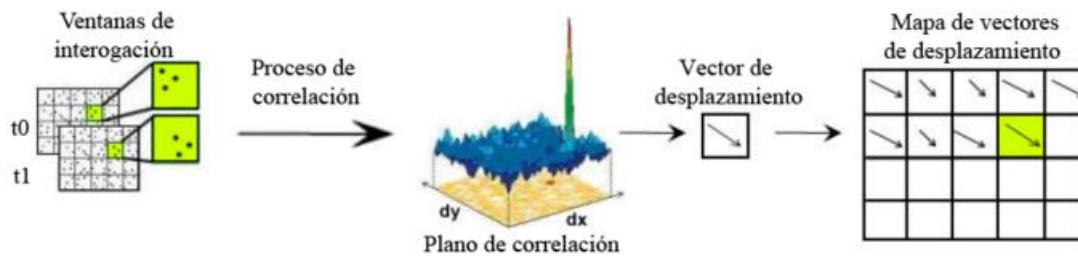


Figura 2.11: Mapa de vectores de desplazamiento a partir del método de correlación cruzada.

Teniendo ventanas muy pequeñas puede que se encuentren muy pocas partículas e incluso ninguna, lo cual haría imposible de realizar un mapeo con las partículas de la segunda imagen, generando vectores inválidos. Un número que se puede tomar como referencia es buscar que haya por lo menos entre 5 y 10 partículas por zona de interrogación [24].

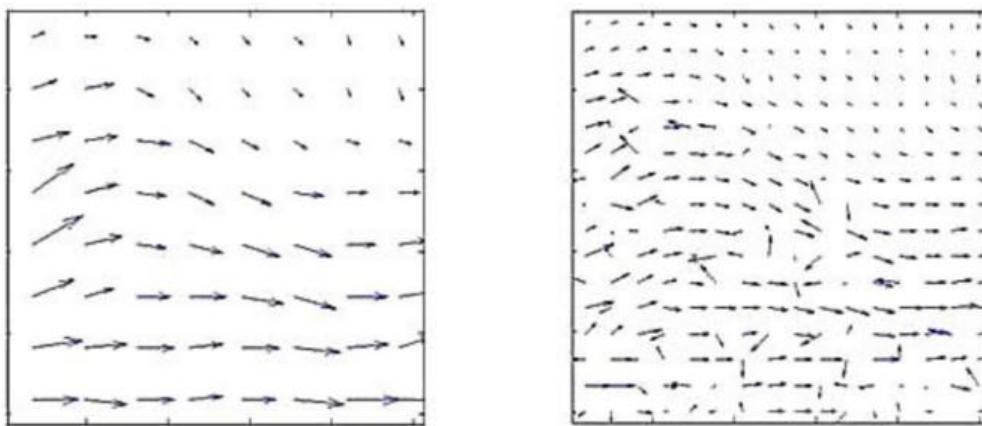


Figura 2.12: Mapa de vectores con baja resolución contra otro de mayor resolución.

La técnica de Multigrid es un método que permite obtener mapas de velocidades con diferentes resoluciones. El procesamiento con correlación cruzada se realiza de forma iterativa, y en cada iteración se reduce un determinado porcentaje el tamaño de las zonas de interrogación para obtener una mayor resolución de los resultados. Generalmente se hacen hasta tres iteraciones, y en cada una se utiliza la información de la iteración anterior [27].

En la actualidad se prefiere utilizar el análisis por correlación cruzada ya que ofrece ventajas importantes respecto al análisis de autocorrelación [24]:

- El pico central de autocorrelación desaparece.
- Se duplica la señal del pico de desplazamiento.
- Se elimina la ambigüedad direccional (un solo pico en lugar de dos simétricos).
- Mejora la relación señal-ruido.

2.4.3. Pre-Procesamiento

La intensidad de iluminación no uniforme de las partículas, el plano láser no uniforme entre los pulsos, la forma irregular de las partículas, y los movimientos fuera del plano, entre otros factores, introducen ruido al plano de correlación al momento de determinar el desplazamiento de las partículas. Por este motivo, mejorar las imágenes antes de realizar el procesamiento puede resultar más que conveniente. El objetivo del pre-procesamiento consiste en mejorar la calidad de las medidas aplicando diferentes técnicas sobre las imágenes. Los métodos de mejora de imágenes recomendados por Raffel en su libro Particle Image Velocimetry [26] centran su objetivo en obtener imágenes donde el contraste y la intensidad del brillo de las partículas tengan un nivel similar, de modo que cada una de ellas contribuya de igual manera a la función de correlación. Un ejemplo de los métodos de mejora de imagen se ve en la Figura 2.13.

- Histogram equalization:

La ecualización del histograma adaptativo de contraste limitado (CLAHE) fue desarrollada para aumentar la legibilidad de los datos de la imagen. CLAHE opera en regiones pequeñas de una imagen, donde las intensidades más frecuentes del histograma de la imagen se extienden a toda la gama de los datos (de 0 a 255 en imágenes de 8 bits). Por lo tanto, las regiones con baja exposición y regiones con alta exposición se optimizan de forma independiente. CLAHE mejora significativamente la probabilidad de detectar vectores válidos en imágenes experimentales en un rango de $4,7 \pm 3,2\%$ [29].

- High-pass:

La iluminación no apropiada puede causar la refracción de la luz sobre el fondo, generando información de baja frecuencia, que puede ser eliminada mediante la aplicación de un filtro de paso alto. Este filtro conserva la mayoría de la información de alta frecuencia, la cual corresponde a la iluminación de las partículas trazadoras. El filtro hace hincapié en la información de la partícula en la imagen, y suprime cualquier información de baja frecuencia en las mismas [29].

- Intensity capping:

El método PIV asume que todas las partículas dentro de un área de interrogación tienen el mismo movimiento. Este no será el caso en la realidad, apenas existe en un flujo uniforme. Partículas brillantes o puntos brillantes dentro del área contribuirán estadísticamente más a la señal de la correlación, lo que puede sesgar el resultado en un flujo no uniforme. El filtro de limitación de intensidad evita este problema. Para ello se selecciona un límite superior de la intensidad en escala de grises, y todos los píxeles que superan el umbral se sustituyen por este límite superior. Por lo tanto, a diferencia de CLAHE, se ajusta sólo una pequeña cantidad de la información de intensidad de los píxeles, lo que limita el potencial impacto negativo de las modificaciones de imagen. El filtrado de intensidad mejora la probabilidad de detectar vectores válidos en imágenes experimentales en un $5,2 \pm 2,5\%$ [29].

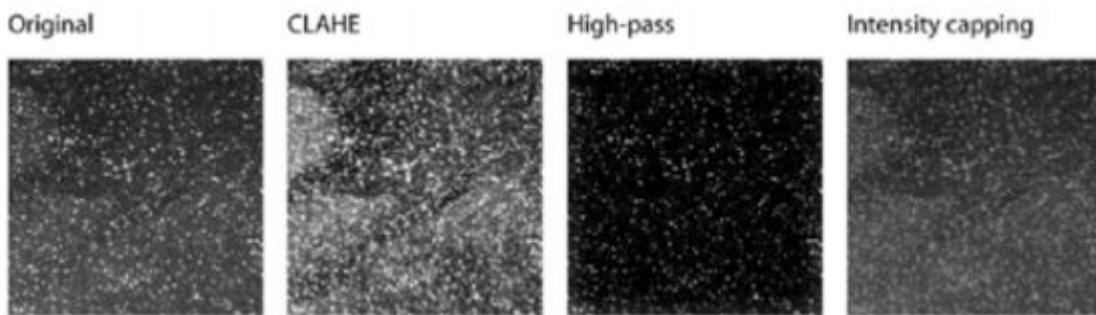


Figura 2.13: Filtros de pre-procesamiento.

2.4.4. Post-Procesamiento

Una vez obtenidos los resultados pueden ser procesados nuevamente para mejorarllos, y/o realizar un análisis de los mismos. Esta etapa es llamada post-procesamiento e incluye las siguientes etapas:

- Validación de los datos:

Aún en condiciones ideales el 5 % de los vectores obtenidos luego de la evaluación de las imágenes son incorrectos. Con algoritmos especiales se debe realizar una validación automática para detectar y eliminar estos vectores a fin de reducir el error que los mismos producen.

- Reemplazo de los datos incorrectos:

Los vectores incorrectos eliminados durante la validación deben ser reemplazados por nuevos vectores para poder hacer un correcto análisis de los resultados.

- Reducción de datos:

Esta etapa tiene como objetivo simplificar el estudio del flujo, utilizando información estadística o datos importantes que faciliten el análisis del comportamiento del fluido; tales como, velocidad media y fluctuaciones, partes periódicas y no periódicas, vorticidad, entre otros.

- Presentación y animación de la información:

Un aspecto importante es ofrecer una presentación gráfica de los resultados, que permita visualizar las características principales del flujo. Dicha presentación puede ser por medio gráficos o animaciones (estas últimas son útiles para casos de series de tiempo o datos en 3D). Entre las formas de presentación mas utilizadas se destacan los gráficos de vectores, que permiten visualizar la dirección y sentido con que se desplazan las partículas, gráficos de magnitud, que permiten apreciar rápidamente la magnitud de los vectores a partir de escalas de colores, gráficos de vorticidad, que describen la rotación del campo del flujo y gráficos de streamlines, que trazan líneas de flujo a partir del mapa de vectores.

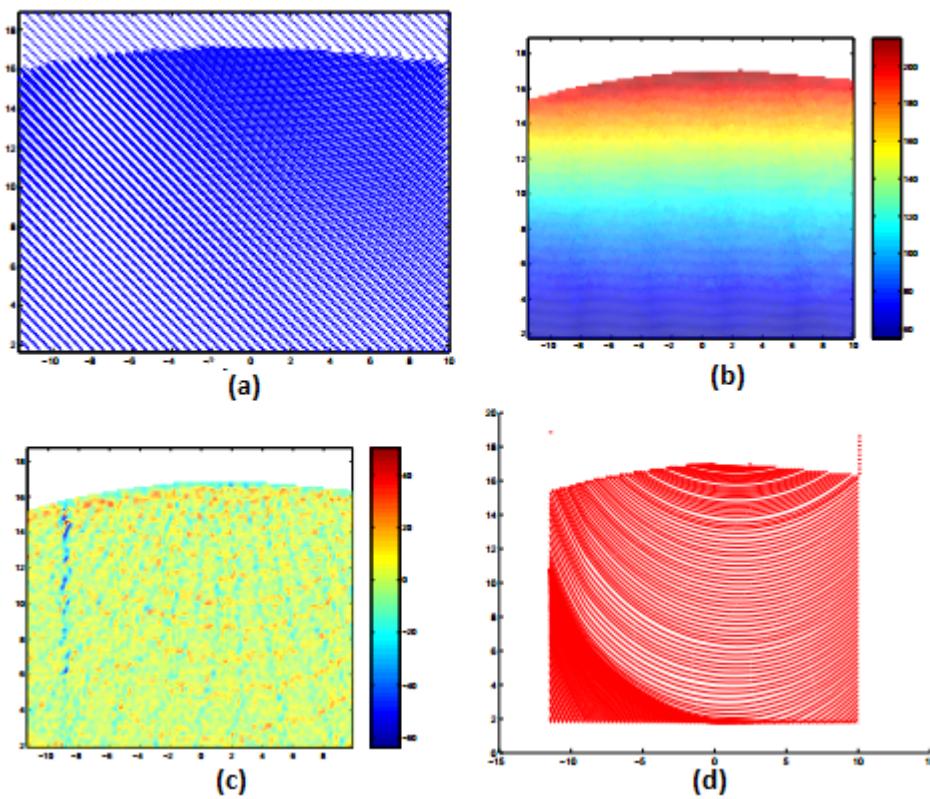


Figura 2.14: (a) Gráfico de Vectores. (b) Gráfico de Magnitud. (c) Gráfico de Vorticidad. (d) Gráfico de Streamlines

Es inevitable tener vectores incorrectos incluso en condiciones ideales. Los mismos pueden deberse a diversos motivos: que las partículas sean confundidas con reflejos de la luz de láser, que las ventanas de interrogación sean demasiado pequeñas para la densidad de partículas presentes en el fluido, o que las partículas ubicadas en los bordes no sean capturadas por la segunda imagen debido a su movimiento fuera del plano láser. Estos motivos causan que, al correlacionar las ventanas de interrogación, el máximo absoluto no represente el desplazamiento real de las partículas en esa ventana.

Para resolver estos problemas se suele utilizar un algoritmo de validación el cual consta de dos etapas: determinación de vectores incorrectos, y sustitución de los mismos. La primera etapa puede realizarse a través de la información obtenida del pico de correlación o a través de una comparación de cada uno de los vectores con

los que se encuentren en su vecindad. En el primer caso el método mas utilizado es Signal-to-Noise ratio (SNR), el cual considera que los picos de correlación obtenidos por debajo de un cierto umbral dan como resultado vectores inválidos o espurios [33]. En el segundo caso los métodos más comunes de validación son: Media global, Media local, Mediana local e Histograma global [26] y [30]. Una opción es definir un porcentaje de diferencia entre un vector y la media de sus vecinos, normalizada por la desviación estándar. Si el resultado sobrepasa un umbral previamente definido por el usuario entonces el vector es eliminado. Posteriormente los vectores eliminados serán reemplazados por unos nuevos calculados a partir de los valores de sus vecinos.

Se define la vecindad de un vector $V(i,j)$ como los n vectores más cercanos a dicho vector. En la Figura 2.15 se observa un ejemplo de una 8-vecindad para el vector central $V(i,j)$.

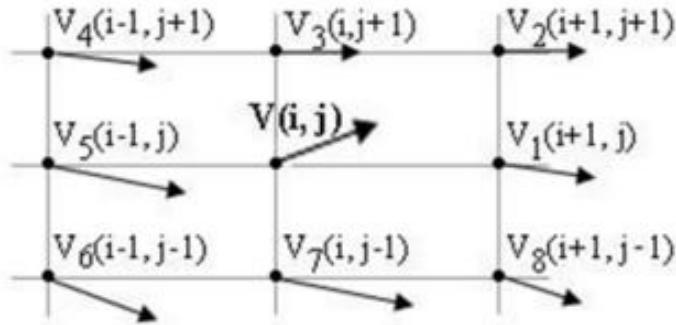


Figura 2.15: Vecindad de 8 vectores.

Una vez eliminados los vectores incorrectos estos son reemplazados por nuevos vectores resultantes de la interpolación de sus vecinos. El método más utilizado es la interpolación bicúbica de sus 4-vecinos. Sin embargo, es posible que en esta etapa no se cuente con la cantidad de vecinos necesaria, producto de las eliminaciones de la etapa anterior. En tal caso, se utiliza la información de los vectores vecinos más cercanos, hasta una distancia máxima de 3 veces $\Delta XZONA$, $\Delta YZONA$ o $\sqrt{(\Delta XZONA^2 + \Delta YZONA^2)}$. Siendo $\Delta XZONA$ y $\Delta YZONA$ el tamaño en pixeles de la ventana de interrogación en x e y respectivamente. En caso de que una primera iteración no logre reemplazar la totalidad de los vectores eliminados, es necesario repetir el proceso hasta que el mapa de vectores esté completo.

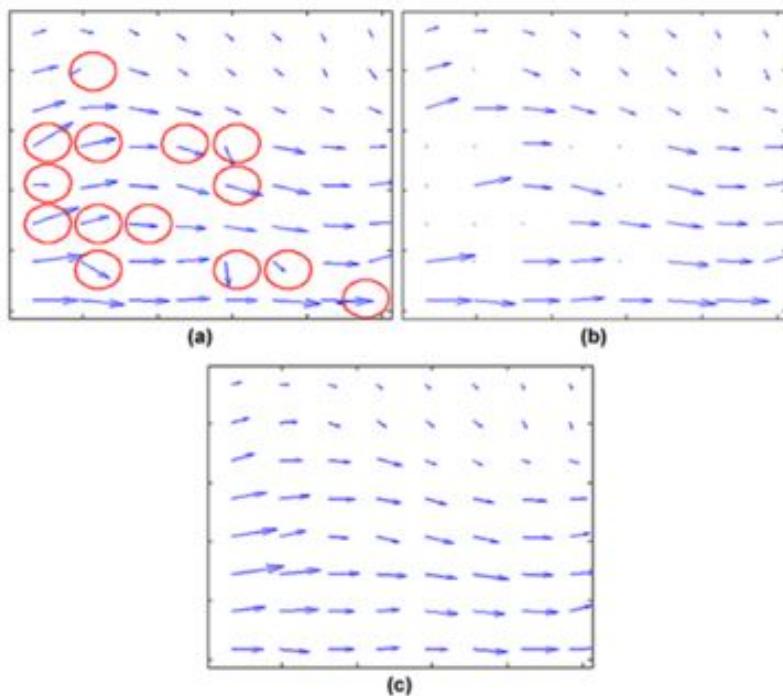


Figura 2.16: (a) Mapa de vectores con vectores inválidos. (b) Mapa de vectores tras la eliminación de vectores inválidos. (c) Mapa de vectores tras el reemplazo de vectores inválidos.

2.4.5. Procesamiento en secuencias de imágenes

Hasta el momento solo se ha mencionado la aplicación de PIV para el análisis de dos imágenes tomadas en un corto intervalo de tiempo. Sin embargo, es posible utilizar esta técnica para el análisis de secuencias de imágenes con diversos objetivos. En fluidos donde el flujo puede variar a través del tiempo, el análisis secuencial de las imágenes puede permitir observar la evolución del mismo. Asimismo, en fluidos cuyo flujo se mantiene constante, dicho análisis secuencial reduce el error producido durante el cálculo PIV mediante la evaluación de los resultados obtenidos. Para esto es posible utilizar diferentes técnicas, como por ejemplo, el cálculo de la media de cada uno de los vectores en los diferentes mapas de vectores resultantes. En conclusión, son llamadas secuencia de imágenes a un conjunto imágenes cuyo orden está dado por el tiempo en el cual fueron tomadas las mismas.

Hay dos formas comunes de considerar una secuencia de imágenes:

- Pares Consecutivos:

Como su nombre lo indica, las imágenes son seleccionadas de a pares consecutivos donde cada imagen pertenece a un único par, es decir, si contamos con cuatro imágenes numeradas del 1 al 4, se formarían dos pares, (1, 2) y (3, 4). De esta forma la cantidad de pares formados será $N/2$, siendo N la cantidad de imágenes de la secuencia. En este caso, el intervalo de tiempo entre las imágenes 1 y 2 deberá ser el mismo que entre las imágenes 3 y 4. Por otra parte, no será necesario mantener dicho intervalo entre las imágenes 2 y 3 ya que estas no serán analizadas conjuntamente.

- Cascada:

Este método, al igual que el anterior, se basa en formar pares de imágenes, pero con la diferencia de que, exceptuando la última imagen, todas las demás serán el inicio de cada par. De esta forma si contamos nuevamente con cuatro imágenes numeradas del 1 al 4, los pares formados serían, (1, 2), (2, 3), (3, 4). Bajo este método, la cantidad de pares formados será $N-1$, siendo N la cantidad de imágenes de la secuencia. Otra diferencia con el método anterior es que permite obtener mayor cantidad de resultados con la misma cantidad de imágenes.

Capítulo 3

Relevamiento de Herramientas PIV

En el siguiente capítulo se presenta un estudio de las diferentes herramientas PIV de software libre más utilizadas. Se realiza un análisis comparativo basado en las funcionalidades que las mismas presentan, y propiedades tales como modificabilidad, extensibilidad y usabilidad. Así mismo, se selecciona un subconjunto que permita abarcar el mayor espectro de características que un sistema PIV requiere.

3.1. Pruebas de Herramientas PIV

Como se mencionó en el Capítulo 2, la velocimetría por imágenes de partículas involucra tres etapas bien identificadas. En primer lugar, un pre-procesamiento de imágenes, cuyo objetivo es mejorar la calidad de las imágenes previo a la etapa de procesamiento, a fin de optimizar los resultados obtenidos. En segundo lugar, el procesamiento PIV propiamente dicho, a través del cual se obtiene la información del campo de velocidades de las partículas. Finalmente, la etapa de post-procesamiento, mediante la cual se eliminan y corrigen los vectores inválidos [26]. Estas etapas comprenden un rol fundamental en la calidad del resultado final obtenido.

En este sentido, se prioriza la necesidad de contar con una herramienta de software que provea un amplio espectro de funcionalidades aplicables en cada una de las etapas. Para ello se realizó un relevamiento de las herramientas PIV de software

libre más utilizadas, y un análisis de las distintas características que pueden ofrecer cada una de ellas.

Este relevamiento incluyó la descarga de las herramientas junto con su código fuente y documentación, a través de los cuales fue posible evaluar diferentes atributos de calidad como modificabilidad, flexibilidad y performance. Así mismo, se realizó un análisis de la capacidad de configuración de los diferentes parámetros necesarios para realizar el procesamiento. Luego, se estudió la facilidad de ejecución de la herramienta sumando la usabilidad como otro atributo de calidad a ser evaluado. En este último caso y como primer medida, se realizaron diferentes pruebas ejecutando el procesamiento PIV con diversas imágenes básicas proporcionadas por las diferentes herramientas. Dichas pruebas tuvieron como objetivo, observar comportamiento del procesamiento PIV de ellas, ya que en muchos casos se obtuvieron errores en la ejecución, solo por ejecutar el procesamiento con imágenes no provistas por la herramienta que estaba siendo evaluada.

3.2. Herramientas PIV relevadas

Durante el relevamiento de las herramientas, se realizó un análisis detallado de cuatro herramientas de software libre. Entre ellas se encuentra “ImageJ” junto con su plugin para el procesamiento PIV, otras herramientas desarrolladas en MATLAB como “matPIV” y “MPIV” y por último se estudió el comportamiento de una herramienta en código Java llamada JPIV.

3.2.1. ImageJ + Plugin PIV

ImageJ es un software Java de código abierto, dedicado ampliamente al tratamiento de imágenes, sobre el cual es posible añadir plugins fácilmente que extienden su funcionalidad básica. En este caso, Plugin PIV [23] permite el cálculo de velocimetría de partículas por imágenes (PIV) de dos modos, el clásico cálculo mediante correlación de imágenes, utilizando igual tamaño de ventana de interrogación para ambas imágenes; o el método de coincidencia de plantillas con el algoritmo coeficiente de correlación normalizado, donde la ventana de interrogación se compara contra una ventana de búsqueda más grande.

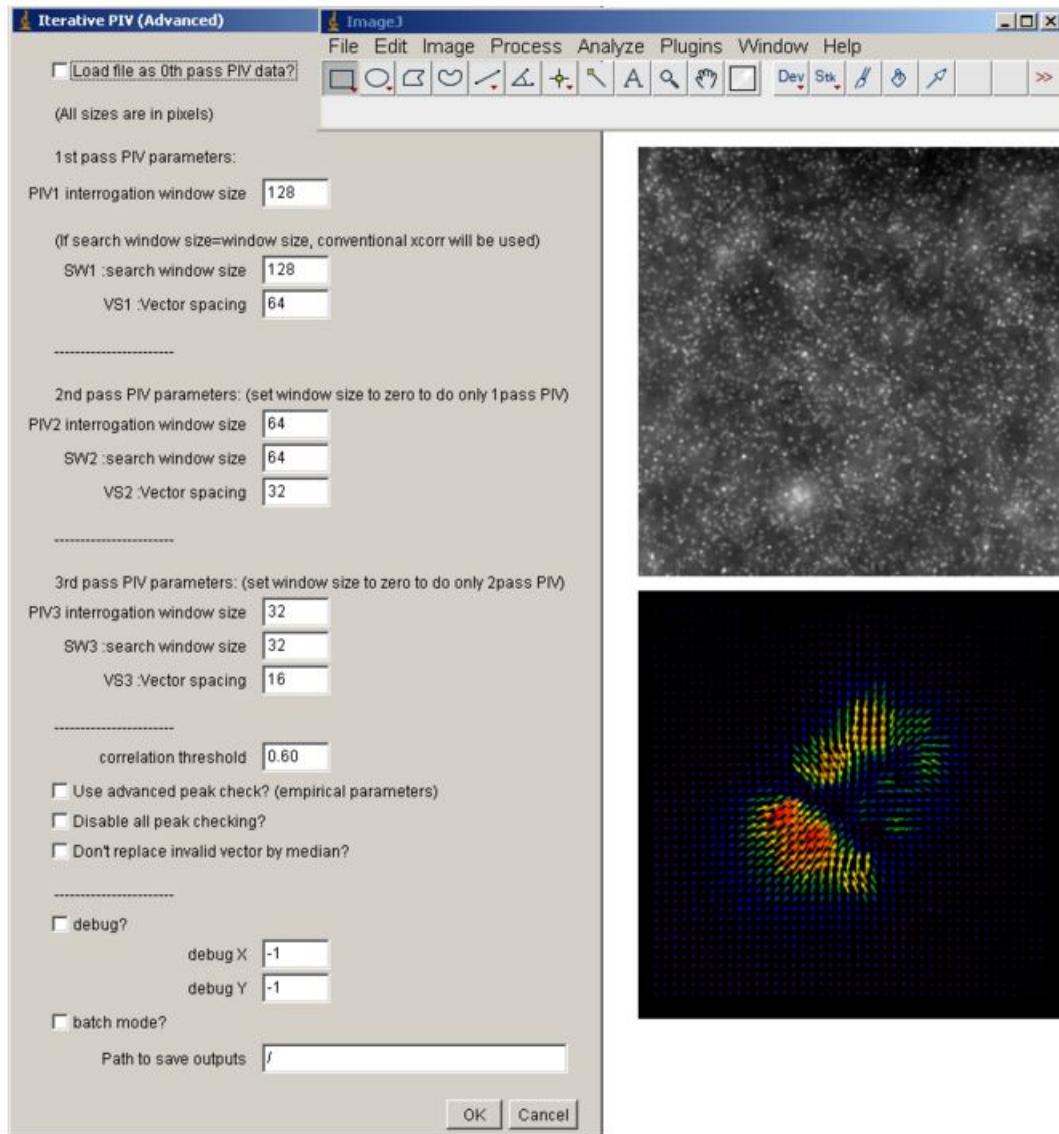


Figura 3.1: Pantalla de configuración del Plugin PIV y resultados obtenidos.

