Universidad Zaragoza
1542

Proyecto Fin de Carrera

Estimación online del flujo óptico a partir de una secuencia de vídeo

Autor/es

Luis Llamas Binaburo

Director/es y/o ponente

Pedro Antonio Piniés Rodríguez

Escuela de Ingeniería y Arquitectura
2013

Repositorio de la Universidad de Zaragoza – Zaguan http://zaguan.unizar.es

Estimación online del flujo óptico a partir de una secuencia de vídeo

Resumen

El objeto del proyecto es la implementación en C++/CUDA de un algoritmo para la estimación del movimiento de los pixels que componen una imagen (Optical Flow). Asimismo se desea acelerar el proceso hasta conseguir su aplicación en tiempo real a partir de imágenes capturadas mediante una webcam. A través de este cálculo se obtiene una gran información de los objetos dinámicos en la escena y supone un paso previo para el desarrollo de una amplia variedad de aplicaciones tanto en la robótica móvil como en visión por computador.

Los fundamentos matemáticos en los que se basa el algoritmo son técnicas de variación total aplicadas al ámbito de la visión. Las funciones empleadas son no diferenciables, por lo que se emplea el método de optimización convexa basado en la transformada de Fenchel-Legendre. Para su resolución se emplea un algoritmo de Proximal Map. Debido a la complejidad matemática de estas técnicas se ha implementado, de forma adicional, una aplicación de filtrado de ruido en tiempo real (Denoise) que ha permitido familizarse con las técnicas empleadas con carácter previo a la implementación del Optical Flow.

Para la aceleración del cálculo hasta conseguir la ejecución en tiempo real de ambas aplicaciones se ha hecho uso de la tecnología CUDA que permite la ejecución de programas en la tarjeta de vídeo (GPU), lo que supone un paradigma de computación paralela SIMT (Single Instruction Multiple Thread), es decir, una misma instrucción ejecutada en múltiples threads.

Los resultados obtenidos recogen, para ambas aplicaciones, la comparación de tiempos de ejecución entre CPU y GPU, la influencia en el mismo de los distintos parámetros que intervienen, la caracterización de la eficacia en la consecución del resultado deseado y, finalmente, la obtención de una zona de real time como combinación de los parámetros de entrada dentro de la cual se consigue la ejecución en tiempo real.

A mi tío y a mi abuela.
Allá donde estéis, espero que os sintáis orgullosos de mi.

Índice

Memoria

1 Introducción 2
1.1 Objetivos 2
1.2 Definición de Optical Flow 2
1.3 Definición de CUDA 3
1.4 Trabajos realizados 5
1.5 Consideraciones adicionales 6
1.5.1 Equipo empleado 6
1.5.2 Igualdad de condiciones 6
1.5.3 Parámetros de simulación 6
2 Fundamentos matemáticos 7
3 Denoise 13
3.1 Planteamiento 13
3.2 Clase Denoise 14
3.3 Resultados 15
3.3.1 Comparación entre CPU y GPU 15
3.3.2 Influencia de los parámetros 16
3.3.3 Relación señal radio SNR 17
3.3.4 Zona tiempo real 18
4 Optical Flow 19
4.1 Planteamiento 19
4.2 Clase pirámide 21
4.3 Clase Optical Flow 22
4.4 Esquema de funcionamiento 23
4.5 Resultados 24
4.5.1 Comparación entre CPU y GPU 24
4.5.2 Influencia de los parámetros 25
4.6 Comparación con Ground Truth 26
4.6.1 Zona tiempo real 27
5 Conclusiones 28
5.1 Trabajos futuros 28

Índice

Apéndice A. Desarrollo matemático

1 Planteamiento del método 30

1.1 Caso general 30

1.2 Modelo Tikhonov 30

1.3 Modelo TV-ROF 30

1.4 Modelo TV-L1 31

2 Método Primal Dual 32

3 Discretización 34

3.1 Discretización caso general 34

3.2 Discretización operadores 35

4 Denoise 36

4.1 Planteamiento 36

4.2 Discretización 36

4.3 Esquema algoritmo 38

4.4 Ecuaciones algoritmo 39

4.5 Ecuaciones algoritmo discretizadas 40

5 Optical Flow 41

5.1 Planteamiento 41

5.2 Discretización 42

5.3 Esquema algoritmo 44

5.4 Ecuaciones algoritmo 47

5.5 Ecuaciones algoritmo discretizadas 48

Bibliografía 50

MEMORIA

1 Introducción

1.1 Objetivos

El objetivo del presente proyecto es la implementación de un algoritmo para el cómputo del Optical Flow. Así mismo se desea acelerar la ejecución del mismo mediante su implementación en CUDA, hasta rangos que permitan la aplicación online a una secuencia de video o, de forma ideal, en tiempo real a partir de video capturado instantáneamente mediante una webcam.