En cuanto a la usabilidad, este plugin presenta una clara pantalla de configuración. Aquí es posible definir parámetros, seleccionar el algoritmo, el tamaño de ventana y la cantidad de iteraciones que realiza el algoritmo.

Como contraparte, presenta una calidad de código bastante pobre, sin una clara y correcta modularización, lo cual reduce drásticamente la modificabilidad del mismo. Realizar un cambio sobre algún método de cálculo o introducir nuevas variables al

existente podría llevar a una amplia refactorización del código fuente.

Los resultados obtenidos luego de las pruebas presentaron errores, pero luego de la utilización de un filtro de post-procesamiento permitió acercar el resultado a lo esperado. Los mismos pueden ser observados tanto en forma analítica como gráfica.

3.2.2. Mpix

Mpix [20] es una herramienta de código abierto desarrollada en MatLab cuyas principales funciones son realizar el procesamiento de las imágenes y validar los vectores incorrectos mediante funciones de post-procesamiento. En cuanto al procesamiento, se destaca la posibilidad de optar por dos algoritmos diferentes. El primero utiliza el cálculo de la correlación cruzada para determinar el desplazamiento de las partículas. El segundo está basado en Diferencias de Mínimos Cuadrados (Minimum Quadratic Differences, MQD). En ambos casos brinda la posibilidad de realizar el procesamiento iterativo. Es decir, se realizan múltiples iteraciones del algoritmo reduciendo un determinado porcentaje el tamaño de las ventanas de interrogación para obtener una mayor resolución de los resultados.

Por otro lado, la validación se realiza sobre los datos del procesamiento, es decir, la información de la matriz de vectores de velocidad. Además, permite seleccionar tres tipos de filtrado (global, por mediana ó estándar). Del mismo modo, es posible seleccionar el modo de interpolación de los vectores a sustituir (sin interpolación, interpolación cúbica, interpolación de Kringing [13]). En esta última opción los autores recomiendan la interpolación de Kringing.

Esta herramienta cuenta con una interfaz de usuario mínima, en la cual se permite configurar las variables básicas para el procesamiento de las imágenes. El post-procesamiento no esta disponible desde la interfaz, sin embargo, las funciones pueden ser realizadas a través de línea de comandos. Los resultados pueden ser presentados tanto analíticamente como gráficamente.

En cuanto a las pruebas realizadas, el programa falló casi en todas; esto se debe a que depende en gran medida de la calidad de las imágenes utilizadas. Los mensajes de error no proporcionan demasiada información que permitiera determinar el origen de los mismos. Por otra parte, los resultados obtenidos por ambos algoritmos presentan una baja performance en relación a las demás herramientas analizadas, y en muchos

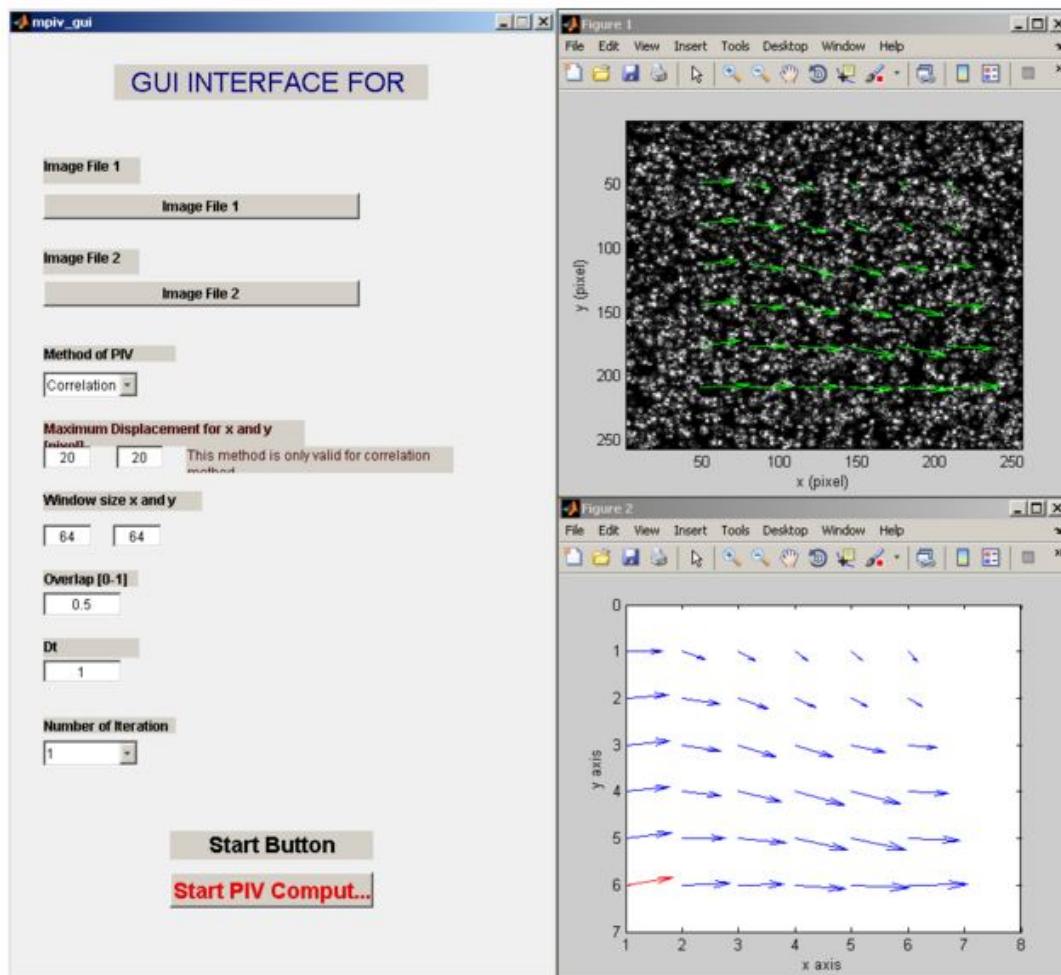


Figura 3.2: Pantalla de configuración de Mpix y resultados obtenidos.

casos fueron incorrectos. no obstante, a diferencia de las demás, permite zonas de interrogación rectangulares.

Respecto a la calidad de código, se puede observar que el mismo se encuentra correctamente modularizado, posibilitando de esta manera mayor facilidad ante la incorporación de modificaciones.

3.2.3. MatPIV

MatPIV [19] es un software de código abierto y desarrollado en MATLAB. En la etapa de procesamiento de PIV, se puede definir, tanto una máscara de procesamiento

y un sistema de coordenadas, como así tambien una región de interés restringiendo la superficie de la imagenes sobre la que se desea realizar el procesamietno.

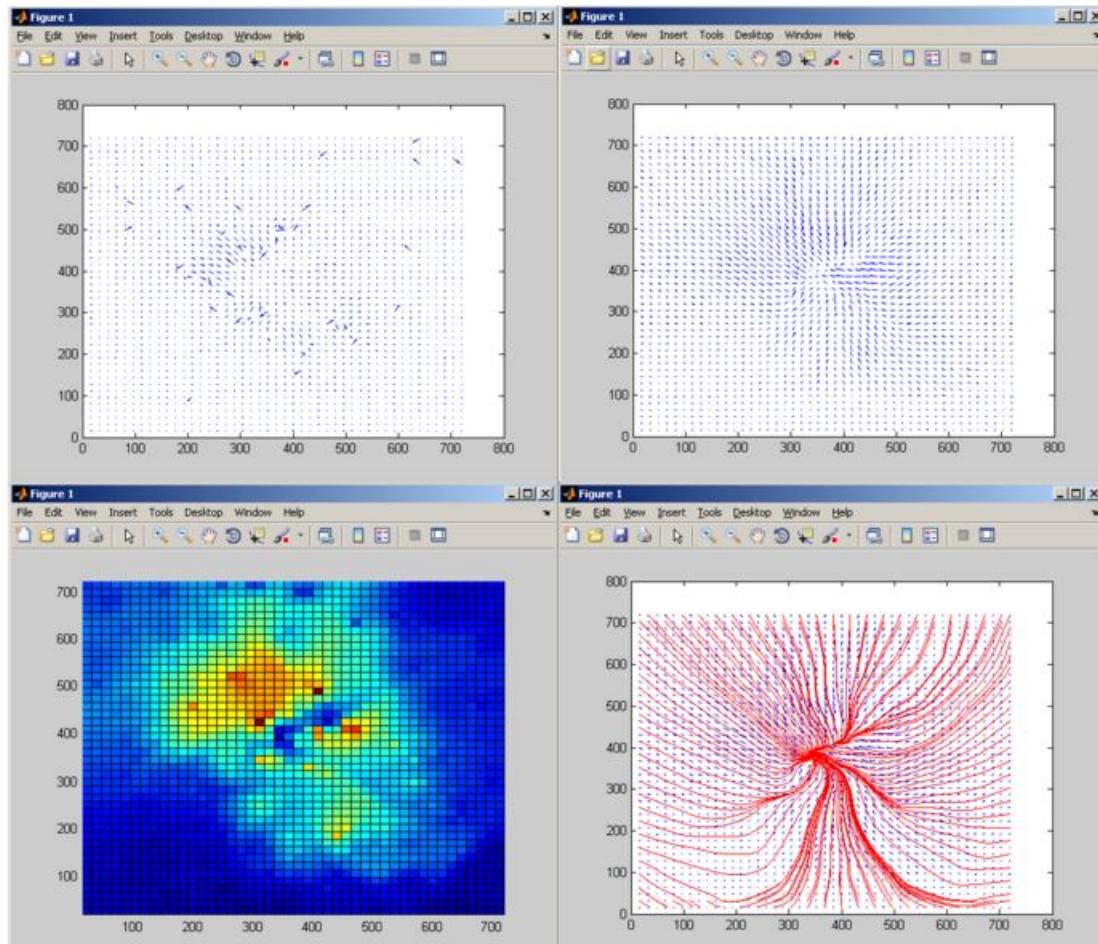


Figura 3.3: Diferentes visualizaciones de la herramienta MatPIV.

Como ventaja respecto del resto de las herramientas relevadas, se observa la implementación de diversos algoritmos de procesamiento como Correlación Cruzada, Diferencias de mínimos cuadrados y Autocorrelación utilizando FFT (*Fast Fourier Transform*).

Por otra parte, entre las funciones adicionales al procesamiento PIV de las imágenes se encuentra la opción de validar resultados por diferentes métodos: por medio de la mediana local, validación por el operador de histograma global, y a través del filtro SnR .

El programa tiene un procesamiento rápido, sin embargo, la carga manual de todos los comandos necesarios para procesar y visualizar los resultados puede ser engorroso para el usuario, ya que no es sencillo ejecutar las funciones de modo tal que el programa retorne los resultados esperados.

En cuanto a la calidad de código, se observa una correcta modularización. Esto permite ubicar fácilmente la implementación de las diferentes funciones correspondientes a los filtros de procesamiento, post-procesamiento y aplicación de máscaras.

A diferencia de las demás herramientas analizadas, MatPIV presenta diversas variantes a la hora de visualizar los resultados, como por ejemplo Gráficos de Vectores, Vorticidad, Magnitud y Streamlines.

3.2.4. JPIV

Al igual que las herramientas anteriores, JPIV es de código abierto, y en este caso implementada en Java. La misma cuenta con una interfaz gráfica donde pueden ser configurados de forma sencilla, los diferentes parámetros a utilizar durante el procesamiento.

El método utilizado para realizar PIV está basado en el cálculo de la correlación cruzada de las imágenes usando Transformaciones Rápidas de Fourier (Fast Fourier Transform, FFT), lo cual permite reducir la complejidad computacional respecto de las técnicas de correlación cruzada que utilizan convolución (Convolution cross-correlation). Sumado a esto, utiliza una API desarrollada por Oracle llamada JAI (Java Advanced Imaging), la cual está optimizada para el procesamiento de imágenes. Debido a estos factores, JPIV brinda una performance de cálculo mayor al resto de las herramientas analizadas.

Adicionalmente, esta herramienta permite aplicar filtros de post-procesamiento sobre los resultados obtenidos. El código fuente se encuentra correctamente modularizado, donde se pueden determinar las clases que intervienen en el procesamiento PIV y sus correctas responsabilidades.

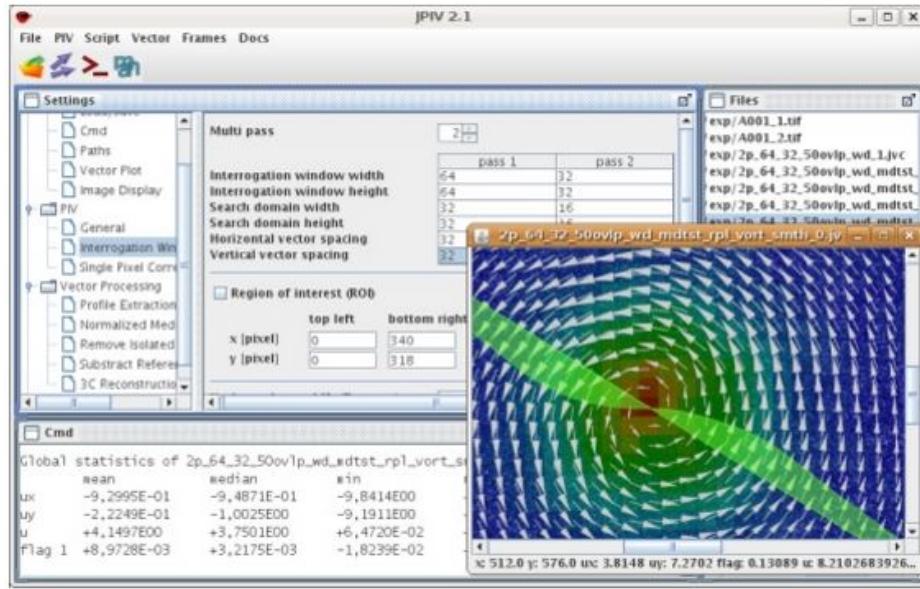


Figura 3.4: Interfaz gráfica de JPIV.

3.2.5. Comparación de las herramientas PIV Analizadas

A continuación, se presenta una tabla comparativa de las herramientas analizadas haciendo hincapié en las similitudes y diferencias que se pueden encontrar en cada una de las etapas del procesamiento.

A partir de este análisis, se puede observar una diversidad de configuraciones y métodos aplicables en las distintas etapas de procesamiento. Por este motivo surge la importancia de contar con una herramienta robusta, escalable, extensible y usable, que permita integrar fácilmente diferentes arquitecturas de software, facilitando al usuario la utilización de una configuración de procesamiento (etapas y métodos) adaptable a sus necesidades.

Herramienta/ Característica		JPIV	ImageJ+ PIV plugin	MatPIV	MPIV
Pre-Procesamiento		no	Provisto por ImageJ	no	no
Procesamiento	Tamaño de Ventana de interrogación	Configurable	Configurable	Configurable	Configurable
	Tamaño de ventana de búsqueda	Configurable	Configurable	no Configurable	no Configurable
	Ventana de interrogación rectangulares	no	no	Si	Si
	Procesamiento iterativo (Multigrid)	Configurable	Hasta 3	Configurable	Hasta 5
	Post-Procesamiento entre iteraciones	Si	Si	Si, pero no configurable	Si, pero no configurable
	Procesamiento de múltiples pares de imágenes	Si	no	no	no
	Región de Interés	Si	no	Si	no
Post-Procesamiento	Algoritmo de procesamiento	Correlación Cruzada (FFT)	Correlación Cruzada	Correlación Cruzada, Diferencias de mínimos cuadrados, Autocorrelación (FFT)	Correlación Cruzada, Diferencias de mínimos cuadrados
	Validación de Vectores	normalized median test	normalized median test y Dynamic mean test	Signal-To-noise ratio filter, Peak height filter, global filter, mean filter y median filter	Standard mean Filter, Median Filter, Global Filter
	Reemplazo de Vectores	Media de la vecindad, eliminación de vectores aislados, median filter, smooting	Media de la vecindad	Linear interpolation, Weighted interpolation, Smoothing	Linear interpolation, Cubic-spline interpolation, Kriging interpolation, Smoothing
	Visualización de resultados	Gráfico de vectores	Gráfico de vectores	Gráfico de Vectores, Vorticidad, Magnitud y Stremlines.	Gráfico de vectores
	Exportar mapa de correlación por vector	Si	no	no	no

Cuadro 3.1: Comparación de las herramientas relevadas en cuanto a sus funcionalidades.

Capítulo 4

Diseño de capa de Abstracción para PIV

En el siguiente capítulo se describe la Capa de Abstracción de la herramienta propuesta, su interfaz, y los mecanismos mediante los cuales es posible incorporar nuevas funcionalidades pertenecientes a diferentes herramientas PIV; por último, se detalla el funcionamiento básico de la capa mostrando la interacción entre la misma y las herramientas incorporadas.

4.1. Alcance de la capa de abstracción

Se planteó la necesidad de contar con una herramienta extensible que permita incorporar fácilmente diferentes funcionalidades para cada una de las etapas de PIV. Además, si bien muchas de las funcionalidades básicas ya se encuentran presentes en las herramientas de software libre, no existe ninguna de ellas que posea un abanico totalmente completo de funcionalidades, lo cual obliga a los usuarios a determinar qué herramienta utilizar según lo requiera. Por estos motivos, la solución propuesta se basó en un patrón *Layer*. Bajo este concepto, es posible agregar nuevas funcionalidades en las capas inferiores, ya sea mediante la implementación de las mismas o realizando llamados a herramientas externas cuando las funcionalidades ya se encuentren implementadas en estas.

De esta forma, se derivó en el desarrollo de una capa de abstracción y una herra-

mienta de software libre. Dicha capa tiene como principal responsabilidad proveer una interfaz uniforme e independiente de las herramientas PIV que se ejecutan por debajo de la misma. Así, como se observa en la Figura 4.1, la herramienta es capaz utilizar las funcionalidades provistas por las distintas herramientas PIV de manera transparente.

Asimismo, la capa de abstracción debe ser capaz de brindar mecanismos para agregar de forma sencilla nuevas funcionalidades, tanto de las herramientas extendidas como de aquellas que aún no fueron incorporadas.

De esta forma, se obtiene una solución que permite combinar las funcionalidades presentes en todas las herramientas PIV extendidas, a fin de contar con un software integral para el procesamiento de velocimetría por imágenes de partículas.

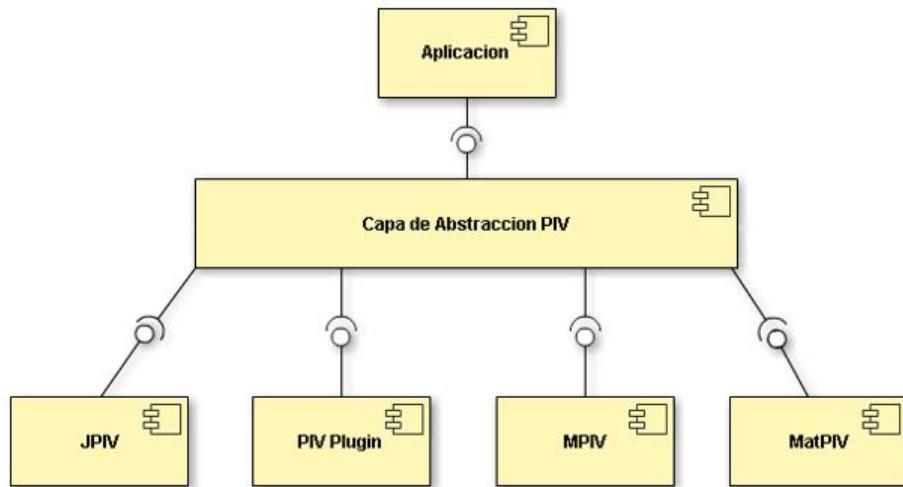


Figura 4.1: Diagrama de C&C de una aplicación comunicada con herramientas mediante la capa de abstracción.

4.1.1. Interfaz

Dadas las características del problema, se propuso una solución donde cada etapa correspondiente al procesamiento PIV (pre-procesamiento, procesamiento y post-procesamiento) se encuentra dividida en subetapas. Las mismas reciben un conjunto de datos de entrada, y generan una salida que es utilizada como entrada de la siguiente subetapa, como se muestra en la Figura 4.2. Dichos datos representan Imágenes

o Mapas de Vectores según corresponda.

4.1.1.1. Filtros

Un filtro representa una funcionalidad, ya sea presente en alguna de las herramientas extendidas o desarrollada según una necesidad del usuario. Estos filtros se aplican secuencialmente y constituyen las diferentes subetapas de procesamiento. Teniendo en cuenta que cada una de estas es independiente entre sí y, por lo tanto no respetan una interfaz en común, es necesario un mecanismo que permita adaptar las funcionalidades de cada una de ellas a la capa de abstracción mediante una interfaz compatible. Por este motivo, se implementó el patrón adapter [11]. Bajo este diseño, los filtros son los encargados de realizar la adaptación, efectuando los llamados a la herramienta correspondiente y proveyendo una interfaz común para las distintas herramientas a instanciar. En el Capítulo 5 se describe en detalle la instanciación de diferentes funcionalidades.

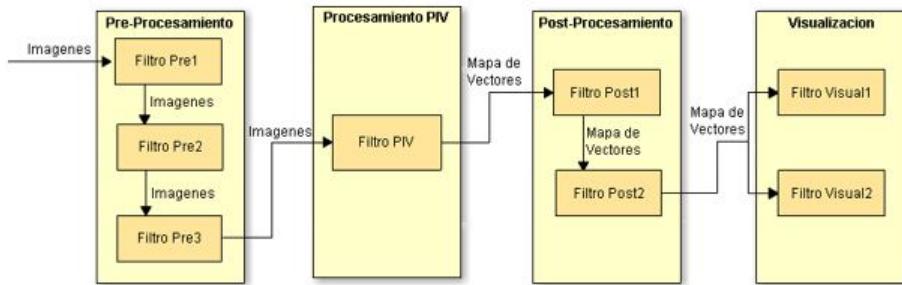


Figura 4.2: Esquema general de división del problema en Filtros.

En la Figura 4.2 se puede observar que cada una de las etapas que forman parte de PIV, reciben y generan diferentes entradas y salidas respectivamente. Para el caso de la etapa de pre-procesamiento, todos los filtros que la componen reciben un conjunto de imágenes como entrada, y producen un conjunto de imágenes como salida, siendo este último el producto de una serie de transformaciones sobre las imágenes de entrada. Posteriormente, la etapa de PIV propiamente dicha, compuesta por un único filtro, recibe el conjunto de imágenes producidos por la etapa anterior y genera el Mapa de vectores correspondiente a la velocidad y dirección de las partículas presentes en el fluido. Luego, el post-procesamiento obtiene como entrada el mapa

de vectores resultante de la etapa de procesamiento, y genera un nuevo mapa de vectores, resultado de la corrección de los vectores erróneos.

Como resultado del análisis realizado en el Capítulo 3, se pueden distinguir claramente tres tipos de filtros correspondientes a cada una de las etapas. Estos filtros fueron materializados sobre la capa de abstracción en las siguientes clases:

- ***FiltroPreProcesamiento***
- ***FiltroPIV***
- ***FiltroPostProcesamiento***

A partir de estas clases, fue posible observar y abstraer un comportamiento común, realizar el procesamiento para una entrada dada, y generar un nuevo elemento como salida. Dicho comportamiento fue representado por el método abstracto ***filtrar()***.

Como se puede observar en la Figura 4.2 existe además una etapa de visualización. Esta se distingue de las demás etapas ya que, si bien recibe datos de entrada, no produce ningún dato de salida, sino que sólo presenta los resultados de forma gráfica. Esto permitió diferenciar los filtros en dos categorías. Aquellos que reciben una entrada y producen una salida, fueron representados por una clase abstracta ***Filtro-Procesable*** (generalizando los filtros de las primeras 3 etapas). Por otra parte, los filtros que reciben una entrada pero no generan ninguna salida fueron representados por la clase abstracta ***FiltroVisualizacion***, cuya funcionalidad fue expresada a través del método abstracto ***visualizar()***.

Luego del análisis de diversos tipos de filtros aplicables a un procesamiento PIV, sobre todo en la etapa de pre-procesamiento, se observó que no todos los filtros reciben como entrada un único elemento y realizan el procesamiento sobre el mismo. En ciertos casos, la entrada puede estar dada por un conjunto de elementos. Por este motivo, se incorporó a los filtros el atributo ***cantElementosProcesables***, el cual indica la cantidad de elementos que deberá recibir como entrada. Así mismo, dado que los filtros de procesamiento deben generar una salida, se agregó el atributo ***cantElementosGenerados*** para especificar la cantidad de elementos resultantes luego de aplicar el filtro. Como se detalla en la sección 4.2.2, estos atributos se

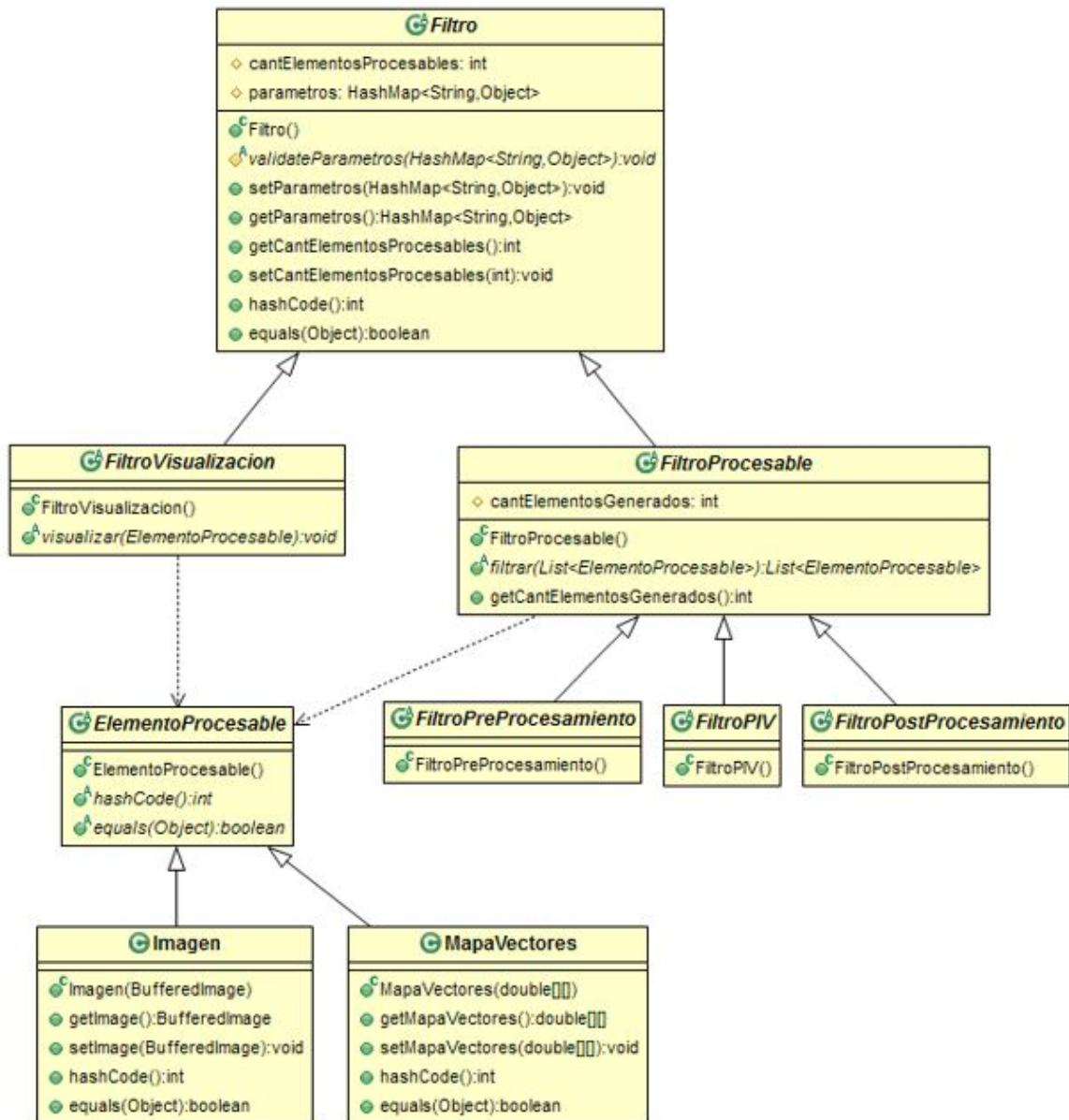


Figura 4.3: Diagrama de clases UML de Filtros y Elementos Procesables.

utilizan para determinar la cantidad de elementos del conjunto que serán procesados por cada filtro.

Por otra parte, cada filtro cuenta un conjunto de parámetros propios, los cuales pueden ajustarse en las diferentes ejecuciones del procesamiento PIV con el fin de

lograr un resultado más cercano a los valores esperados de velocidad y dirección de las partículas. Cómo se puede ver en la Figura 4.3, en la clase **Filtro** los parámetros fueron representados por medio de un **HashMap<String, Object>**, en donde el String correspondiente a la clave establece el nombre del parámetro, y el Object el valor del mismo. Este patrón de diseño, llamado Property Container [7], brinda la flexibilidad de que en cada Filtro defina la cantidad y tipo de parámetros que le sean necesarios.

Además, la clase **Filtro** cuenta con el método abstracto **validateParametros(HashMap<String, Object>, parameters)**, el cual debe ser implementado en cada Filtro que se desee incorporar a la herramienta. Este método tiene como objetivo el manejo de excepciones ante la modificación de los parámetros, determinando la validez de los mismos e informando el error correspondiente de ser necesario. El mismo es invocado por el método template [11] **setParametros(HashMap<String, Object>, parameters)** previo a guardar los datos en el *hash*. De esta forma, se evita almacenar datos incorrectos que podrían producir errores a la hora de ejecutar el filtro.

4.1.1.2. Elementos procesables de entrada y salida

Como se observa en la Figura 4.3, los filtros que interactúan con la herramienta propuesta reciben y generan un elemento representado por la clase abstracta **ElementoProcesable**. Dicho elemento procesable puede ser una **Imagen** o un **Mapa Vectores**.

En el primer caso, la **Imagen** fue representada internamente por un **BufferedImage**, el cual contiene la imagen correspondiente. Para el caso del **Mapa Vectores**, se utilizó una matriz **double[][]** que almacena en cada fila el resultado de cada ventana. Esta información incluye posición de la ventana, desplazamiento promedio de la ventana y otros parámetros propios de la herramienta. Debido a que las herramientas a extender pueden poseer una representación diferente de la información, se determinó un conjunto de datos representativo al mapa de vectores utilizado por un procesamiento PIV estándar. Bajo ese concepto, se planteó la interfaz que las herramientas deben respetar, donde las columnas de la matriz de vectores son las siguientes:

Posición en X	Posición en Y	Desplazamiento en X	Desplazamiento en Y	Módulo del desplazamiento	Velocidad en X	Velocidad en Y	Módulo de la velocidad
---------------	---------------	---------------------	---------------------	---------------------------	----------------	----------------	------------------------

Cuadro 4.1: Representación del Mapa de Vectores.

Cabe aclarar que las herramientas extendidas pueden agregar columnas adicionales a dicha matriz, pero debe tenerse en cuenta que cualquier herramienta que no prevea el uso de dicha columna en sus filtros podría producir la pérdida de dichos datos. Por este motivo, estos casos excepcionales deben ser tratados como tales, sabiendo que pueden limitar la vinculación con otras herramientas extendidas.

4.2. Arquitectura interna de la capa

4.2.1. Funcionamiento general basado en *Pipes and Filters*

Aplicar PIV puede expresarse como una secuencia de etapas de procesamiento, en donde cada una de ellas solo depende de la salida de su etapa predecesora, y todas están conectadas por el flujo de datos. Además, como se ha mencionado en el Capítulo 2, se puede aplicar la técnica de PIV para el análisis de secuencias de imágenes, con lo cual se cuenta con un formato de datos uniforme que es intercambiado a través de los filtros. Dada estas características, la capa de abstracción fue diseñada en base a la arquitectura *Pipes and Filters* [8].

El patrón *Pipes and Filters* permite dividir la tarea de un sistema en varias etapas de procesamiento secuencial. Estas etapas están conectadas por el flujo de datos a través del sistema, donde los datos de salida de un paso son la entrada del paso siguiente. Cada paso de procesamiento está implementado con un componente filtro (*filter*). Un filtro consume y entrega datos incrementalmente (en lugar de consumir todos los datos antes de producir una salida), a fin de obtener una baja latencia. La entrada de datos al sistema, la cual proviene de una fuente de datos (*source*), es un conjunto de **ElementosProcesables**. Los datos de salida son almacenados en un sumidero de datos (*sink*), el cual, de igual forma que la fuente de datos, es representado por un conjunto de **ElementosProcesables**. La fuente de datos, los filtros y la salida están conectados secuencialmente por medio de tuberías (*pipes*). Cada *pipe* implementa el flujo de datos entre etapas de procesamiento adyacentes.

La secuencia de filtros combinados por medio de *pipes* se conoce como *pipeline* de procesamiento. En la Figura 4.4 se puede observar el comportamiento básico de una arquitectura de este tipo.

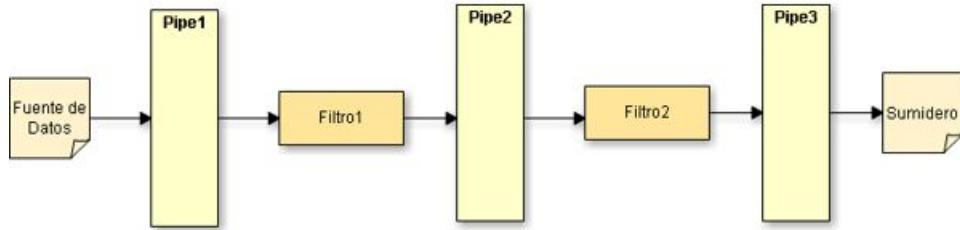


Figura 4.4: Arquitectura basada en Pipes and Filters.

El hecho de optar por una arquitectura basada en *pipes and filters* permitió conceptualizar el sistema en términos de combinación de componentes (filtros), donde cada uno es independiente de su vecino. De esta forma se logró una mayor modificabilidad del sistema, como así también aumentar la *performance* debido a la ejecución concurrente de los componentes. Dicha concurrencia se obtiene cuando una etapa de procesamiento dispone de un mínimo de elementos procesables para empezar con la ejecución.

Para comprender mejor esta característica, supongamos un ejemplo con dos filtros denominados Filtro1 y Filtro2 pertenecientes a una etapa del procesamiento. El primer filtro es capaz de procesar un sólo elemento por vez, y generar sólo un elemento de salida. Por otra parte, el Filtro2 procesa 2 elementos y genera una única salida. En la Figura 4.5 se puede observar el estado inicial de este escenario.

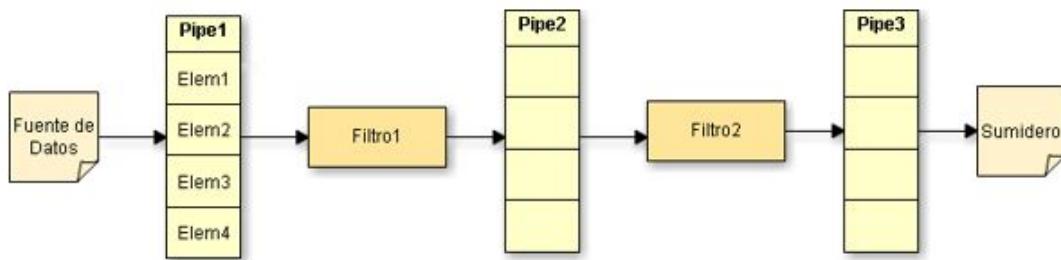


Figura 4.5: Estado inicial para un ejemplo de procesamiento con Pipes and Filters.

Teniendo en cuenta que el Filtro1 es capaz de procesar de a un elemento por vez, toma esta cantidad de elementos del *pipe* de entrada y comienza a procesar. Mientras

que el Filtro2 está esperando los dos elementos correspondientes para empezar su procesamiento (Figura 4.6).

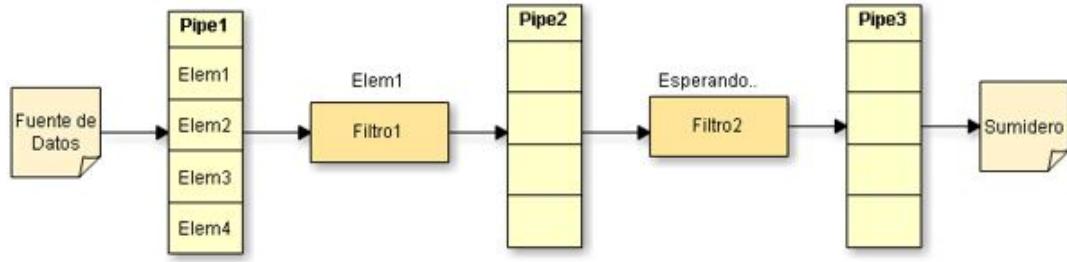


Figura 4.6: Primer ciclo del escenario de ejemplo.

Luego, el Filtro1 coloca en el Pipe2 el elemento1 ya procesado, y toma el siguiente elemento para empezar con su procesamiento. Como el Filtro2 requiere de dos elementos, queda aún a la espera del siguiente para comenzar a procesar (Figura 4.7).

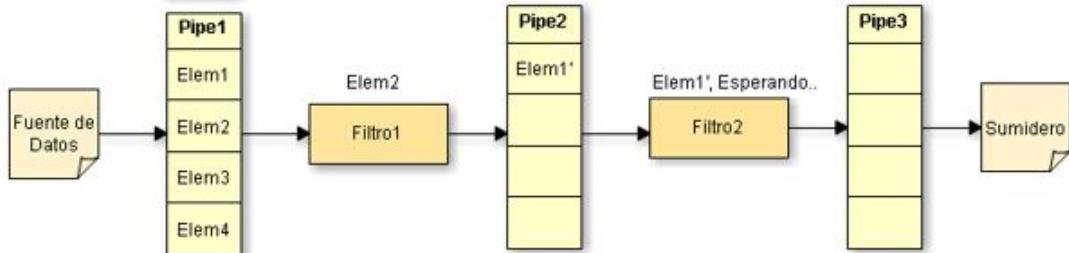


Figura 4.7: Segundo ciclo del escenario de ejemplo.

Una vez concluida la etapa anterior, el Filtro1 termina de procesar el elemento2, lo coloca en el Pipe2, y toma el elemento3 para ser procesado. En este punto, el Filtro2 cuenta con los dos elementos que necesita, y comienza su ejecución concurrente con el Filtro1, sin la necesidad de que este último haya terminado de procesar todos los elementos, por lo cual se logra un aumento la performance del sistema (Figura 4.8).

Por último, este proceso se repetirá hasta que ambos filtros hayan terminado de procesar todos los elementos como se observa en la Figura 4.9.

A partir del ejemplo se puede observar que un elemento debe esperar que todos los elementos anteriores del pipe sean procesados antes de que este sea seleccionado para comenzar su procesamiento. Si tomamos como ejemplo el caso anterior, el elemento3

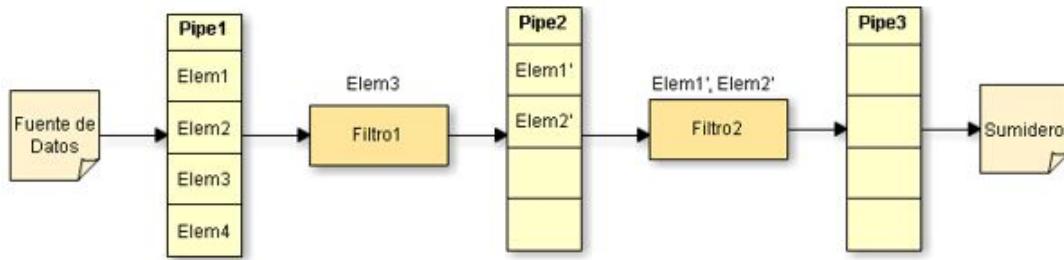


Figura 4.8: Tercer ciclo del escenario de ejemplo.

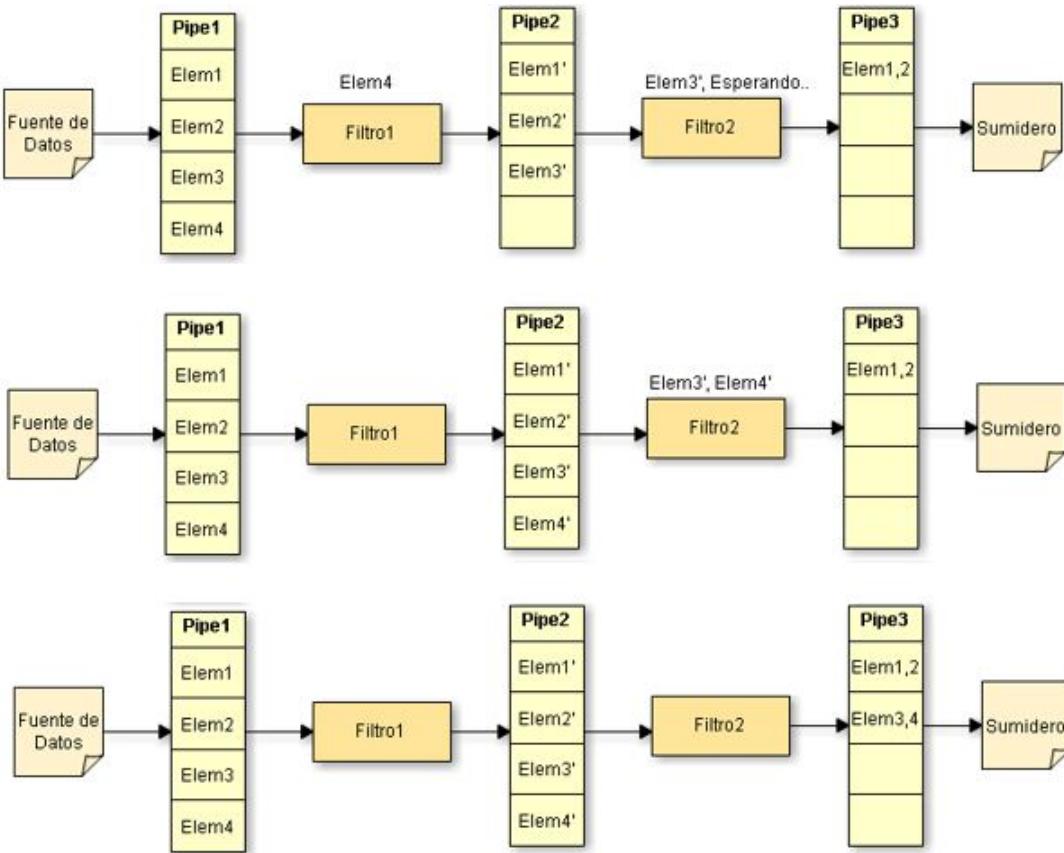


Figura 4.9: últimos ciclos del escenario de ejemplo.

no es procesado por el Filtro1 hasta que no se haya terminado el procesamiento de elemento1 y elemento2.

En resumen, se tienen todos los datos de entrada necesarios. Además, los elementos que se encuentran dentro del pipe, pueden ser procesados de manera independiente.

te (el elemento3 se podría procesar al mismo tiempo que elemento1 y elemento2, ya que el mismo no depende del procesamiento de sus antecesores). Teniendo en cuenta estas características se extendió la arquitectura propuesta considerando la paralelización de filtros de un mismo nivel. De esta forma, puede existir una instancia de Filtro ejecutando concurrentemente por cada conjunto de elementos disponibles para ser procesados, donde el tamaño del conjunto está dado por la cantidad de elementos de entrada que requiera cada filtro.

La nueva arquitectura permite aumentar la performance del sistema, donde los elementos no deben esperar en el pipe hasta que el filtro esté disponible para procesarlos, ya que hay una instancia de Filtro esperando por ellos.

Es necesario tener en cuenta que los elementos de entrada poseen un orden pre establecido. Esto se debe a que en los sistemas PIV, la entrada representa un conjunto de elementos tomados en orden secuencial y no debe alterarse, como es el caso de una secuencia de imágenes tomadas consecutivamente, donde cada imagen debe ser evaluada con la siguiente.

Dado que las instancias de un filtro se ejecutan de manera concurrente, no es posible garantizar una relación entre el orden de los elementos del pipe de entrada y la finalización del procesamiento de los mismos. Es decir, si no se coordinan las acciones, y los filtros graban a medida que finalizan, podría perderse el orden de los elementos entre un pipe, obteniendo resultados incorrectos. Por este motivo, es necesario tener en cuenta las siguientes consideraciones:

1. Cada instancia de Filtro debe tener conocimiento de la ubicación de el o los elementos en el pipe que debe procesar.
2. Cada instancia de Filtro debe tener conocimiento de la ubicación en la que los elementos procesados deberán ser almacenados dentro del pipe.

Considerando esta situación, se analiza una ejecución similar al ejemplo de la Figura 4.5 pero utilizando múltiples filtros en concurrencia.

Como se observa en la Figura 4.10, en el estado inicial se cuenta con cuatro instancias del Filtro1 encargadas de procesar todos los elementos del Pipe1. Esto se debe a que la cantidad de elementos de entrada para este filtro es de un único

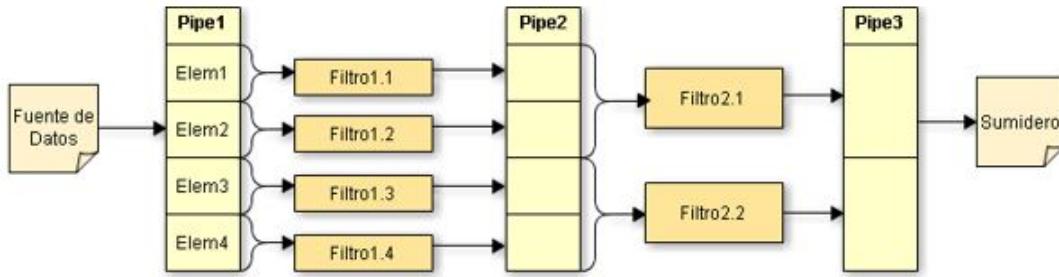


Figura 4.10: Estado inicial para un ejemplo con múltiples filtros concurrentes.

elemento. Asimismo, se cuenta con dos instancias del Filtro2, ya que el tamaño del Pipe2 es de cuatro elementos y este filtro recibe como entrada dos elementos.

Luego, cada instancia del Filtro1 puede tomar del Pipe1 el elemento que le corresponda para empezar su procesamiento, mientras que las instancias del Filtro2 esperan a tener sus datos de entrada disponibles (Figura 4.11).

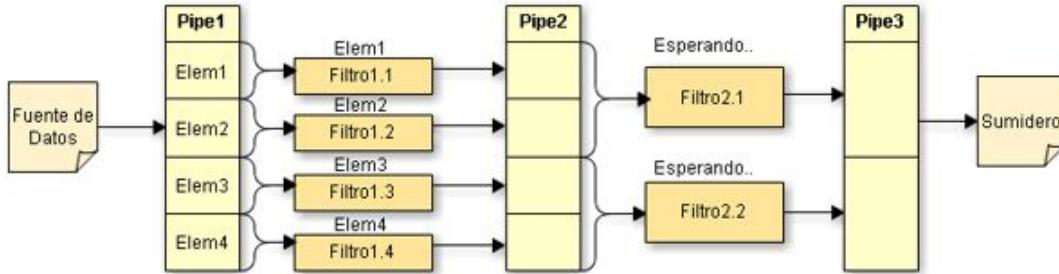


Figura 4.11: Primer ciclo del escenario de ejemplo con múltiples filtros concurrentes.

Una vez que las instancias del Filtro1 se terminan de procesar, habrán colocado los elementos correspondientes en el Pipe2. En este punto las instancias del Filtro2 cuentan con los elementos necesarios para iniciar su procesamiento de manera concurrente (Figura 4.12).

Finalmente, las instancias del Filtro2 terminan de procesar guardando los resultados en el Pipe3 (Figura 4.13). De esta manera, se logra incrementar la *performance* respecto del escenario anterior.

En conclusión, esta arquitectura no solo permite extender fácilmente las funcionalidades de las herramientas de PIV preexistentes, sino que además brinda la posibilidad de realizar procesamiento multi hilo aumentando la performance del pro-

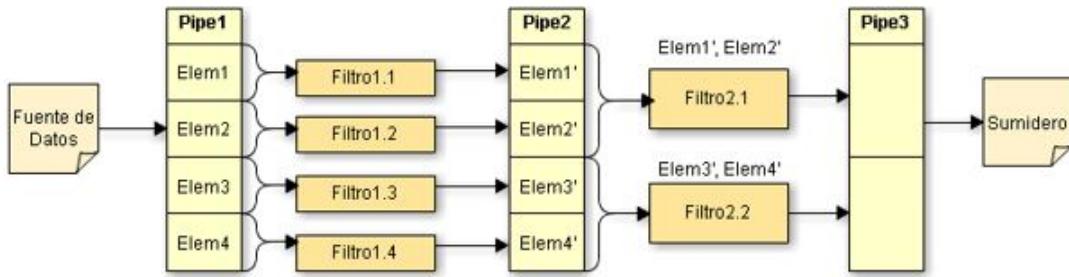


Figura 4.12: Segundo ciclo del escenario de ejemplo con múltiples filtros concurrentes.

cesamiento completo. Cabe destacar que esta característica no se encuentra presente en las herramientas PIV analizadas, salvo un caso y de manera más acotada. En la sección 6.2 se analiza un caso de estudio mostrando las mejoras de *performance* obtenidas.

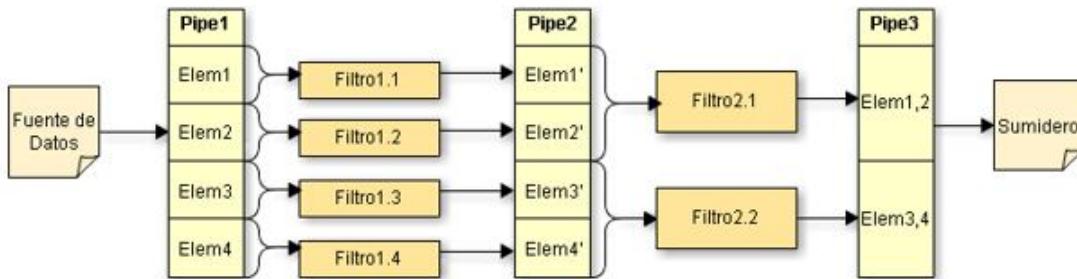


Figura 4.13: Ciclo final del escenario de ejemplo con múltiples filtros concurrentes.

4.2.2. Diseño de la arquitectura interna de la capa

4.2.2.1. Unidades elementales de la arquitectura *Pipes and Filters*

En la sección anterior se describió la capa de abstracción basada en una arquitectura *Pipes and Filters*. Como su nombre lo indica se destacan dos componentes principales, los *Pipes* y los *Filtros*.

En primer lugar, los componentes *Filters* fueron implementados mediante una clase **Proceso**, la cual representa una instancia de un Filtro en ejecución. Sus responsabilidades son: tomar los elementos que le correspondan del *Pipe*, ejecutar el

método ***filtrar()*** con dichos elementos y guardar el resultado de los elementos procesados en el *Pipe* siguiente.

Por otra parte, el componente *Pipe* fue representado por la clase ***Buffer***. Esta clase está compuesta por un arreglo de elementos procesables. La principal responsabilidad de este componente es mantener un acceso sincronizado a los datos almacenados, tanto para escribir como para acceder a los mismos. Esto se refleja mediante los métodos ***getElem(int index)*** y ***putElem(int index, ElementoProcesable elem)***.

Un punto a tener en cuenta a la hora de trabajar con múltiples hilos es el acceso ordenado a los datos. Para esto los métodos ***getElem*** y ***putElem*** fueron sincronizados mediante monitores [28] a modo de garantizar el acceso exclusivo de los diferentes hilos a un mismo *buffer*.

```
public synchronized ElementoProcesable getElem(int index) {
    while (buffer[index] == null)
        wait();
    return buffer[index];
}

public synchronized void putElem(int index, ElementoProcesable
    elem) {
    buffer[index] = elem;
    notifyAll();
}
```

Como se observa en el código, si un proceso intenta tomar un elemento del *buffer* mediante el método ***getElem*** y este aún no fue almacenado por un proceso de una etapa anterior, detiene su ejecución hasta ser notificado. Por otra parte, cuando algún proceso guarde un elemento en dicho buffer mediante el método ***putElem***, se notifica a todos los procesos que están esperando por algún elemento en este buffer. De este modo, aquellos que estaban esperando por el elemento insertado podrán accederlo y continuar su ejecución, mientras que los restantes volverán a entrar en espera. Por simplicidad, como el número de hilos que acceden al mismo *buffer* no es alto, se utilizó un solo monitor por *buffer*. En un futuro podría cambiarse a un monitor por cada elemento para evitar despertar hilos innecesariamente.

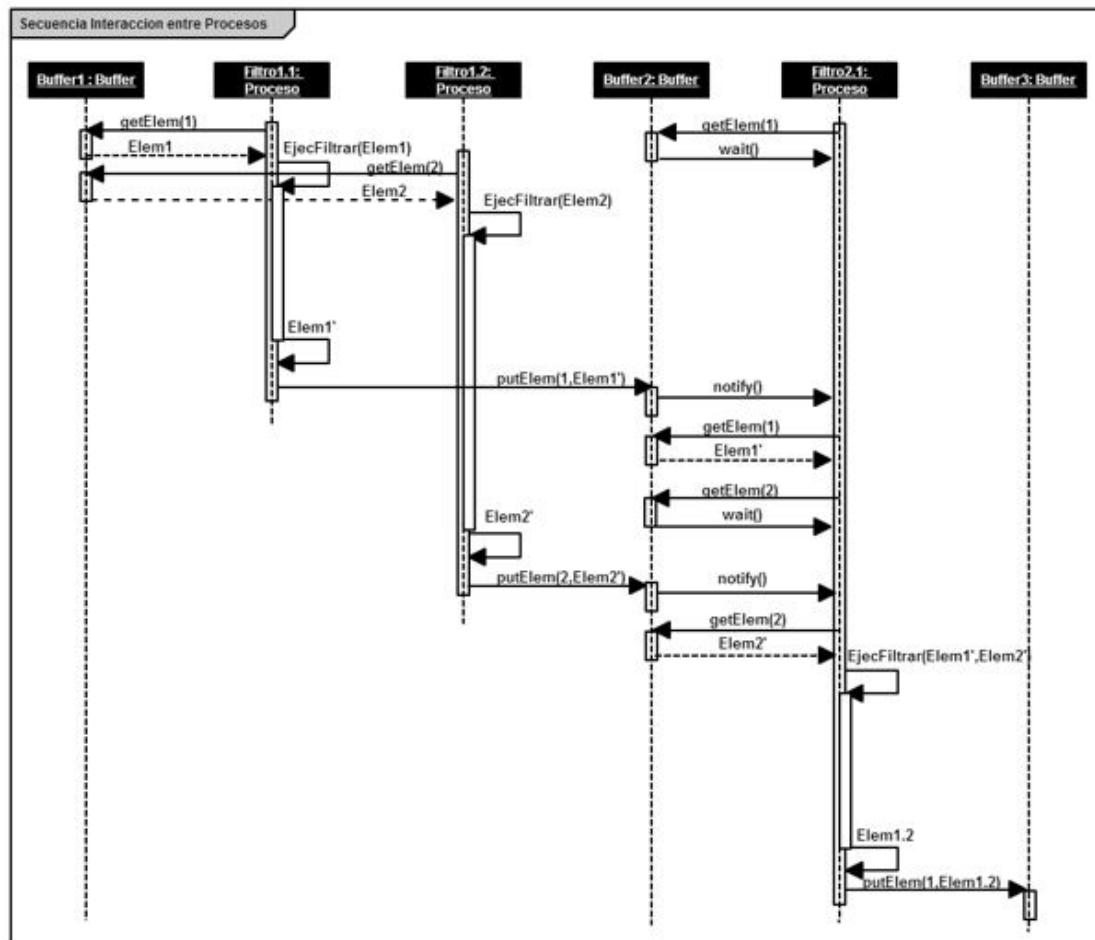


Figura 4.14: Diagrama de secuencia de interacción entre procesos.

En la sección anterior se mencionaron dos situaciones a tener en cuenta:

1. Cada instancia de Filtro debe tener conocimiento de la ubicación de el o los elementos en el *pipe* que debe procesar: Para esto la clase **Filtro** tiene un atributo **cantElementosProcesables**, que establece la cantidad de elementos que el filtro necesita como entrada. Por otro lado, la clase **Proceso** cuenta con el atributo **numeroProceso** para determinar el número de procesos creado para un mismo filtro. Ambos atributos permiten calcular la ubicación de el o los elementos a procesar, y dicha responsabilidad fue delegada a la clase denominada **Seleccionador**.

2. Cada instancia de Filtro debe conocer la ubicación en el *pipe* donde se almacena los elementos procesados: Para esto la clase **FiltroProcesable** cuenta con un atributo **cantElementosGenerados**, el cual multiplicado por el atributo **numeroProceso** de la clase **Proceso** permite determinar la posición del buffer en la que serán insertados.

Como se mencionó, la arquitectura está diseñada para aplicar PIV (involucrando sus tres etapas bien definidas) para una secuencia de elementos, donde cada filtro selecciona el conjunto de elementos a va procesar. Para modelar esta característica se creó la clase abstracta llamada **Seleccionador**, donde las clases que la extiendan deben implementar el método **seleccionar(Buffer input, Filtro filtro, int numeroProceso)**, el cual es encargado de retornar los elementos que el filtro debe tomar del *buffer* para un determinado número de proceso. A partir del Capítulo 2 se puede identificar dos formas de realizar este tipo de selección: pares consecutivos y cascada. Además del tipo de selección se debe tener en cuenta que los filtros poseen una cantidad de elementos de entrada variable. Por esto, el **Seleccionador** debe generar conjuntos de igual tamaño al conjunto de entrada requerido por el filtro.

Para implementar estos dos tipos de selección se crearon dos clases que extienden a **Seleccionador** y se denominaron **SeleccionadorPares** y **SeleccionadorCascada**.

En el caso del **SeleccionadorPares**, los elementos son seleccionados de a conjuntos consecutivos donde cada elemento pertenece a un único conjunto, es decir, si contamos con cuatro elementos numerados del 1 al 4, y un filtro cuya entrada requiere 2 elementos, se formarán dos pares, (1, 2) y (3, 4). En cambio, el **SeleccionadorCascada** se basa en formar pares de elementos, pero con la diferencia de que, exceptuando el último elemento, todos los demás serán el inicio de cada conjunto. De esta forma si contamos nuevamente con cuatro elementos numerados del 1 al 4, y un filtro cuya entrada requiere de 2 elementos, los pares formados serán, (1, 2), (2, 3), (3, 4).

Cabe aclarar que el método **seleccionar** retornará un único par dependiendo del número de proceso que solicite la selección de los elementos. Así, basado en los anteriores ejemplos, el método **seleccionar** para con un número de proceso igual a 2, retornaría el conjunto (3, 4) si se trata de un **SeleccionadorPares**, y el conjunto

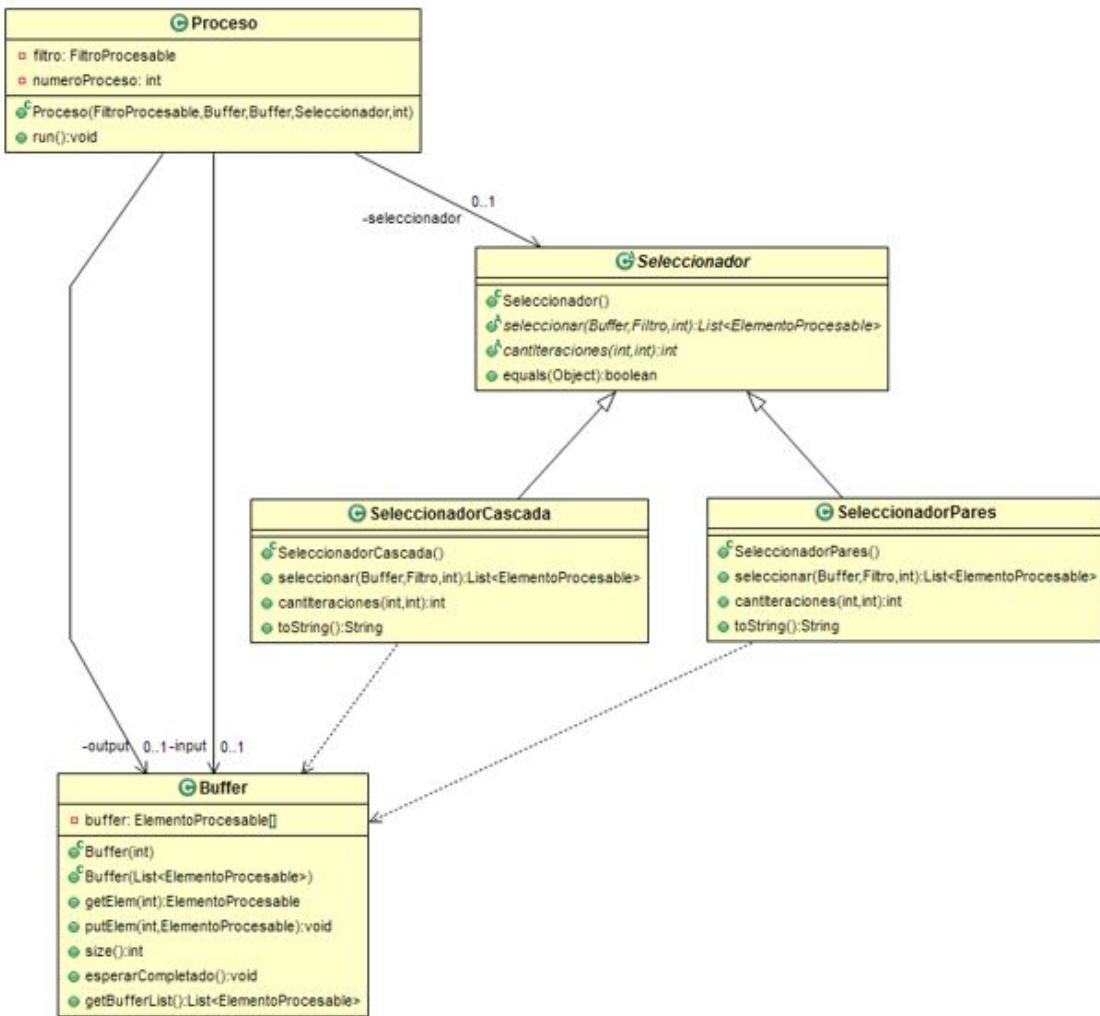


Figura 4.15: Diagrama de clases UML de Procesos, Seleccionadores y Buffers.

(2, 3) si se trata de un *SeleccionadorCascada*.

4.2.2.2. Unidades de procesamiento

En la sección anterior se mencionaron todos los componentes básicos (*buffers*, procesos y seleccionadores) para la implementación del sistema a través de la arquitectura de *pipes and filters*. Sin embargo, aún no se ha mencionado cómo dichos elementos son creados y comunicados en cada una de las etapas del sistema PIV (pre-procesamiento, procesamiento y post-procesamiento) de modo que se inicie la

ejecución de cada uno de los procesos dando lugar al resultado final. Para ello se creó una clase abstracta **Procesador**. Esta clase, a través del método **procesar(Buffer input)**, es responsable de crear y dar inicio a todos los hilos necesarios para ejecutar un conjunto determinado de filtros sobre un conjunto de elementos de entrada. Para esto, adicionalmente debe gestionar la comunicación entre los procesos creando los buffers necesarios. El resultado es el buffer de salida sobre el cual se pueden consultar los datos resultantes del procesamiento. Además, se implementó el método **procesarList(Buffer input)**, que a diferencia del anterior, retorna todos los *buffers* intermedios utilizados para el procesamiento, brindando la posibilidad de obtener resultados parciales. Esto es de utilidad, por ejemplo, para observar la evolución de las imágenes tras la aplicación de cada filtro en la etapa de pre-procesamiento.

En la Figura 4.16, se muestra un diagrama de las actividades que realizan dentro del método **procesarList**.

```
procesarList(Buffer input) {
    Crear lista de Buffers;
    Agregar input a la lista de Buffers;
    Por cada filtro f del Procesador {
        output = nuevo Buffer;
        Agregar output a la lista de Buffers;
        Determinar cantidad de procesos a generar para el
        filtro f;
        Por cada proceso a generar{
            p = nuevo Proceso(f,input,output,
                seleccionador,nroProceso);
            Iniciar p;
        }
        input = output;
    }
    retornar lista de Buffers;
}
```

De esta forma, fue posible pensar un Procesador como un conjunto de Filtros, los cuales pueden ser aplicados sobre una secuencia de elementos de entrada almacenados en un Buffer mediante los métodos procesar y procesarList.

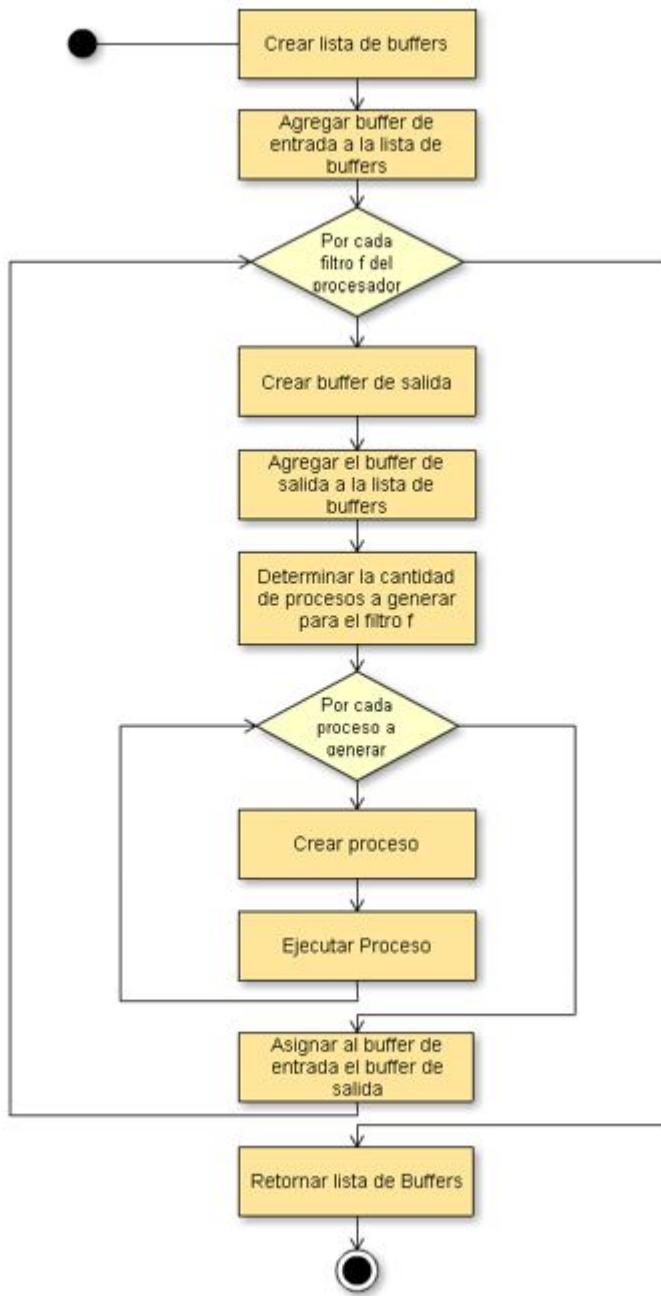


Figura 4.16: Diagrama de actividades de procesamiento.

Dado que el procesamiento puede ser dividido en tres etapas, pre-procesamiento, procesamiento PIV y post-procesamiento, y que, en cada una de las ellas solo se

deben ejecutar los filtros correspondientes a dicha etapa (carece de sentido ejecutar un filtro de post-procesamiento durante una etapa de pre-procesamiento) se implementaron tres clases correspondiente a las etapas que involucran el procesamiento. La clase **ProcesadorPIV** está compuesta por un único FiltroPIV, mientras que las clases **PreProcesador** y **PostProcesador** se componen de un conjunto de **FiltroPreProcesamiento** y **FiltroPostProcesamiento** respectivamente. De esta forma, se busca evitar un procesamiento inapropiado, ya que los filtros no pueden ser usados en etapas incorrectas.

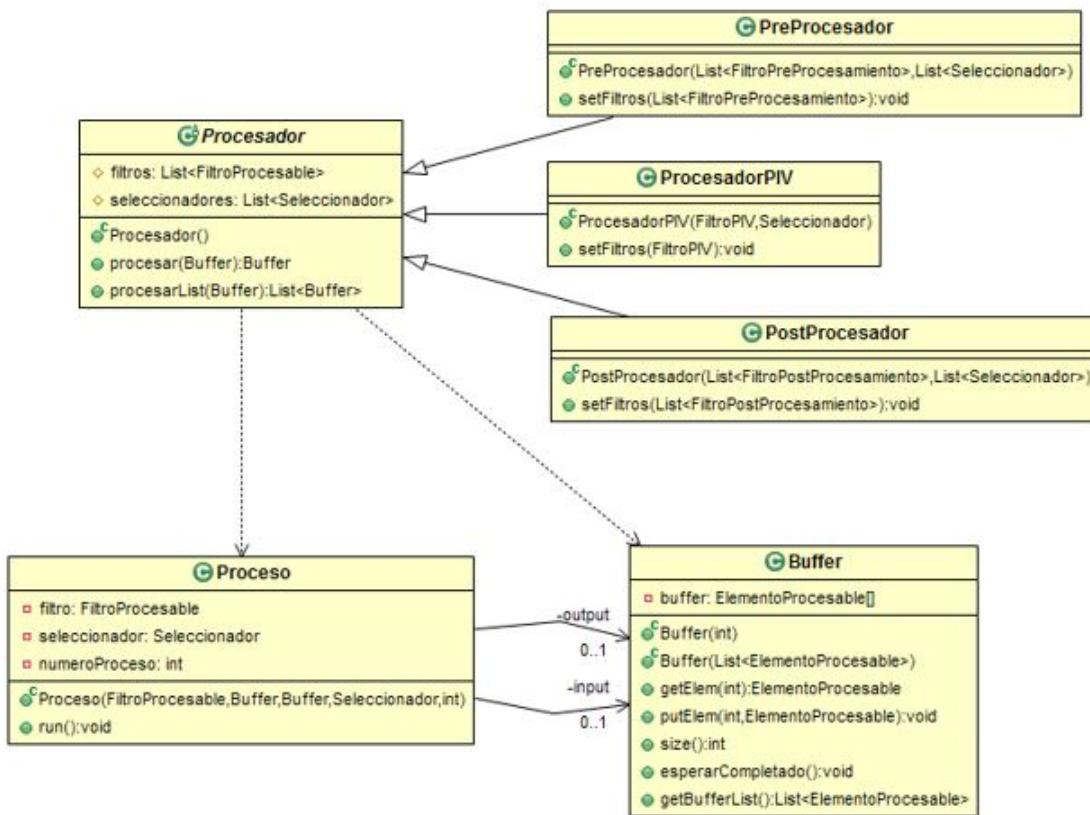


Figura 4.17: Diagrama de clases de procesadores.

4.2.2.3. Estructuras adicionales

Debido a que el procesamiento de las imágenes puede involucrar un alto costo computacional, es importante reducir el tiempo de proceso en la medida que sea

posible. Dado el problema a resolver y la solución planteada anteriormente, el procesamiento total está compuesto por una secuencia de filtros que deben ser ejecutados. Para ello, se propuso un mecanismo que, ante la modificación de algún filtro para obtener nuevos resultados a partir de un procesamiento con la misma entrada, evite el reprocesamiento de los filtros predecesores y el cálculo de resultados que ya son conocidos con anterioridad.

La implementación de dicho mecanismo cuenta con una estructura de almacenamiento caché que almacena los resultados previamente calculados. De esta forma, antes de aplicar un filtro sobre un conjunto de imágenes, se accede a la caché y se comprueba si dicho cálculo no fue realizado con anterioridad y puede ser reutilizado.

Este tipo de estructuras suele tener un espacio limitado debido a que, por razones de memoria, no es posible mantener en la caché todos los resultados calculados con anterioridad. Por este motivo, al momento de insertar una nueva entrada se controla si aún existe espacio de almacenamiento disponible, y en caso de ser necesario, hay que determinar cuál es el elemento a reemplazar para la nueva entrada. Esta característica no es un aspecto menor ya que el elemento a desalojar no podrá ser recuperado y, por lo tanto, se debe elegir un algoritmo de reemplazo que logre mantener en la estructura aquellas entradas que probablemente sean usadas con mayor frecuencia en procesamientos futuros.

Algunos de los algoritmos que se pueden aplicar a estructuras de tipo caché son los aleatorios, FIFO, MRU, entre otros [25]. Sin embargo, el más usado, y que a su vez fue elegido para la arquitectura propuesta, es el llamado LRU. La política de reemplazo de LRU consiste descartar primero los elementos menos usados recientemente. Para ello, el algoritmo lleva el seguimiento de los elementos a los que se va accediendo a fin de determinar el siguiente elemento a reemplazar.

Para implementar esta estructura se creó la clase ***CacheManager***. La misma contiene un atributo llamado ***cache*** representado a través de una tabla de hash. Como se muestra en la Figura 4.18, dicha tabla tiene como campo clave un objeto ***CacheEntry***, el cual está compuesto por un filtro y la entrada de dicho filtro, y como valor una lista de elementos procesables, correspondientes al resultado de ejecutar el filtro sobre la entrada. De esta manera, al momento de aplicar un filtro, se verifica si existe en la caché un resultado previamente calculado para esa misma entrada, con

el objetivo de obtenerlo sin el costo computacional que requiere el procesamiento.

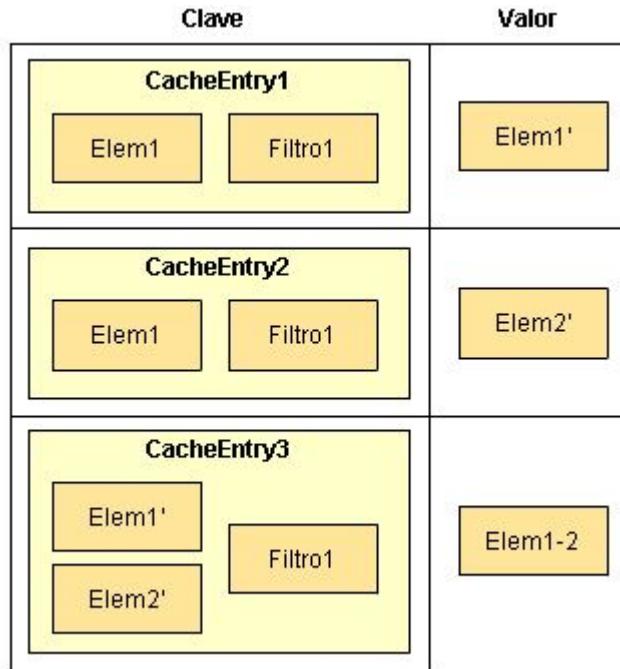


Figura 4.18: Estructura de caché implementada.

Para mantener el seguimiento de los elementos bajo el algoritmo LRU, se cuenta con una lista de CacheEntry llamada ***recentUsed***, la cual contiene las N entradas usadas recientemente ordenadas según el momento en que fueron usadas por última vez. Por lo tanto, las entradas usadas más recientemente estarán posicionadas por sobre las menos usadas recientemente. Debido a esto, es necesario actualizar la lista ***recentUsed*** ante cada acceso a la caché, posicionando la entrada solicitada por sobre el resto de las entradas. Asimismo, si una entrada no se encuentra en la caché, debe ser posible almacenarla. Para ello, si no hay espacio en la caché, debe seleccionarse una entrada a ser reemplazada. Dado que la lista ***recentUsed*** se encuentra ordenada en todo momento, es posible determinar con facilidad que el último elemento de la lista es aquel que ha sido utilizado menos recientemente y, por lo tanto, debe ser reemplazado, dejando espacio disponible que la nueva entrada.

```
get(entrada) {
    si (existe valor para esa entrada)
        Posicionar entrada en el fin de la lista resentUsed;
    retornar valor de la entrada;
}

add(nueva entrada, valor) {
    si (no hay espacio)
        Eliminar la primer entrada;
    Almacenar la nueva entrada y su valor en cache;
    Posicionar nueva entrada en el fin de la lista resentUsed;
}
```

Por último, cada proceso creado debe consultar la caché para determinar si es necesario realizar el procesamiento del filtro, y en caso de serlo, debe actualizar la misma con el nuevo resultado generado por dicho filtro. En la Figura 4.19 se muestra un diagrama de clases vinculado a las estructuras adicionales, donde la caché es implementada a través del patrón singleton [11] con el objetivo de obtener la misma instancia en todos los procesos.

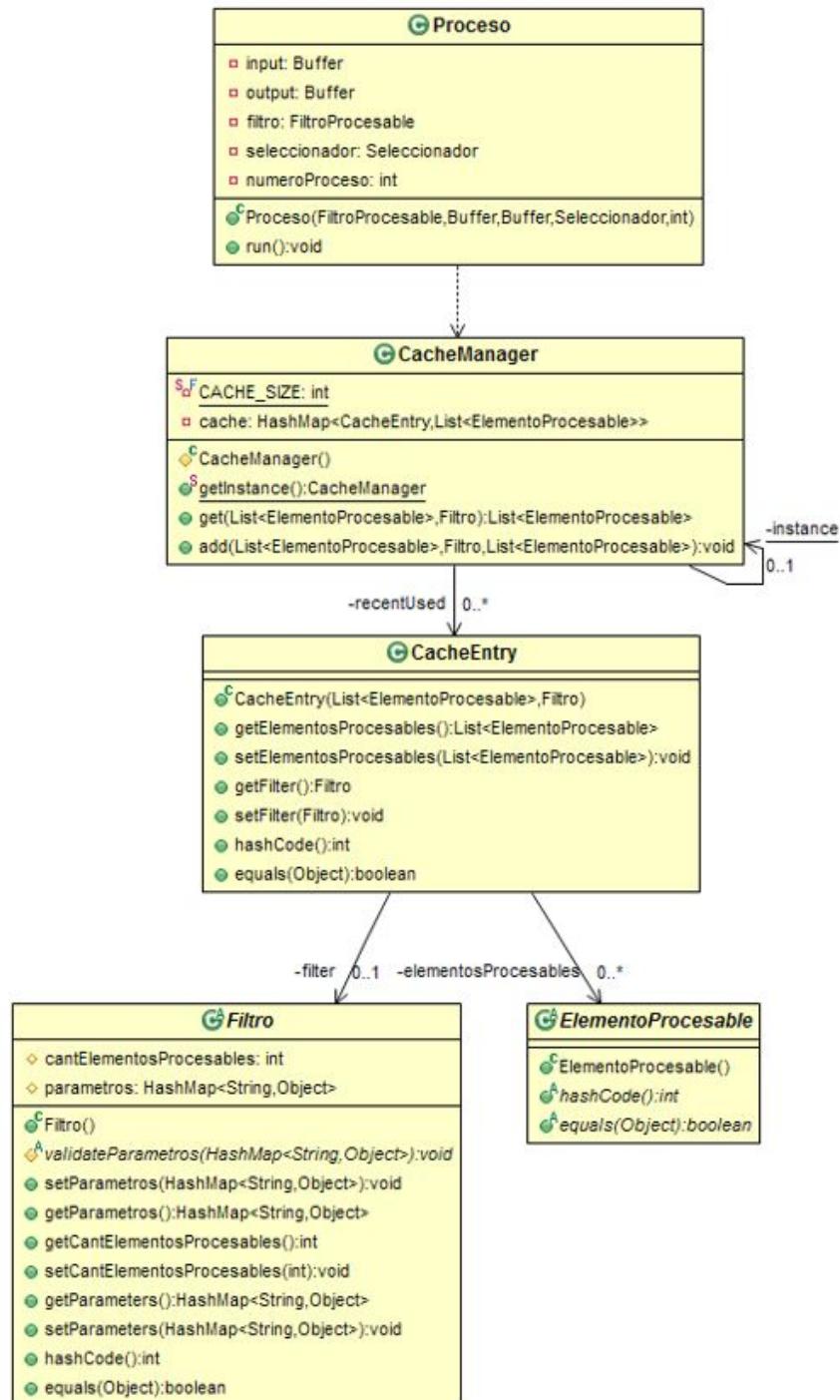


Figura 4.19: Diagrama de clases de Caché.

Capítulo 5

Instanciación

En el presente capítulo se describe cómo es posible instanciar un filtro a partir de los mecanismos provistos por la capa de abstracción propuesta. Se presentan las clases y métodos que se deben extender o implementar a fin de incorporar una nueva funcionalidad. Además, se muestra la forma de comunicar dicho filtro con una herramienta de PIV existente. Para finalizar se presenta un resumen de los pasos necesarios para que un filtro pueda ser incorporado de manera dinámica por la capa PIV.

5.1. Creación de un nuevo filtro

Como se mencionó en el capítulo anterior, se diseñó una interfaz que permite la adaptación e incorporación de filtros vinculados a las funcionalidades presentes en las herramientas extendidas. Dicho mecanismo fue implementado a través del patrón adapter [11], donde cada filtro efectúa llamados a la herramienta extendida, y debe instanciarse de las clases ***FiltroPreProcesamiento***, ***FiltroPIV***, ***FiltroPostProcesamiento*** o ***FiltroVisualizacion*** en función del tipo de filtro a modelar.

De esta forma, para instanciar un nuevo filtro es necesario realizar los siguientes pasos:

1. Crear la clase principal del filtro, extendiendo de ***FiltroPreProcesamiento***, ***FiltroPIV***, ***FiltroPostProcesamiento*** o ***FiltroVisualizacion***.

2. Implementar el constructor correspondiente, incluyendo el constructor sin parámetros con los valores por defecto.
3. Implementar el método *filtrar()* para el caso de los filtros de procesamiento, o el método *visualizar()* para los filtros de visualización.
4. Implementar el método *validateParametros()* encargado de verificar la validez de los parámetros.

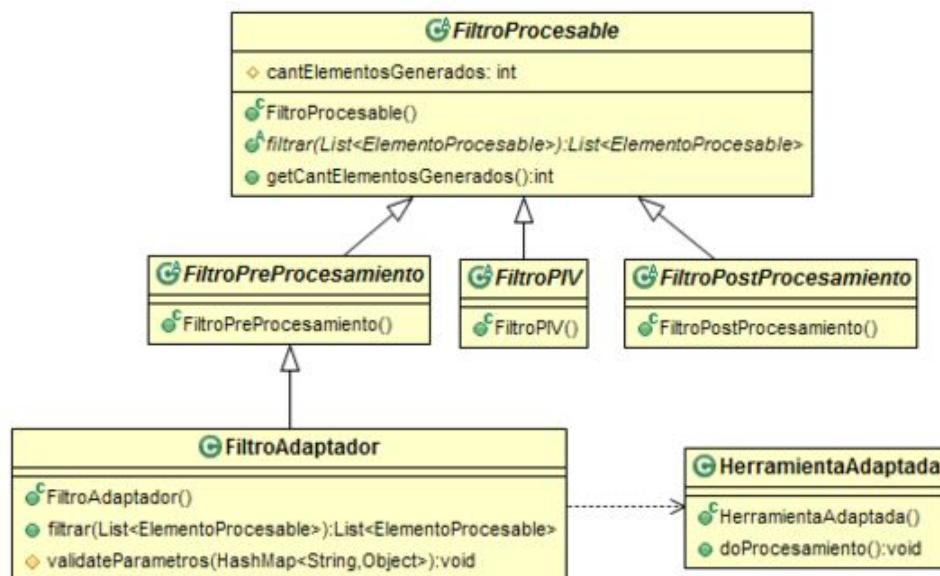


Figura 5.1: Diagrama de clases de patrón Adapter.

A continuación, se presenta un ejemplo de instanciación basado en el filtro de pre-procesamiento **Find Maxima** provisto por la herramienta ImageJ [15] con el objetivo de incorporar dicha funcionalidad al sistema. Dicho filtro determina los máximos locales en una imagen y crea una imagen binaria.

En primera instancia es necesario determinar la etapa a la que el filtro corresponde, es decir pre-procesamiento, procesamiento PIV, post-procesamiento o visualización. En este ejemplo se trata de un filtro de pre-procesamiento, por lo que es necesario crear una clase que extienda de **FiltroPreProcesamiento**.

```
public class FiltroFindMaxima extends FiltroPreProcesamiento
```

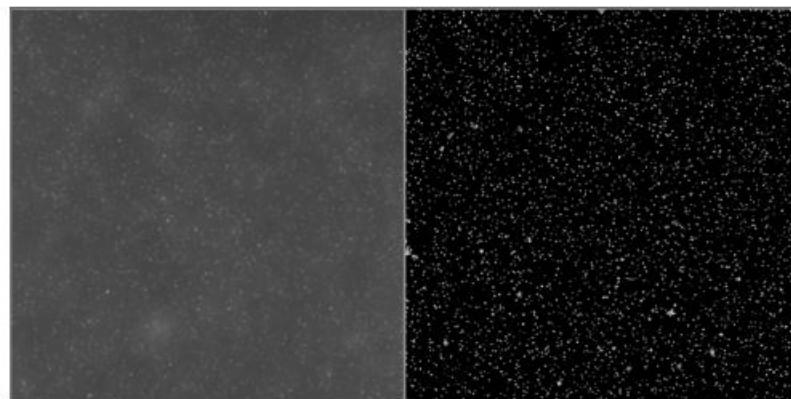


Figura 5.2: Resultado de aplicar el filtro Find Maxima sobre una imagen.

Una vez extendida la clase correspondiente, se debe proceder a la implementación del filtro propiamente dicho. Para ello deben implementarse en primera instancia los constructores correspondientes, incluyendo el constructor sin parámetros. Este último es necesario, como se explicará en la sección 5.2, para permitir a la capa crear instancias de este filtro de manera dinámica.

Los parámetros que el filtro utilice deben ser almacenados en la tabla de *hash* llamada **parametros** ubicada en la clase abstracta **Filtro**. Se debe considerar el par (clave, valor) con el nombre y valor del parámetro respectivamente. El objetivo es que los mismos puedan ser obtenidos mediante el método **getParametros()**, el cual forma parte de la interfaz provista por dicha clase. En este ejemplo los parámetros del filtro Find Máxima son NoiseTolerance y ExcludeEdges.

```
// Nombre de los Parametros
private static final String NOISE_TOLERANCE= "NoiseTolerance";
private static final String EXCLUDE_EDGES= "ExcludeEdges";

// Constructor vacio con parametros por defecto
public FiltroFindMaxima() {
    this(5, false);
}
```

```
// Constructor con parametros
public FiltroFindMaxima(int noiseTolerance, boolean
    excludeEdges) {
    this.cantElementosProcesables = 1;
    this.cantElementosGenerados = 1;
    parametros = new HashMap<String, Object>();
    parametros.put(NOISE_TOLERANCE, noiseTolerance);
    parametros.put(EXCLUDE_EDGES, excludeEdges);
}
```

Posteriormente, debe implementarse el método *filtrar()* por tratarse de un filtro de pre-procesamiento. Dicho método es el responsable de realizar el procesamiento sobre la entrada utilizando la funcionalidad provista por la herramienta que se está incorporando.

```
@Override
public List<ElementoProcesable> filtrar(List<
    ElementoProcesable> input) {
    List<ElementoProcesable> elemFil = new ArrayList<
        ElementoProcesable>();

    //Funcionalidad provista por la herramienta extendida
    ImagePlus imp1 = new ImagePlus("", ((Imagen) input.get(0)).
        getImage());
    ImageProcessor ip = imp1.getProcessor();
    int noiseTol = (Integer) parametros.get(NOISE_TOLERANCE);
    boolean excludeEdges = (Boolean) parametros.get(
        EXCLUDE_EDGES);
    ByteProcessor bProc = mf.findMaxima(ip, noiseTol, 1,
        excludeEdges);
    ImagePlus imp2 = new ImagePlus("", bProc);

    elemFil.add(new Imagen(imp2.getBufferedImage()));
    return elemFil;
}
```

Por último, se debe implementar el método ***validateParametros()***, cuyo objetivo es determinar si los valores de un conjunto de parámetros son válidos para el filtro. El mismo es llamado por el método template ***setParametros()*** provisto por la clase Filtro, a fin de evitar que se almacenen datos inválidos o incorrectos en la tabla de hash. Este mecanismo se realiza a través del manejo de excepciones, por lo que dicho método debe lanzar una excepción ***FilterException*** con el mensaje correspondiente.

```
@Override
public void validateParametros(HashMap<String, Object>
parameters) throws FilterException {
    if (((Integer) parameters.get(NOISE_TOLERANCE)) < 0)
        throw new FilterException("El parámetro " +
NOISE_TOLERANCE + " debe ser mayor o igual a 0"
);
}
```

5.2. Instanciación de un nuevo filtro

Una vez creado el conjunto de filtros, los mismos deben poder ser añadidos a la herramienta. A fin de brindar mayor extensibilidad se optó por implementar un mecanismo que permita cargar los filtros sin necesidad de recompilar, de modo que, si se desean agregar nuevos filtros luego de haber compilado la aplicación, solo requiera iniciarla o recargar los filtros en caso de que ya esté iniciada. Para esto, los filtros son añadidos a través de un archivo con extensión JAR, el cual es cargado por la aplicación con el objetivo de poder usar los filtros en el procesamiento de las imágenes.

El proceso de carga de los filtros se delegó al método ***loadFilters()*** de la clase ***PluginFilterManager***. Este método recorre el directorio donde se encuentran los archivos *.jar que contienen los filtros a agregar, y los añade al objeto ***filtersClassLoader*** donde luego son buscadas al momento de crear las instancias. Asimismo, este directorio está predefinido con anterioridad a través de la variable de configuración ***Settings.filtersPath***, cuyo valor por defecto es el directorio "filtros" ubicado

dentro de la ruta de ejecución de la aplicación.

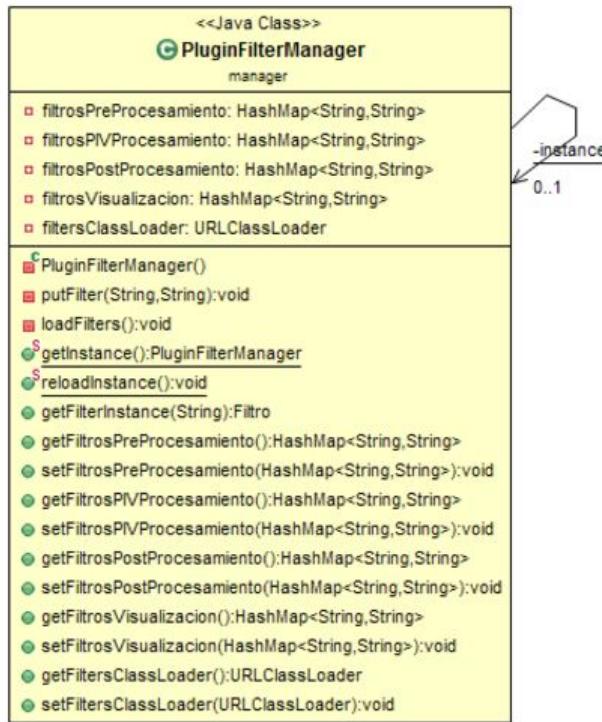


Figura 5.3: Diagrama de Clase PluginFilterManager.

Por otra parte, cada uno de los archivos `*.jar` debe tener en su interior, además del conjunto de filtros, un archivo de configuración, llamado `filters.config`, el cual debe exponer una lista con el conjunto de filtros que se desean cargar en el sistema. Cada línea de dicho archivo debe corresponderse con un filtro existente y debe estar organizada en dos campos separados por el símbolo “:” de la siguiente forma:

[package.nombreClase]:[Nombre del filtro]

Cuadro 5.1: Estuctura del archivo de configuración.

donde el significado de cada uno de los campos es el siguiente:

- Primer campo: especifica el nombre de la clase de uno de los filtros existentes en el paquete de filtros. Es importante incluir en este campo el nombre del

paquete al cual pertenece la clase, ya que de otra forma los filtros no podrán ser cargados por la herramienta.

- Segundo Campo: especifica el nombre correspondiente al filtro, que será mostrado en la interfaz gráfica.

El archivo resulta útil para evitar el recorrido de todos los directorios contenidos dentro del JAR buscando las clases correspondientes a los filtros a añadir.

Luego, por cada filtro existente en dicho archivo de configuración, se agrega, mediante el método *putFilter()*, una entrada en el HashMap correspondiente dependiendo de la superclase a la cual pertenece el filtro a añadir. De esta forma, los HashMap *filtrosPreProcesamiento*, *filtrosPIVProcesamiento*, *filtrosPostProcesamiento* y *filtrosVisualizacion* están compuestos por el par clave-valor, correspondiente al nombre de la clase y el nombre del filtro de todos los filtros incorporados a partir de los archivos de configuración.

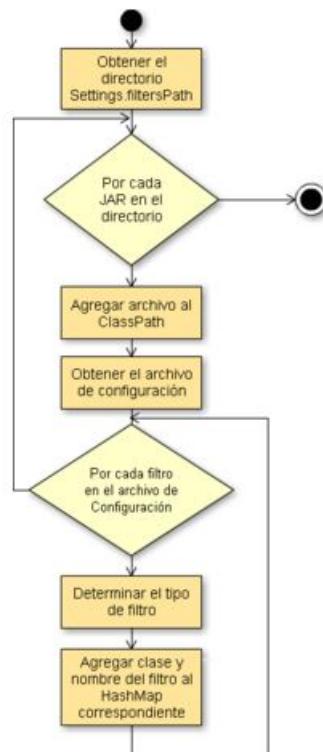


Figura 5.4: Diagrama de actividades de la carga de filtros.

Continuando con el ejemplo anterior, considerar la siguiente estructura de paquetes con algunos filtros adicionales a modo de ejemplo.

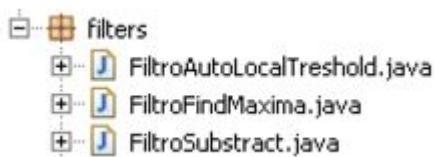


Figura 5.5: Ejemplo de estructura de paquetes.

El archivo de configuración incluido dentro del JAR debe tener la siguiente información para que los filtros sean cargados por la aplicación.

```
filters.FiltroFindMaxima:Find Maxima  
filters.FiltroSubstract:Substract  
filters.FiltroAutoLocalTreshold:Auto Local Treshold
```

Cuadro 5.2: Ejemplo de archivo de configuración.

Contar con el objeto ***filtersClassLoader*** del tipo *URLClassLoader*, permite crear, en tiempo de ejecución, instancias de clases que no se encuentran dentro del *BuildPath* del sistema. Esto posibilita extender la funcionalidad ofrecida sin necesidad de recompilar la aplicación completa.

Una vez cargados los archivos *.jar en el *ClassLoader*, es posible obtener una instancia de cualquier filtro a partir de su nombre de clase mediante el método *getFilterInstance()*. Es decir, si se desea, por ejemplo, obtener una instancia del filtro ***FindMaxima***, se debe llamar al método utilizando como parámetro el nombre de la clase de la siguiente forma:

```
getFilterInstance("filters.FiltroFindMaxima")
```

5.3. Consideraciones adicionales

Durante el desarrollo del sistema surgieron algunas dificultades relacionadas con la incorporación de nuevas herramientas. Una de ellas fue la necesidad de contar con un conjunto de archivos organizados en una estructura de directorios específica

requerida para la correcta ejecución de la herramienta. Esta característica dio lugar a diseñar algún mecanismo mediante el cual dichas herramientas cuenten con todos los recursos necesarios para su ejecución, de forma tal que la capa de abstracción pueda extenderlas correctamente manteniendo la información en una estructura ordenada. Para ello se decidió agregar a la estructura de directorios del sistema la carpeta **resources**, dentro de la cual se almacena toda la información adicional necesaria para la ejecución de las herramientas extendidas.

Por otra parte, se presentó otra situación vinculada con el *ClassLoader*. Si bien este objeto permite crear instancias a partir de un archivo JAR (*.jar) que no se encuentre dentro del *BuildPath* del sistema, al momento de cargar el archivo JAR, no agrega las librerías adicionales que este requiere recursivamente. Las mismas tampoco pueden estar en formato JAR incluidas dentro del archivo JAR original. El *ClassLoader* requiere que todas las clases necesarias para la ejecución de los filtros se encuentren dentro del archivo JAR en formato *.class. Para esto todas las dependencias utilizadas deben ser extraídas, y luego incluidas dentro del archivo JAR que el *ClassLoader* carga.

Capítulo 6

Análisis de resultados

En este capítulo se proponen diferentes casos de estudio relacionados con problemas de flujos de fluidos. El objetivo de obtener un análisis comparativo de los valores resultantes respecto de los valores esperados, y mostrar las ventajas de la utilización de múltiples herramientas encapsuladas dentro de la capa de abstracción. Finalmente, se describen los pasos necesarios para la configuración y utilización de la herramienta propuesta.

6.1. Caso de estudio 1: Desplazamiento manual de una imagen

En primera instancia, se optó por validar la herramienta a través de un caso de estudio simple, en donde se pueda visualizar y determinar con facilidad el movimiento de las partículas. Para realizar este caso de estudio se tomó una imagen obtenida a partir de un caso de test proporcionado por una de las herramientas extendidas [23]. Se realizó un desplazamiento de la imagen 5 píxeles hacia la derecha y 5 píxeles hacia abajo, como se muestra en la Figura 6.1.

Dado que en este caso el desplazamiento fue realizado de forma manual, los vectores resultantes de un procesamiento PIV deben tener una magnitud de 5 píxeles en ambas coordenadas X e Y.

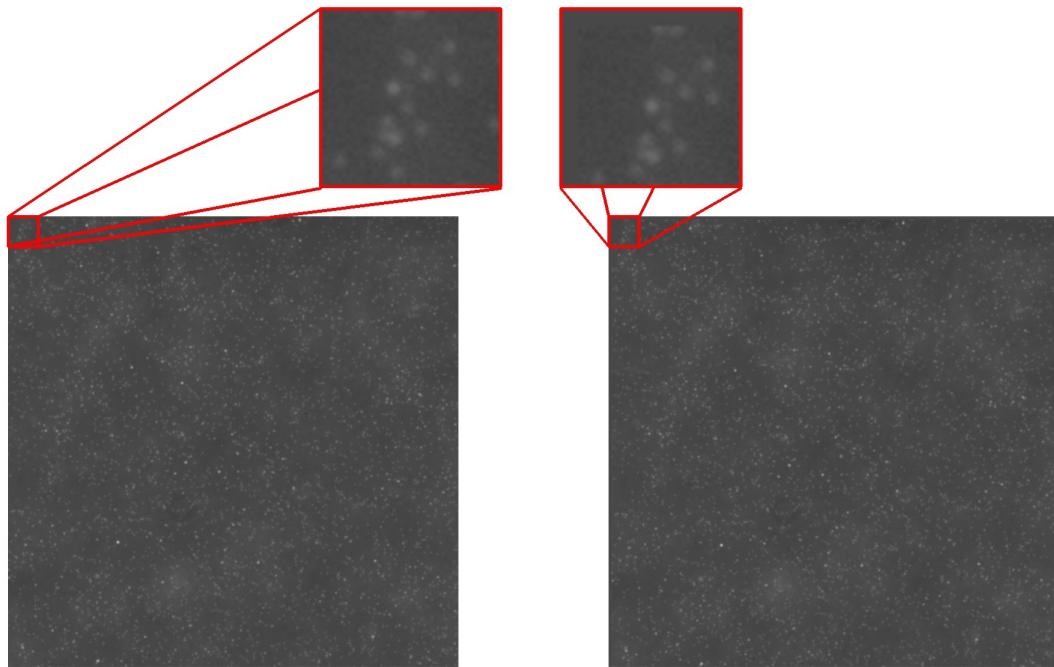


Figura 6.1: Imágenes utilizadas para el Caso de estudio 1.

6.1.1. Procesamiento PIV

Primeramente se comenzó por analizar el flujo a través de un procesamiento de correlación cruzada provisto por una de las herramientas extendidas, en la sección 5.1 y 5.2 se explican los procedimientos necesarios para implementar e incorporar un nuevo filtro a la herramienta propuesta. Para realizar este procedimiento sobre la aplicación desarrollada se debe comenzar por seleccionar el conjunto de imágenes a ser evaluadas (Figura 6.2).

La capa de abstracción no establece limitaciones respecto a la cantidad, tamaño y formato de las imágenes. Sin embargo, no es posible garantizar que las herramientas extendidas soporten dichas características. De esta forma, los requisitos de tamaño y calidad de las imágenes quedan sujetos a los filtros que sean utilizados, mientras que la cantidad de imágenes que pueden ser procesadas concurrentemente depende directamente de los recursos del subsistema sobre el que se esté ejecutando.

Una vez seleccionadas las Imágenes del sistema de archivos, las mismas se podrán visualizar a modo de lista en el panel “Imágenes” como se observa en la Figura 6.3.

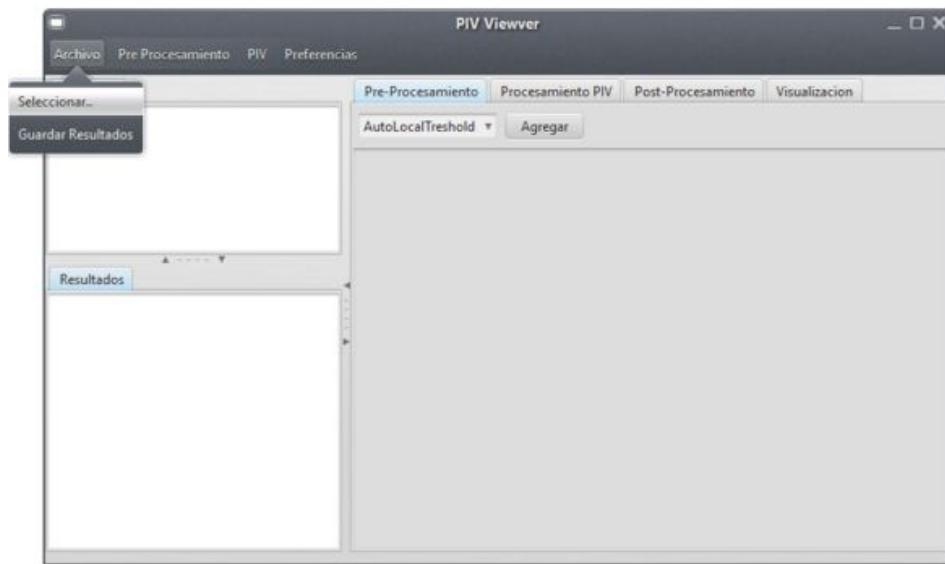


Figura 6.2: Selección de imágenes utilizando la herramienta.

Posteriormente para realizar el procesamiento se debe dirigir a la pestaña “Procesamiento PIV” para seleccionar y configurar el filtro PIV a utilizar.

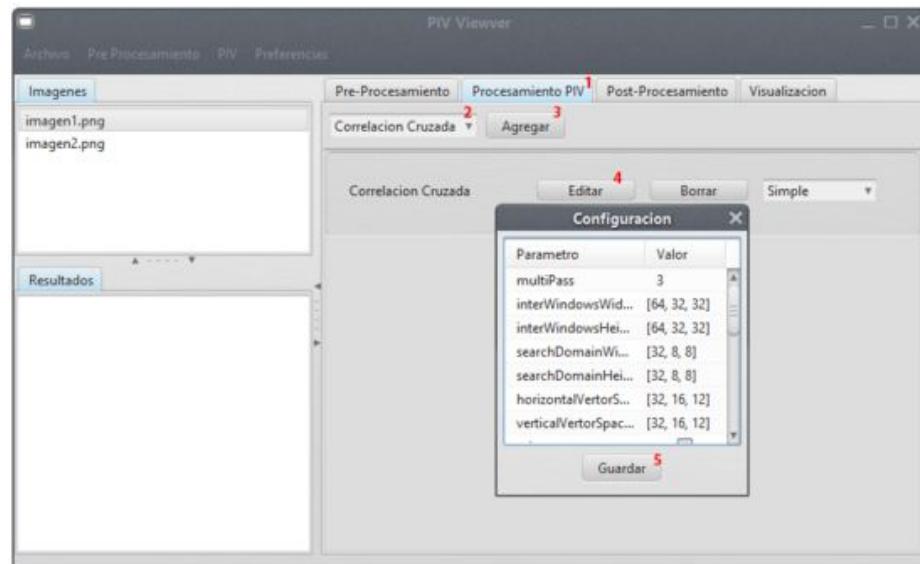


Figura 6.3: Selección y configuración de un filtro de procesamiento PIV.

Para este análisis se utilizó un filtro de correlación cruzada el cual extiende funcionalidad provista por la herramienta JPIV [16]. Entre los parámetros más relevantes

se destaca el uso de 3 pasadas con un tamaño de ventana de correlación para la pasada final de 32x32 pixeles y el uso de filtros de post-procesamiento entre las pasadas, como reemplazo de vectores inválidos por la media y *smoothing*.

Además, se extendió la representación gráfica del mapa de velocidades provisto por la herramienta JPIV, de forma que permita apreciar claramente el mapa de velocidades resultante del procesamiento PIV.

Teniendo en cuenta la naturaleza del caso de estudio, se implementó un filtro de visualización adicional que calcula el valor del desplazamiento medio en X e Y sobre toda la imagen y lo muestra por pantalla con el objetivo de realizar un análisis del valor real esperado respecto del valor obtenido mediante el cálculo.

Para agregar un filtro de visualización, sobre la pestaña “Visualización” se debe seleccionar y editar, en caso de ser necesario, los filtros que desean ser utilizados (Figura 6.4).

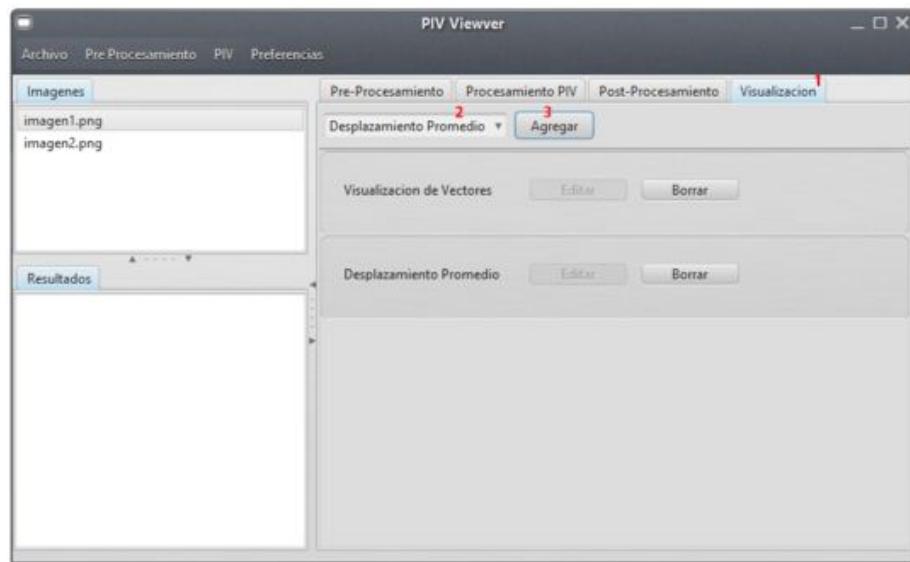


Figura 6.4: Selección y configuración de un filtro de Visualización.

Una vez configurados los filtros que van a utilizarse, debe seleccionar desde el panel “Imagenes” las imágenes que desea procesar, y dirigirse a la opción del menú “PIV → Ejecutar PIV” (Figura 6.5).

Luego de ésta operación, la herramienta comienza a procesar las imágenes según los filtros configurados. Cuando este proceso concluye se pueden observar en pantalla

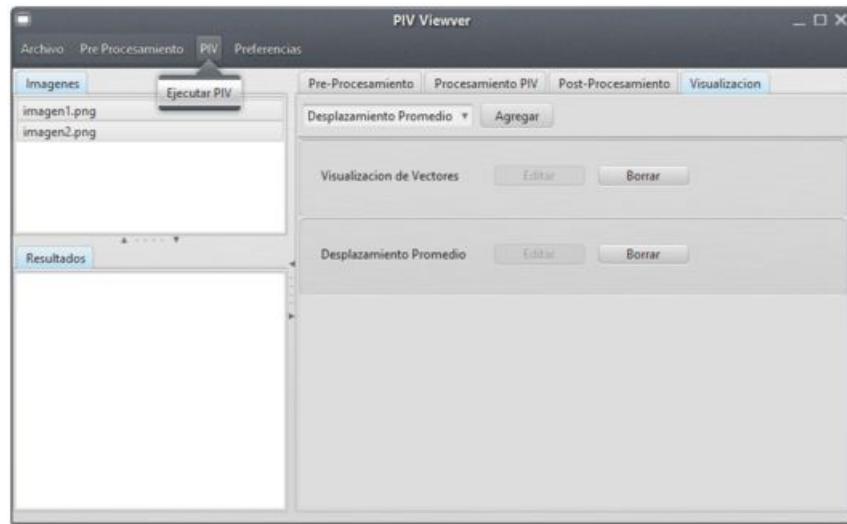


Figura 6.5: Ejecutar PIV utilizando la herramienta.

los resultados representados de acuerdo a los filtros de visualización previamente seleccionados. En este caso se mostrarán los resultados como se observa en la Figura 6.6.

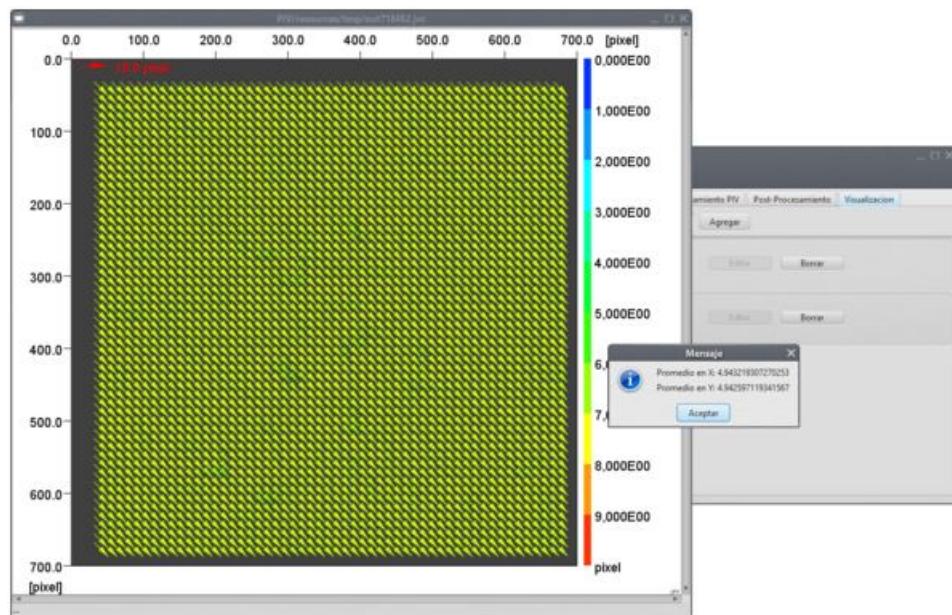


Figura 6.6: Resultados de los filtro de visualización tras la aplicación del procesamiento PIV.

Finalmente, para verificar los resultados se calculó el promedio de todos los vectores, cuyo valor es de 4.943 y 4.942 en X e Y respectivamente.

6.1.2. Procesamiento PIV con Pre-procesamiento

El siguiente paso consistió en tratar de mejorar los resultados, y para ello se utilizaron dos filtros de pre-procesamiento provistos por la ImageJ [15]. En el Capítulo 5 se describe la incorporación de filtros en el sistema desarrollado.

Primeramente se seleccionó la técnica de pre-procesamiento CLAHE detallada con anterioridad en la Sección 2.4.3.

Para agregar dichos filtros en la herramienta desarrollada, sobre la pestaña “Pre-Procesamiento” se selecciona el filtro a utilizar (Figura 6.7).

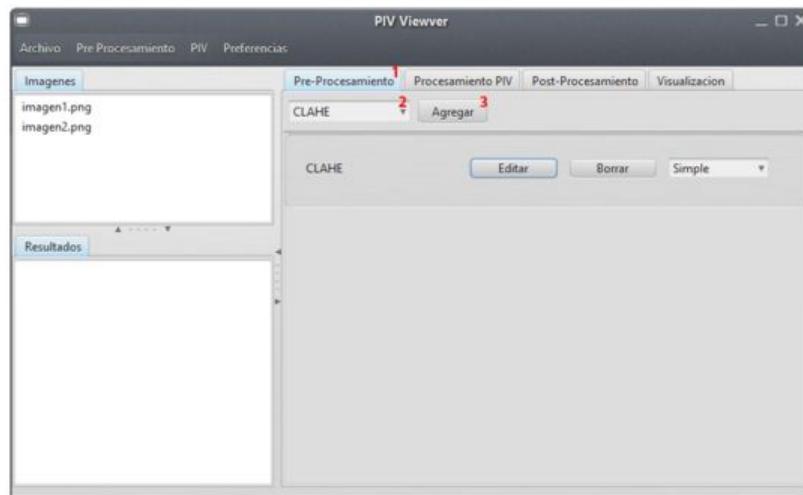


Figura 6.7: Selección y configuración de un filtro de Pre-procesamiento.

Así mismo, es posible previsualizar el efecto de la etapa de pre-procesamiento sobre las imágenes sin necesidad de aplicar todas las etapas del procesamiento. Para acceder a ésta característica se debe utilizar la opción de menú “Pre Procesamiento → Previsualizar”.

A partir de la aplicación del filtro mencionado se obtuvo un desplazamiento promedio en X de 4.950 y un desplazamiento promedio en Y de 4.949, lo cual muestra una mejora respecto de los resultados obtenidos sin aplicar una etapa de pre-procesamiento.

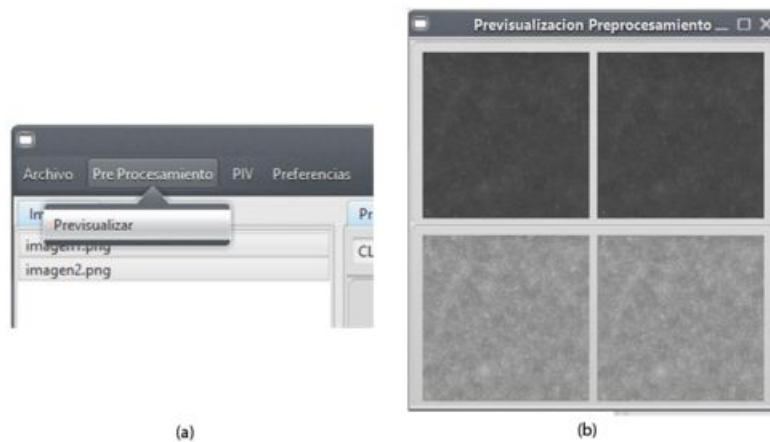


Figura 6.8: Previsualización de la etapa de pre-procesamiento.

De forma similar a la anterior, se analizó la aplicación del filtro de pre-procesamiento llamado High Pass (Figura 6.9).

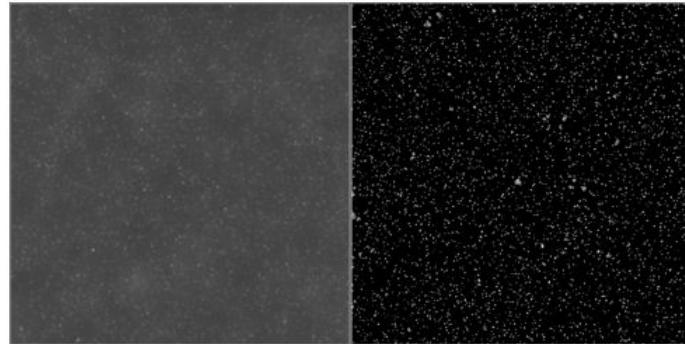


Figura 6.9: Resultado de la aplicación del filtro HighPass sobre una de las imágenes.

Los resultados en este caso fueron mejores respecto de los obtenidos mediante el filtro de pre-procesamiento CLAHE, obteniendo 4.977 para el valor del desplazamiento promedio en X y 4.979 para el valor del desplazamiento promedio en Y.

6.1.3. Procesamiento PIV con Pre y Post-procesamiento

Finalmente, se analizó el impacto del post-procesamiento sobre los resultados obtenidos anteriormente. Si bien el procesamiento PIV provisto por JPIV realiza un post-procesamiento entre cada una de las pasadas, la herramienta desarrollada

permite aplicar otras técnicas, como las provistas por MatPIV [19]. Como ya se ha mencionado anteriormente, los detalles de la incorporación de nuevos filtros se encuentran en el Capítulo 5.

Al igual que en los casos anteriores, para aplicar un filtro de post-procesamiento se debe elegir sobre la pestaña “Post-Procesamiento” los filtros deseados.

Los filtros aplicados para estas prueba fueron “Global Filter”, el cual elimina los vectores significativamente más grandes o pequeños que la mayoría de los vectores, y “NaN Interpolation”, que interpola los vectores eliminados a partir de los vectores vecinos.

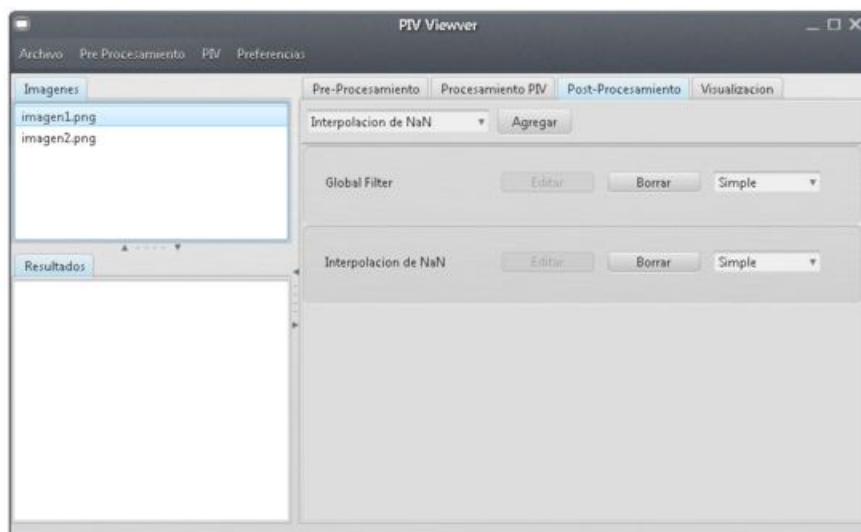


Figura 6.10: Selección y configuración de un filtro de Post-procesamiento.

Los resultados obtenidos aplicando previamente un pre-procesamiento CLAHE corresponden a un promedio de 4.952 píxeles en ambas coordenadas, lo cual representa una mejora respecto de los resultados arrojados previamente.

De la misma manera, se obtuvieron mejoras sobre los resultados cuando se aplica un pre-procesamiento HighPass, siendo el desplazamiento promedio en X e Y de 4.976 y 4.980 pixeles respectivamente.

6.1.4. Comparativa de los resultados

A continuación se presenta un gráfico con los resultados obtenidos en cada una de las pruebas.

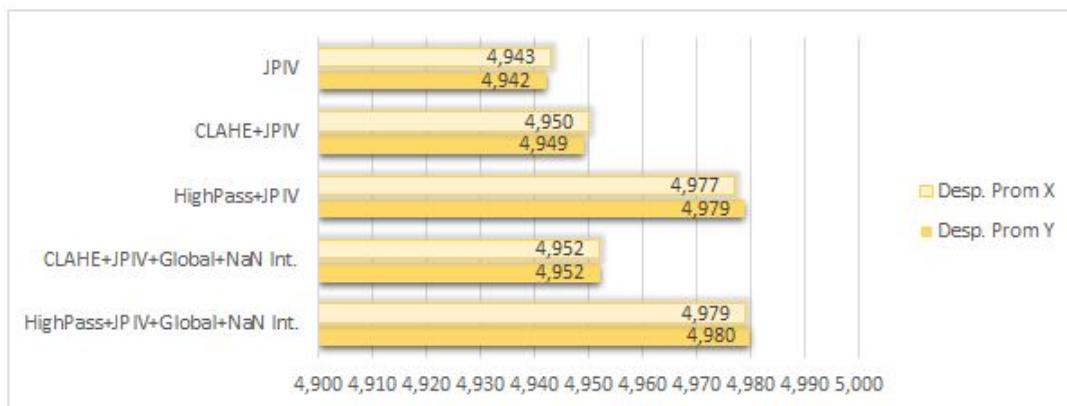


Figura 6.11: Gráfico comparativo de los resultados para el Caso de estudio 1.

Como se observa en el mismo, tanto los filtros de pre-procesamiento, en primera instancia, como los filtros de post-procesamiento, aplicados posteriormente, permiten mejorar los resultados obtenidos. Se observa que, para este caso en particular, el filtro de pre-procesamiento *HighPass* es aquel que presenta mejores resultados. Por otra parte, dadas las características del problema, el filtro de Media Global permite, en menor medida, obtener una mejora en la salida. Esto nos permite destacar, en este caso, que la calidad de los resultados depende principalmente del filtro de pre-procesamiento aplicado.

El siguiente cuadro presenta la comparativa de los resultados correspondientes a cada una de las pruebas, donde se observan los filtros aplicados en cada caso.

Este simple caso de estudio permitió verificar la mejora en los resultados a partir de la combinación de funcionalidades provistas por diferentes herramientas implementadas sobre la capa de abstracción propuesta.

Prueba	Filtros Aplicados	Herramientas que se extienden	Desp Promedio en X (pixel)	Desplazamiento Promedio en Y (pixel)
1	Correlación Cruzada	JPIV	4.943	4.942
2	CLAHE	ImageJ	4.950	4.949
	Correlación Cruzada	JPIV		
3	HighPass	ImageJ	4.977	4.979
	Correlación Cruzada	JPIV		
4	CLAHE	ImageJ	4.952	4.952
	Correlación Cruzada	JPIV		
	Global Filter Nan Interpolation	MathPIV		
5	HighPass	ImageJ	4.979	4.980
	Correlación Cruzada	JPIV		
	Global Filter Nan Interpolation	MathPIV		

Cuadro 6.1: Tabla comparativa de los resultados obtenidos en cada prueba.

6.2. Caso de estudio 2: Repositorio de imágenes de prueba

En este caso se utilizaron un conjunto de imágenes de prueba provistas por la herramienta JPIV [16]. Dicho caso se corresponde con la generación de imágenes sintéticas simulando el flujo de Poiseuille dentro de un canal rectangular. La salida es un conjunto de pares de imágenes correspondientes a diferentes planos dentro del mismo. Teniendo en cuenta esto, se optó por analizar dicha prueba desde el punto de vista del atributo de calidad “perfomance”, con el objetivo de mostrar la capacidad de la arquitectura propuesta (Capítulo 4) en lo que respecta a la ejecución de procesos en paralelo.

El caso de prueba constó de la ejecución de dos diferentes filtros sobre trece pares de imágenes. Se realizó una comparación en términos de tiempo de cálculo, entre la ejecución de los filtros en la herramienta desarrollada y la herramienta que provee la funcionalidad.

El primero de los filtros analizados fue el filtro de correlación cruzada provisto por JPIV. La herramienta JPIV utiliza una librería llamada JAI (Java Advanced Imaging), la cual paralleliza el procesamiento de las imágenes. Debido a esto, la concurrencia agregada por la herramienta propuesta no presentó grandes mejoras de *performance*. En la Figura 6.12 se presenta una comparativa entre ambas herramientas, donde se observa una mejora del 2.35 % producida por la capa de abstracción sobre las secciones de código no parallelizadas por la herramienta JPIV.

En segunda instancia, se analizó el filtro de pre-procesamiento CLAHE provisto por ImageJ. En este caso, se pudo mejorar la *performance* de la librería original mediante la ejecución concurrentemente incorporada por la capa. La utilización de la capa de abstracción posibilitó reducir el tiempo de ejecución del filtro aplicado sobre 26 imágenes, de un tiempo promedio de 41226.6ms a 13581.4ms, obteniendo una aceleración de 3.04 veces el tiempo original.

Por lo expuesto anteriormente, se pudo verificar la mejora que presenta la ejecución concurrente cuando se trabaja con un conjunto de imágenes consecutivas sobre las cuales se pretende analizar el comportamiento en tiempo y espacio de las partículas. En este sentido, la capa de abstracción desarrollada permite añadir concurrencia

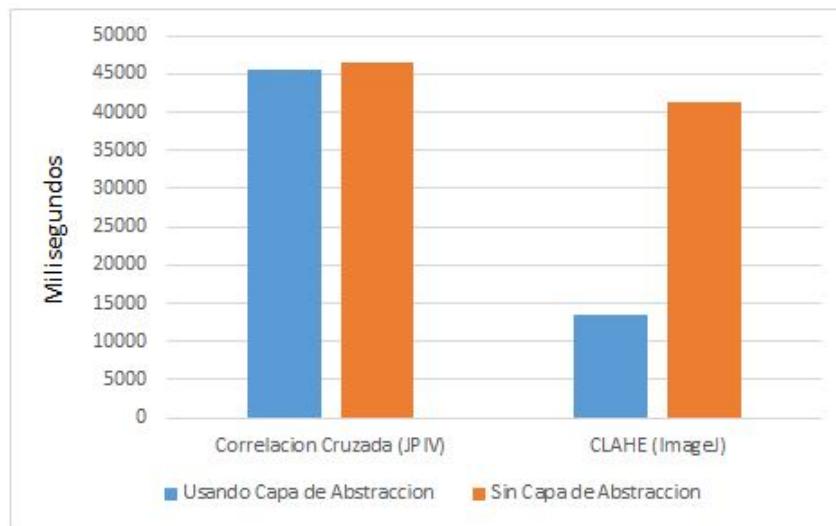


Figura 6.12: Comparativa de performance entre las funcionalidades ejecutadas sobre la capa de abstracción y sobre la herramienta original.

en aquellas herramientas extendidas que no la implementan, brindando en estos casos, una mejora significativa en términos de tiempo de cálculo. Por otra parte, si la herramienta extendida ya presenta un procesamiento concurrente, la capa de abstracción no degrada la performance significativamente, o presenta pequeñas mejoras mediante la paralelización de las secciones de código no paralelizadas.

6.3. Caso de estudio 3: Problema de la cavidad cúbica

Este caso de estudio considera el problema de la cavidad cúbica a partir del trabajo [9]. En el mismo, se presenta un estudio analítico, numérico y experimental del flujo de laminar en una cavidad cúbica. El fluido es un aceite dieléctrico utilizado para la refrigeración de los transformadores de distribución y potencia. Como la mayoría de los líquidos, este aceite exhibe viscosidad dependiente de la temperatura.

La cavidad cúbica de interés tiene una diferencia de temperatura impuesta entre dos paredes verticales opuestas, mientras que las otras paredes están aisladas y poseen idénticas características. Como resultado de la configuración, a medida que pasa el

tiempo, el flujo tiende a ser convectivo.

En la Figura 6.13 se puede ver la configuración del experimento, donde la cavidad está representada por un cubo de tamaño 0,1m x 0,1m x 0,1m. En dicho experimento se utilizó una cámara digital capaz de tomar 5 imágenes por segundo (fps), un láser en modo continuo, y una lente cilíndrica para expandir el láser con el fin de generar una lámina de luz que ilumina el plano de medición. La prueba consistió en captar cinco diferentes planos situados en 0.01m, 0.02m, 0.03m, 0.04m y 0.05m.

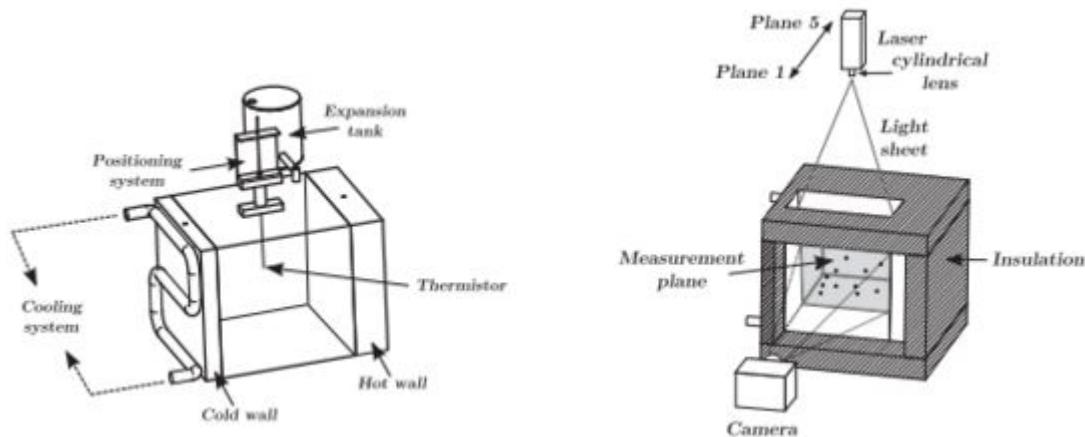


Figura 6.13: Configuración de la cavidad cúbica.

El artículo plantea cuatro casos de estudio, donde se analiza el comportamiento de la cavidad de acuerdo a la aplicación de diferentes temperaturas en las paredes fría y caliente. Los autores proponen el estudio de los perfiles de velocidad horizontal y vertical, ejes Z y X de la cavidad. Los resultados se comparan con el cálculo analítico, respecto de los obtenidos de forma experimental a través de una herramienta de procesamiento PIV. En el presente trabajo se analizó sólo el perfil de velocidad horizontal.

Luego de establecer un contacto con parte de los autores del artículo referenciado, fue posible obtener un conjunto de imágenes correspondientes al primer plano, con valores de temperatura de 50 °C y 30 °C para las paredes caliente y fría respectivamente. El tiempo entre la captura de las imágenes utilizadas fue de 0.2s, y dada la resolución de las mismas, se determinó la región de interés correspondiente a la cavidad cúbica para los puntos P1=(222px, 65px) y P2=(1419px, 1251px). Debido a que

la distancia entre las paredes de la cavidad era de 0.1m, se pudo estimar el tamaño representado por cada píxel, siendo el mismo aproximadamente 0.0000084m. Estos datos son muy importantes para determinar la velocidad a partir del desplazamiento en píxeles obtenido por el procesamiento PIV.

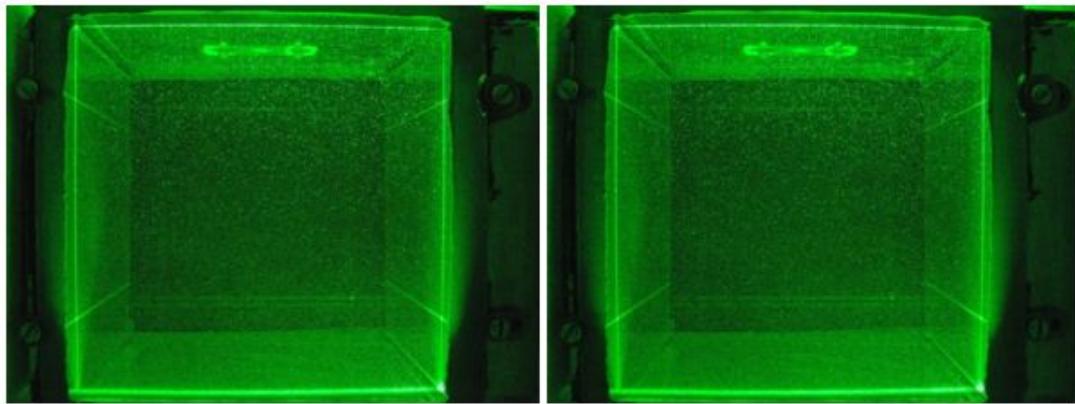


Figura 6.14: Imágenes experimentales de la cavidad cúbica.

A partir de las imágenes experimentales (Figura 6.14) se procedió a realizar el estudio del perfil de velocidad horizontal con la herramienta PIV propuesta. Se realizó una comparación de los resultados a partir de la aplicación de diversos filtros de pre y post-procesamiento incluidos en la capa de abstracción desarrollada, y extendidos de diferentes herramientas externas. Como se verá más adelante, esto permitió nuevamente verificar las ventajas de la herramienta implementada al combinar funcionalidades de otras aplicaciones.

En primer instancia se estudió el perfil de velocidad utilizando la herramienta propuesta y aplicando únicamente un filtro de procesamiento PIV. Si bien se contaba con dos filtros para esta funcionalidad extendidos desde las herramientas JPIV y MatPIV, se optó por utilizar el de JPIV ya que los resultados numéricos fueron más aproximados a los obtenidos (Figura 6.15). Posteriormente, se analizó el mismo caso aplicando adicionalmente un filtro de pre-procesamiento CLAHE (Figura 6.16), el cual utiliza la librería de ImageJ; y por último se buscó una mejora de los resultados mediante la utilización del filtro de post-procesamiento LocalFilter extendido desde la herramienta MatPIV, aunque en este caso la aplicación del filtro no permitió mejorar los resultados.

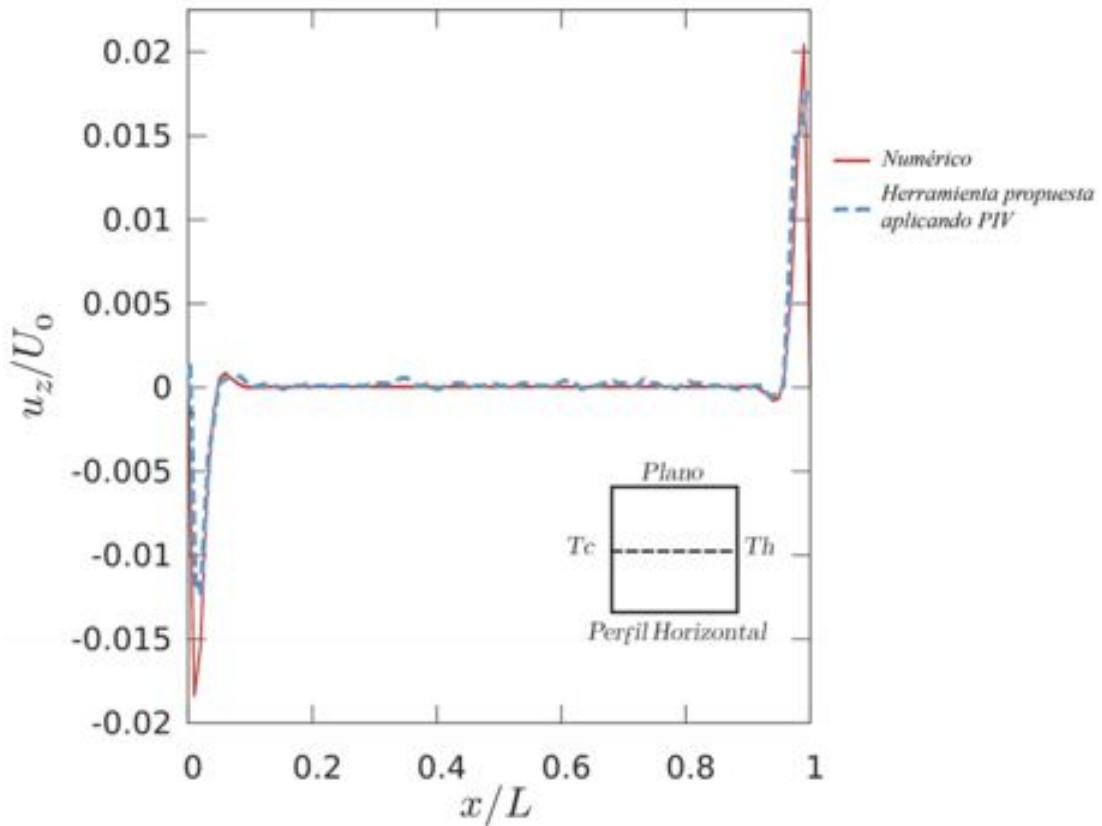


Figura 6.15: Comparación entre el perfil de velocidad horizontal obtenido numéricamente y el resultado obtenido con la herramienta propuesta a partir de un filtro el procesamiento PIV.

En las Figuras 6.15 y 6.16 se puede ver que se obtienen resultados similares a los presentados en el artículo mediante cálculo numérico. Sin embargo, a fin de realizar un análisis más exhaustivo, se muestra un gráfico comparativo con los valores del perfil de velocidad horizontal próximos a las paredes fría y caliente, las cuales presentan el mínimo y máximo de la función respectivamente.

En el Cuadro 6.2, se muestra una tabla comparativa de los resultados obtenidos por la herramienta propuesta en cada una de las pruebas. Además, se exponen los filtros de pre-procesamiento, procesamiento y post-procesamiento aplicados en cada una de ellas y la herramienta externa de la cual se extiende dicha funcionalidad. Además se presentan los datos numéricos y experimentales presentes en el artículo referenciado a fin de validar los resultados obtenidos.

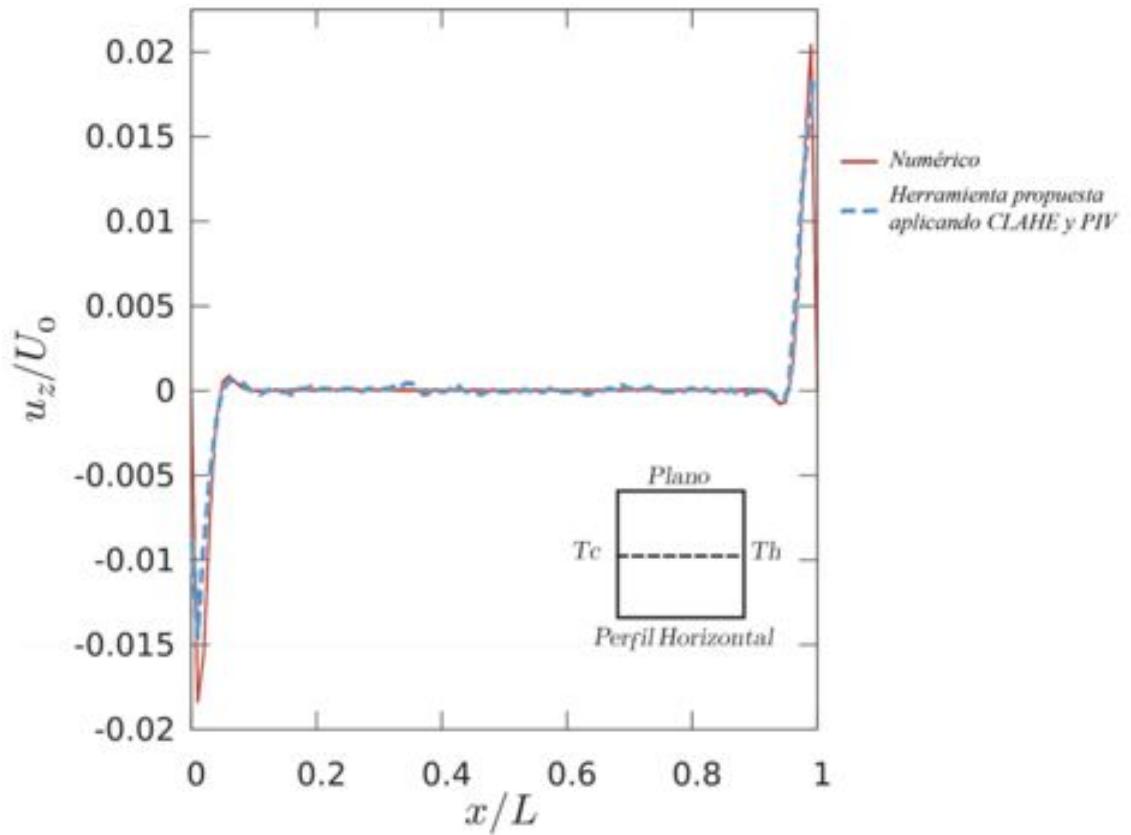


Figura 6.16: Comparación entre el perfil de velocidad horizontal obtenido numéricamente y el resultado obtenido con la herramienta propuesta a partir de un filtro de pre-procesamiento CLAHE y un filtro de procesamiento PIV.

Como se observa en el Cuadro 6.2 y en la Figura 6.17, el pre-procesamiento CLAHE provisto por ImageJ permitió mejorar los resultados que se habían obtenido en primera instancia, utilizando solo el filtro de correlación cruzada provisto por JPIV. Esto confirma nuevamente que pre-procesar las imágenes a fin de aumentar su calidad impacta directamente sobre la mejora de los resultados. Por otra parte, el filtro de post-procesamiento provisto por la herramienta MatPIV no permitió mejorar los resultados obtenidos previamente; sin embargo, dicha herramienta posibilitó añadir de modo sencillo un nuevo filtro de visualización para generar streamlines del caso de estudio (Figura 6.18).

Por último, se desarrollaron filtros adicionales que permitieron obtener y analizar los resultados del caso de estudio de manera más sencilla. Primeramente, un filtro de

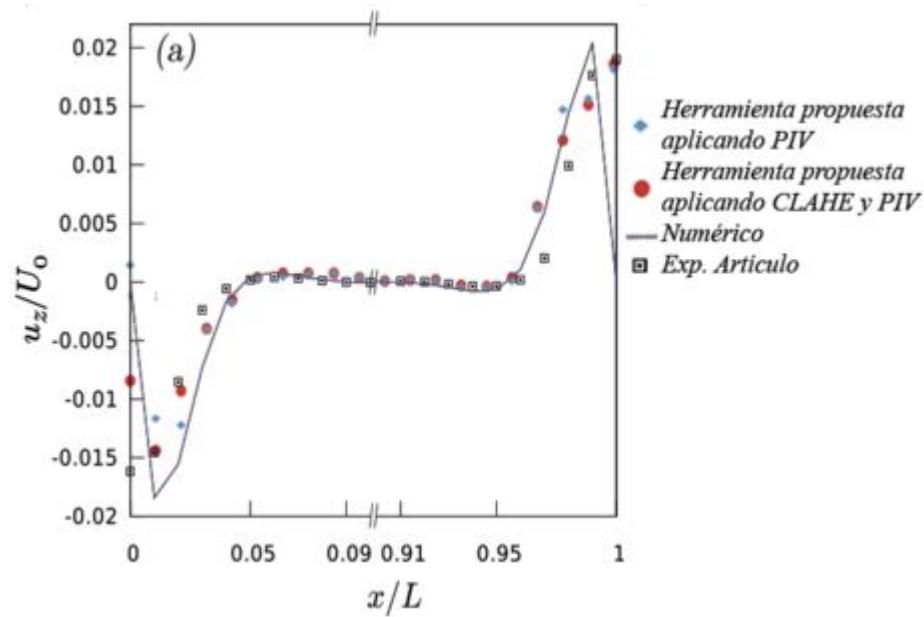


Figura 6.17: Comparación entre el perfil de velocidad horizontal obtenido numéricamente y el resultado obtenido con la herramienta propuesta a partir de un filtro de pre-procesamiento CLAHE y un filtro de procesamiento PIV.

Prueba	Filtros Aplicados	Herramientas que se extienden	Desp Mínimo (pared fria)	Desp Máximo (pared caliente)
1	Correlación Cruzada	JPIV	-0,0123	0,0181
2	CLAHE	ImageJ	-0,0146	0,0186
	Correlación Cruzada	JPIV		
3	CLAHE	ImageJ	-0,0146	0,0186
	Correlación Cruzada	JPIV		
	Global Filter Nan Interpolation	MathPIV		
	Numérico		-0,0184	0,0204
	Experimental según el artículo referenciado		-0,0147	0,0198

Cuadro 6.2: Cuadro comparativo de los resultados para el Caso de estudio 3.

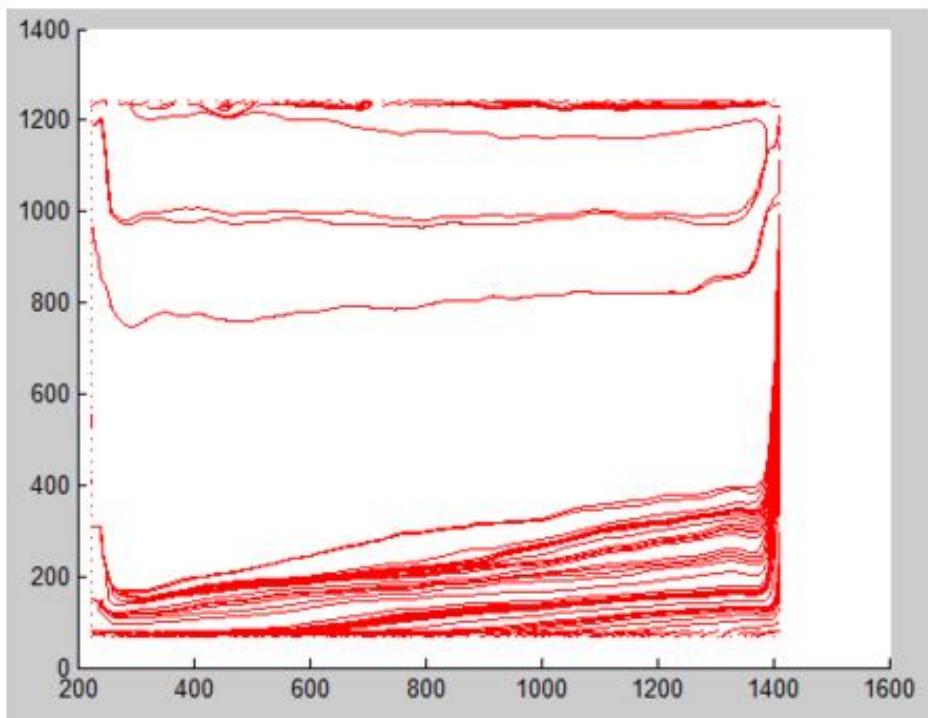


Figura 6.18: Streamlines correspondientes al caso de estudio 3.

post-procesamiento permitió calcular y almacenar en el mapa de vectores las velocidades correspondientes utilizando los parámetros: tiempo entre imágenes y tamaño del pixel. Esto fue necesario ya que herramientas como JPIV solo trabajan con el desplazamiento en píxeles sin calcular las velocidades de cada vector. En segunda instancia, se implementó un filtro de visualización que permitió almacenar en una planilla de cálculo el perfil de velocidad deseado, lo cual se utilizó en el análisis de los resultados obtenidos tras cada prueba. Esto marca nuevamente la flexibilidad y extensibilidad de la herramienta desarrollada para incorporar nuevas funcionalidades adaptadas a las necesidades del usuario.

6.4. Caso de estudio 4: Cálculo PIV para un canal poiseuille

En este último caso se realizó el estudio experimental de un flujo laminar de Poiseuille donde las partículas siguen trayectorias definidas (todas las partículas que pasan por un punto en el campo del flujo siguen la misma trayectoria). En este tipo de flujos, la ley de Poiseuille considera que el mismo tiene un perfil de velocidades parabólico, donde la velocidad máxima se tiene en el centro del plano 2D [31] (Figura 6.19)

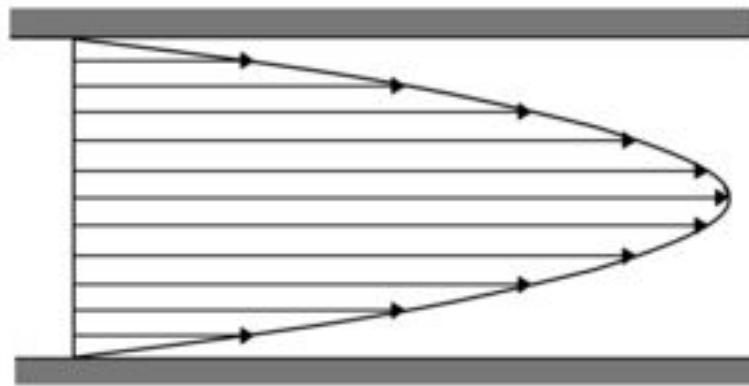


Figura 6.19: Perfil de velocidad parabólico de un flujo poiseuille.

6.4.1. Configuración experimental

El dispositivo experimental es un canal rectangular de 96cm x 11cm x 9.7cm cuya parte superior se encuentra descubierta, y fue construido a partir de paredes de vidrio de 5mm de espesor. Dentro de dicho canal se colocó el fluido de modo tal de alcanzar un nivel de 5 cm de altura. Para generar un flujo continuo se utilizó un motor bomba de desagote de lavarropas con un caudal de 22 litros/minuto (Figura 6.20). Las mangueras de entrada y salida de la bomba se sujetaron en los extremos del canal tal como se puede ver en la Figura 6.21, tratando de generar continuidad del flujo y evitando perturbaciones que introduzcan errores en el cálculo. Sobre la manguera de entrada se colocó una llave limitadora de caudal a fin de regular la

velocidad del flujo. Con el objetivo de lograr un flujo laminar, se dispuso de un filtro construido a partir de tubos plásticos de 6mm de diámetro (Figura 6.22) colocado a continuación de la manguera que generaba la entrada del fluido al canal. En la Figura 6.23 se puede observar una imagen de la configuración final del canal.

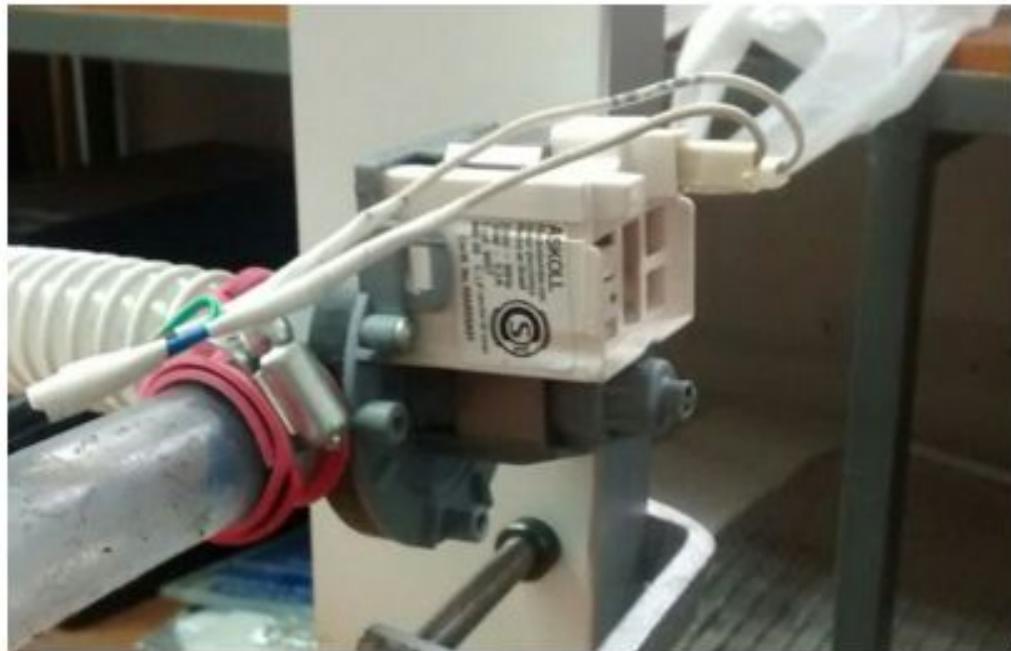


Figura 6.20: Motor de bomba de desagote utilizado para el experimento.



Figura 6.21: Mangueras de entrada y salida del fluido conectadas al canal.

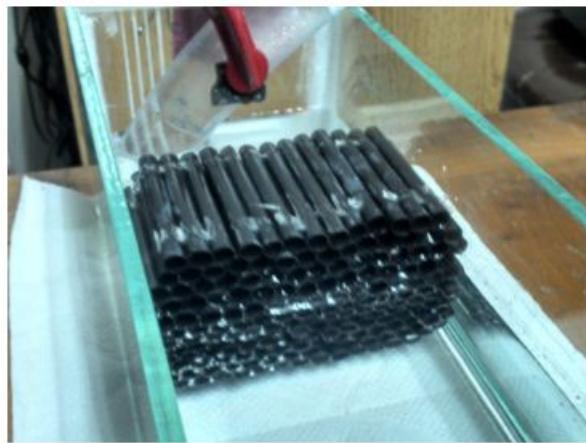


Figura 6.22: Filtro utilizado para laminarizar el flujo.



Figura 6.23: Configuración final del canal.

Para obtener de las imágenes necesarias para el procesamiento PIV se utilizó una cámara digital modelo Lumix DMC-GK5 de 16,05 megapíxeles con grabación de vídeo Full HD 1920px X 1080px, capaz de tomar imágenes con velocidad de muestreo de 60 frames por segundo. Para lograr mantener la cámara en una posición fija evitando desplazamientos de la imagen, se utilizó un soporte por sobre el canal a una distancia de 26 cm, lo cual resultó adecuado para captar las partículas trazadoras posteriormente insertadas en el fluido (Figura 6.24).

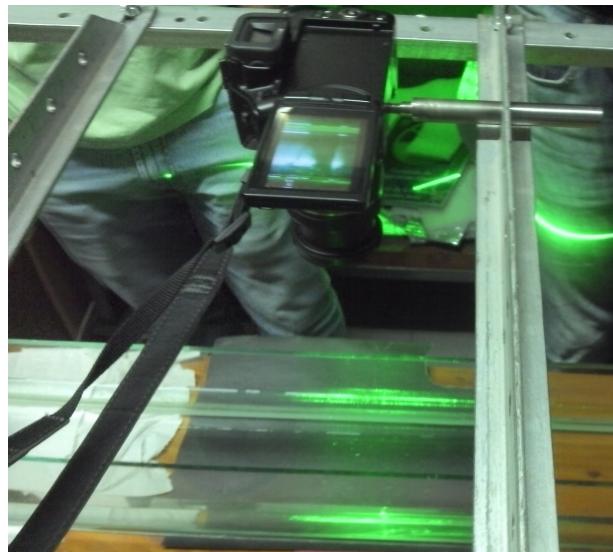


Figura 6.24: Cámara utilizada para la obtención de las imágenes.

Mediante un láser y una lente cilíndrica colocados de forma perpendicular a las paredes de mayor longitud del canal, fue posible generar el plano horizontal de luz láser que iluminó el área de medición. Dicho láser y lente se ubicaron de tal modo que el plano quede situado a una altura media respecto de la altura máxima del fluido dentro del canal, aproximadamente 2,5 cm. (Figura 6.25).

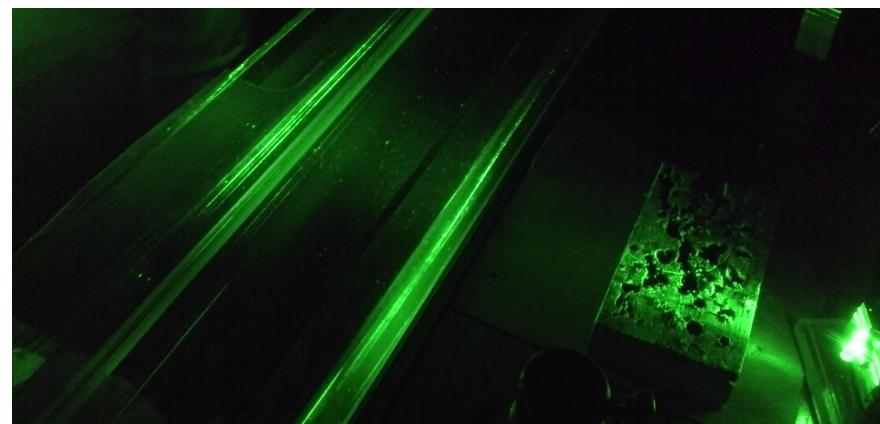


Figura 6.25: Láser generando el plano de luz sobre el canal.

Finalmente, se utilizaron microesferas de vidrio de un diámetro de $10\mu\text{m}$ con una densidad de 1.1 g/cm^3 como partículas trazadoras de flujo (Figura 6.26). Al ser

iluminadas dichas partículas, la luz dispersada es grabada por la cámara digital de enfoque perpendicular al plano láser.



Figura 6.26: Microesferas de vidrio utilizadas como partículas trazadoras.

Una vez configurado el experimento, se dió inicio a la circulación del fluido dentro del canal. Se realizó una grabación de aproximadamente 6 minutos con el objetivo de analizar la evolución del flujo del fluido a partir de imágenes tomadas en diferentes instantes de tiempo. En la Figura 6.27 se puede ver un ejemplo de las imágenes obtenidas a partir de la grabación.

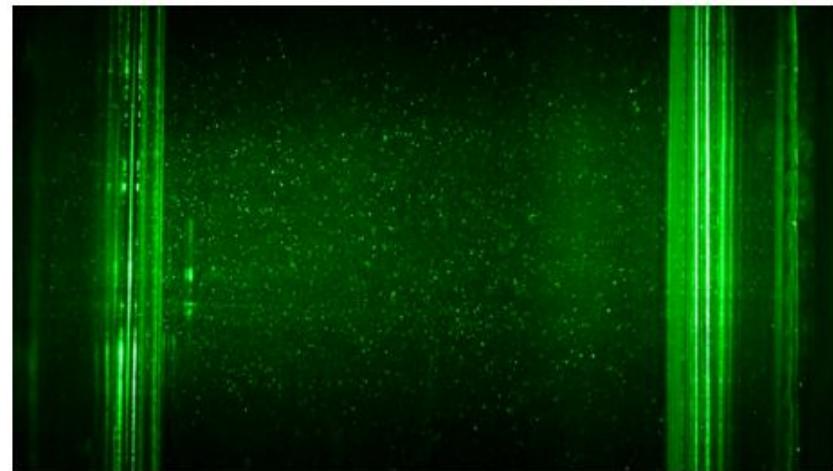


Figura 6.27: Imagen del plano de luz obtenida por la cámara de enfoque perpendicular al plano.

6.4.2. Resultados experimentales

Luego de realizadas las grabaciones del experimento se analizó el flujo del fluido en diferentes instantes de tiempo con el objetivo observar el comportamiento del mismo. La teoría de los flujos laminares de Poiseuille (LPF) [6] expone que durante los pasos iniciales el fluido se encuentra en fase de desarrollo, mientras que a medida que avanza el tiempo el mismo comienza a estabilizarse.

Teniendo en cuenta que la velocidad del fluido no era lo suficientemente alta como para seleccionar pares consecutivos de frames del video, y poder apreciar correctamente el desplazamiento de una partícula, se optó por seleccionar una imagen cada 4 frames de video. Dado que la cámara utilizada logra obtener 60 frames por segundo, el tiempo entre imágenes es de $1/60 * 4 = 0.066$ segundos. Por otra parte la región rectangular de interés a ser evaluada en la imagen se delimitó por los puntos $P1 = (260\text{px}, 320\text{px})$ y $P2 = (1600\text{px}, 960\text{px})$ determinando un tamaño de píxel de 0.000081m .

Para el análisis de este caso de estudio, se utilizó la herramienta propuesta aplicando un filtro de pre-procesamiento CLAHE a fin de mejorar la calidad de las imágenes, y el filtro PIV de correlación cruzada provisto por JPIV. En este sentido fue necesario trabajar con un tamaño de ventana de correlación mínimo de 32×32 (mayor a los anteriores casos) dada la relación entre la densidad de partículas y la resolución de las imágenes.

En la Figura 6.28 se puede apreciar los resultados para distintos tiempos compatibles con la teoría LPF. Durante los primeros instantes de tiempo se observa un flujo LPF en desarrollo con un perfil de velocidad completamente plano. Conforme avanza el tiempo es posible apreciar cómo el flujo LPF comienza a desarrollarse tendiendo a un perfil de velocidad parabólico.

Este caso de estudio permitió realizar la experiencia de aplicar la técnica PIV basada en un caso real, con todos los aspectos y problemáticas de configuración que la misma requiere. Además posibilitó contrastar los resultados obtenidos por la herramienta desarrollada a partir de imágenes propias, con el objetivo de establecer un análisis posterior de dichos resultados.

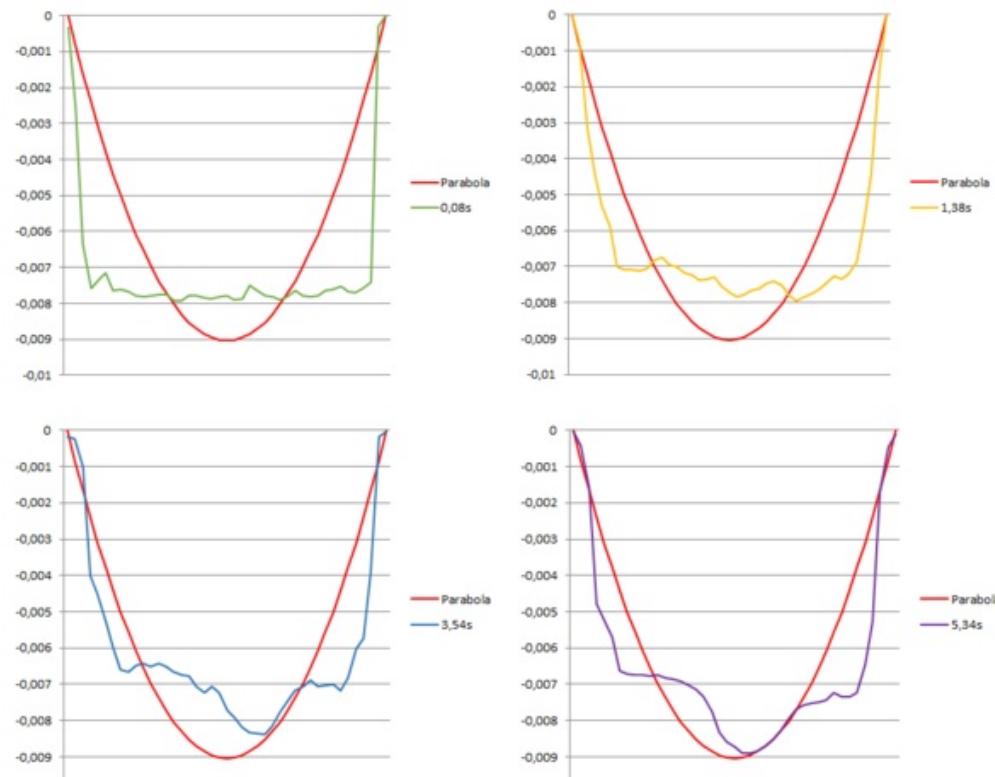


Figura 6.28: Perfiles de velocidad correspondientes a los diferentes instantes de tiempo.

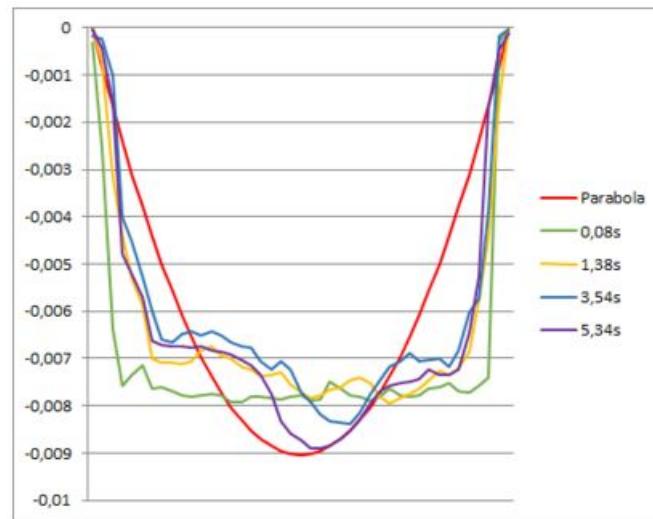


Figura 6.29: Comparación de los perfiles de velocidad.

6.4.3. Inconvenientes presentados

El primer inconveniente que se presentó fue la necesidad de disminuir la introducción de ruido en el canal. Para esto, previamente al armado del mismo, fue necesario limpiarlo utilizando alcohol isopropílico con el objetivo de eliminar todas las posibles imperfecciones que pudieran inferir en los resultados obtenidos, y adicionalmente purgar la bomba de cualquier partícula indeseada que pudiese contener.

En segunda instancia, fue necesario acondicionar el lugar para evitar el ingreso de la luz exterior. El objetivo de esto, fue permitir a la cámara captar la luz emitida por el láser, obteniendo una mayor distinción de las partículas trazadoras en el fluido.

Por último, luego de probar con diferentes cámaras la toma de imágenes y videos, se pudo verificar que la calidad del dispositivo de captura repercute fuertemente en la apreciación de las partículas en movimiento. Esto llevó a utilizar una cámara de alta calidad. Entre las características que debe poseer la misma se pueden resaltar:

- Es importante que el zoom sea óptico para no perder calidad.
- El tamaño de la imagen (resolución) debe ser suficiente para captar las partículas con claridad.
- El foco debe ser manual, ya que el mismo debe estar sobre el plano láser, y un foco automático podría cambiar el foco al fondo u movimientos circundantes.
- El tiempo de exposición del obturador debe ser lo suficientemente chico como para que el movimiento de las partículas no cause una imagen movida, pero no tan chico como para no captar la luz de la mayor parte de las partículas.

Conclusiones

En el presente trabajo se implementó una solución basada una capa de abstracción, la cual posibilita abstraer distintas herramientas de procesamiento PIV basadas en software libre. Además, no solo permite incorporar las funcionalidades de las herramientas abstraídas, sino que también, implementar nuevas, combinarlas y ejecutarlas en una secuencia de etapas predefinidas (pre-procesamiento, procesamiento PIV, post-procesamiento y visualización). Dicha combinación de funcionalidades posibilitó la ejecución de filtros sobre el conjunto de datos de entrada y salida de cada una de las etapas involucradas, logrando en la mayoría de los casos analizados una mejora en los resultados obtenidos respecto de los calculados individualmente por las herramientas. Además, se logró realizar un análisis más exhaustivo de los casos mencionados mediante la utilización de diferentes filtros de visualización.

En la etapa de investigación de las herramientas PIV más utilizadas, se pudo verificar que las mismas no poseían tratamiento de pre-procesamiento sobre el conjunto imágenes de entrada. Por eso vale la pena remarcar la mejora aportada por la capa de abstracción desarrollada, permitiendo incluir nuevas funcionalidades, e incluso de herramientas cuyo dominio no está orientado a la velocimetría de imágenes de partículas. La división del procesamiento en etapas permitió independizar el tratamiento de cada uno de los filtros, agregar múltiples formas de visualización de los resultados, y mejorar los resultados numéricos, reduciendo, por ejemplo, el ruido existente en las imágenes.

En lo que respecta al diseño de la herramienta, se priorizó la agregación futura de nuevas funcionalidades. Por ese motivo, se implementó una arquitectura en donde los filtros a utilizar pueden ser incorporados sin la necesidad de modificar el código del núcleo de la capa de abstracción. En caso de requerirlo sólo será necesario agregarlas

de forma externa, posibilitando al usuario a la utilización de nuevos filtros adaptados a las necesidades de la prueba que se esté analizando. Este mecanismo de incorporación de nuevas funcionalidades provisto por el software desarrollado, posee la ventaja de no tener la necesidad de recompilar la aplicación para que la capa de abstracción sea capaz de detectar las nuevas funcionalidades. Solo es necesario empaquetar los archivos de código que definen los nuevos filtros en un archivo “.jar” y copiarlos en el directorio correspondiente para que la herramienta pueda cargarlos una vez que la misma se inicia. Esto brinda mayor extensibilidad, posibilitando la agregación de nuevos filtros de forma dinámica y desligando a los usuarios de la necesidad de manipular el código interno de la capa. No obstante esto, pueden presentarse casos excepcionales donde la forma de representación de las estructuras de datos pueda generar dificultades a la hora de intercambiar las mismas entre las diferentes herramientas extendidas. De todas formas, la capa de abstracción se diseñó de forma que sea simple de modificar acorde al nuevo requerimiento, para luego recompilar la misma en estos casos excepcionales.

Teniendo en cuenta que algunas de las herramientas analizadas al momento de realizar el relevamiento no poseían ejecución concurrente de procesos, otra característica que se tuvo presente a la hora de pensar la arquitectura, fue brindar la posibilidad de realizar procesamiento multi hilo. El objetivo es aumentar la performance del procesamiento completo. Si bien, como se analizó durante los resultados presentados en el Capítulo 6, algunas de las herramientas no presentaron mejoras significativas debido a que las mismas ya tenían un procesamiento concurrente, en otros casos se lograron mejoras de hasta 3 veces en términos de velocidad de cálculo.

Luego de analizar las herramientas relevadas se pudo verificar que las mismas realizan un recálculo de los resultados, aún cuando el procesamiento que se desea hacer ya se había realizado con anterioridad sobre el mismo conjunto de datos de entrada. Por este motivo, se planteó el uso de una caché que guarda los resultados generados para cada entrada por cada filtro, posibilitando al usuario realizar pruebas con mayor fluidez, evitando el reprocesamiento de filtros. Esta estructura juega un rol muy importante cuando se realizan procesamiento de imágenes de gran tamaño, en donde el tiempo de cálculo es significativo.

También, se consideraron casos de estudio donde es necesario contar con una he-

rramienta que permita el procesamiento de un conjunto de imágenes consecutivas, cuando se busca analizar el comportamiento de las partículas en tiempo y espacio. Ya que la mayoría de las herramientas relevadas no cuentan con esta funcionalidad, la capa de abstracción se desarrolló de modo que permita realizar este tipo de procesamiento independientemente de la funcionalidad que ejecute por debajo.

De esta forma, se desarrolló una herramienta de código abierto y multiplataforma, la cual ofrece grandes facilidades para ser extendida mediante la incorporación de nuevos filtros. Esto permite integrar de manera transparente, funcionalidades provistas por diversas herramientas que, trabajando cooperativamente, posibilitan la obtención de mejores resultados centrados en el caso que desea ser estudiado.

Por último, la herramienta fue probada y validada en diversos casos de estudio, comparando los resultados obtenidos contra resultados conocidos analítica y/o numéricamente. Esto permitió verificar la importancia y utilidad de contar con un diseño modificable que posibilite extender y aplicar filtros en cada una de las etapas, obteniendo una mejora sustancial de los resultados. En los casos de estudio analizados se consideraron desde imágenes sintéticas, obtenidas de repositorios para el análisis de PIV, hasta imágenes experimentales, capturadas a partir de casos reales (el caso experimental). Los experimentos realizados brindaron la posibilidad de comprender y profundizar en la amplia variedad de aspectos que se deben tener en cuenta al momento de generar las imágenes necesarias para realizar un procesamiento PIV.

Trabajos futuros

Si bien se incorporaron funcionalidades preexistentes de las herramientas de PIV referenciadas, existe la posibilidad de generar nuevos filtros, mejorar los existentes, y agregar nuevas funcionalidades. Ejemplos de esto pueden ser el análisis estadístico de los datos, nuevas formas de visualización de mapas de vectores y simulaciones del movimiento de un flujo a partir de un conjunto de pares de imágenes.

Un factor a tener en cuenta es el intervalo de tiempo en que son tomadas las imágenes respecto de la velocidad a la que se mueve el fluido. Es decir, que un intervalo largo con un fluido que se mueve a alta velocidad podría producir que las partículas se escapen del área capturada por las imágenes, mientras que un intervalo

pequeño con un fluido que se mueve lentamente podría dar lugar a que las partículas no presenten un desplazamiento significativo. Por este motivo, puede resultar interesante agregar en un futuro la detección inteligente del intervalo, obteniendo como entrada un conjunto de imágenes omitiendo aquellas imágenes que representarían un desplazamiento muy pequeño.

En cuanto a la posibilidad de ejecutar PIV en 3D, actualmente la capa de abstracción solo se encuentra preparada para el procesamiento en 2D. Sin embargo, es posible inferir que no traería demasiadas dificultades poder adaptarla. Básicamente habría que extender la representación de las estructuras de datos, ya sea utilizando matrices de velocidades en 3D, imágenes holográficas, imágenes estereoscópicas, entradas basadas en múltiples planos u otros elementos procesables que sean requeridos según la técnica utilizada.

Otra posible extensión es modificar la capa de abstracción para acceder al hardware de la cámara y al láser (usando controladores). Esto permitiría integrar la herramienta con el hardware de captura para controlar todos los componentes del experimento. De esta forma, sería posible obtener un flujo continuo de pares de imágenes tomadas (procesadas en tiempo real), logrando un sistema completo de PIV.

Por último, informalmente se ha propuesto a los grupos con los que se estableció contacto, ya sea para obtener imágenes o datos experimentales, que utilicen la herramienta desarrollada. Para ello, la misma se dejará disponible en un sitio web conjuntamente con un manual de uso y datos de contacto.

Bibliografía

- [1] Ronald J Adrian. Scattering particle characteristics and their effect on pulsed laser measurements of fluid flow: speckle velocimetry vs particle image velocimetry. *Applied optics*, 23(11):1690–1691, 1984.
- [2] Ronald J Adrian. Particle-imaging techniques for experimental fluid mechanics. *Annual review of fluid mechanics*, 23(1):261–304, 1991.
- [3] Ronald J Adrian. Statistical properties of particle image velocimetry measurements in turbulent flow. *SPIE MILESTONE SERIES MS*, 99:191–191, 1994.
- [4] Ronald J Adrian. Twenty years of particle image velocimetry. *Experiments in fluids*, 39(2):159–169, 2005.
- [5] Dwayne A Bourgoine, Carolyn Q Judge, Joshua M Hamel, Steven L Ceccio, y David R Dowling. Lifting surface flow, pressure, and vibration at high reynolds-number. 2001.
- [6] Florent Brunet, Emmanuel Cid, Adrien Bartoli, et al. Simultaneous image registration and monocular volumetric reconstruction of a fluid flow. págs. 1–12, 2011.
- [7] James Carey, Brent Carlson, y Tim Graser. *SanFrancisco design patterns: blueprints for business software*. Addison-Wesley Professional, 2000.
- [8] Paul C Clements. *Software architecture in practice*. 2002.
- [9] Paola A Córdoba, Nicolás Silin, y Enzo A Dari. Natural convection in a cubical cavity filled with a fluid showing temperature-dependent viscosity. *International Journal of Thermal Sciences*, 98:255–265, 2015.

- [10] Henry de Pitot. Description d'une machine pour mesurer la vitesse des eaux courantes et le sillage des vaisseaux. 1966.
- [11] Erich Gamma, Richard Helm, Ralph Johnson, y John Vlissides. *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.
- [12] Qi Gao, HongPing Wang, y GongXin Shen. Review on development of volumetric particle image velocimetry. *Chinese Science Bulletin*, 58(36):4541–4556, 2013.
- [13] Hasan Gunes y Ulrich Rist. Spatial resolution enhancement/smoothing of stereo-particle-image-velocimetry data using proper-orthogonal-decomposition-based and kriging interpolation methods. *Physics of Fluids (1994-present)*, 19(6):064101, 2007.
- [14] Klaus D Hinsch y Heiko Hinrichs. Three-dimensional particle velocimetry. págs. 129–152, 1996.
- [15] ImageJ. <http://imagej.nih.gov/ij/index.html>.
- [16] JPIV. www.jpiv.vennemann-online.de/.
- [17] Richard D Keane y Ronald J Adrian. Theory of cross-correlation analysis of piv images. *Applied scientific research*, 49(3):191–215, 1992.
- [18] Lingli Liu, Hairong Zheng, Logan Williams, y Robin Shandas. Effect of contrast microbubble concentration on quality of echo particle image velocimetry (echo piv) data: Initial in vitro studies. *Ultrasonics Symposium*, págs. 1560–1563, 2003.
- [19] MatPIV. www.mn.uio.no/math/english/people/aca/jks/matpiv/.
- [20] MPIV. www.oceanwave.jp/softwares/mpiv/index.php.
- [21] Christopher JD Pickering y Neil A Halliwell. Speckle photography in fluid flows-signal recovery with two-step processing. *Applied optics*, 23(8):1128–1129, 1984.
- [22] PIVAnalizer. fiji.sc/wiki/index.php/PIV_analyser.

- [23] PIVPlugin. sites.google.com/site/qingzongtseng/piv.
- [24] Ajay K Prasad. Particle image velocimetry, review article. *Science*, 79, No.1:51–60, 2000.
- [25] Thomas Roberts Puzak. Analysis of cache replacement-algorithms. 1985.
- [26] Markus Raffel, Christian E Willert, Jürgen Kompenhans, et al. *Particle image velocimetry: a practical guide*. Springer, 2013.
- [27] Fulvio Scarano y Michel L Riethmuller. Iterative multigrid approach in piv image processing with discrete window offset. *Experiments in Fluids*, 26(6):513–523, 1999.
- [28] Abraham Silberschatz, Peter B Galvin, Greg Gagne, y A Silberschatz. *Operating system concepts*, tomo 4. Addison-Wesley Reading, 1998.
- [29] William Thielicke y Eize J Stadhuis. Pivlab—towards user-friendly, affordable and accurate digital particle image velocimetry in matlab. *Journal of Open Research Software*, 2(1):e30, 2014.
- [30] Jerry Westerweel. *Digital particle image velocimetry: theory and application*. TU Delft, Delft University of Technology, 1993.
- [31] Frank White. *Fluid Mechanics*. McGraw-Hill Series, 2010.
- [32] Christian E Willert y Morteza Gharib. Digital particle image velocimetry. *Experiments in fluids*, 10(4):181–193, 1991.
- [33] Zhenyu Xue, John J Charonko, y Pavlos P Vlachos. Particle image velocimetry correlation signal-to-noise ratio metrics and measurement uncertainty quantification. *Measurement Science and Technology*, 25(11):115301, 2014.