1.2 Definición de Optical Flow

El Optical Flow consiste en la estimación del movimiento y velocidad de los distintos elementos que componen una imagen. Representa uno de los componentes fundamentales para el desarrollo de la visión por ordenador.

Las siguientes imágenes muestran un ejemplo de cálculo de Optical Flow a partir de las imágenes capturadas desde un vehículo en movimiento.

Imagen 1. Imagen capturada

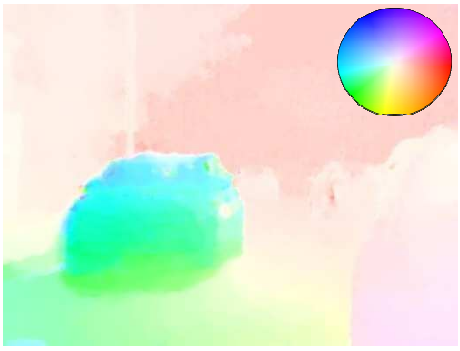


Imagen 2. Optical flow

Los resultados del algoritmo son la velocidad de cada pixel, almacenada en un vector con dos componentes. Esta información resulta demasiado densa para ser interpretada por un humano, por lo que es necesario establecer una representación de la misma. De forma usual se emplea la codificación mediante el código de colores mostrado en el círculo de la esquina superior derecha de la imagen.

Para los humanos la visión representa, sin duda, uno de principales mecanismos de percepción. Ello se debe a que la visión goza de una serie de características que la convierte en especialmente útil. Por un lado, la geometría proyectiva permite un rango de actuación a grandes distancias, potencialmente hasta el infinito. Por otro lado, la visión proporciona una gran cantidad de información del entorno y objetos que nos rodean, a través del color y la forma. Por tanto, resulta lógica la existencia de un marcado interés en que nuestras máquinas dispongan de la misma capacidad sensorial.

Este interés se ve potenciado con el desarrollo tecnológico que recientemente se ha realizado en la fabricación y calidad de cámaras digitales, que ha permitido que se conviertan en sensores habituales, relativamente baratos, ligeros y de pequeño tamaño, fácilmente integrados en gran variedad de dispositivos cotidianos.

Como aplicaciones inmediatas que pueden desarrollarse a partir de los métodos desarrollados en el presente proyecto se pueden citar, entre otros, los siguientes:

- Seguridad vial, mediante su integración en vehículos se puede dotar al mismo de un comportamiento inteligente ante la detección de obstáculos.
- Interfaz de usuario, permitiendo la interacción a distancia con programas o videojuegos, sin la necesidad de sensores adicionales a una webcam.
- Posicionamiento robots, añadiendo robustez a los algoritmos actuales.
- Mediciones industriales, en las que se requiera la determinación de la velocidad de los distintos elementos implicados en el proceso.
- Videovigilancia y detección de intrusos, mejorando la sensibilidad y precisión de los sistemas actuales de videovigilancia.

1.3 Definición de CUDA

CUDA son las siglas de Compute Unified Device Architecture, una tecnología propietaria del fabricante de tarjetas gráficas Nvidia, que permite ejecutar cálculos en la tarjeta gráfica (GPU).

Imagen 3. Logotipo Nvidia

Imagen 4. Tarjeta gráfica TESLA

CUDA contempla un compilador propio, una extensión de lenguaje C++, así como un conjunto de herramientas necesarias para codificar adecuadamente los algoritmos para su ejecución en GPU.

Las tarjetas gráficas modernas disponen de una elevada capacidad de cálculo y, especialmente, de una elevada capacidad de paralización que permiten ejecutar un gran número de hilos de forma simultánea.

Esta capacidad es fruto del desarrollo tecnológico que han sufrido estos dispositivos para desarrollar su funcionalidad principal. Efectivamente, el tratamiento y visualización de gráficos tridimensionales requieren la realización de un gran número de operaciones en coma flotante, normalmente de forma independiente por cada pixel o punto 3D en la escena. Las tarjetas gráficas han evolucionado, adaptando su arquitectura e incorporando modificaciones que les permitan desarrollar esta función.

La tecnología CUDA pone a la disposición del desarrollador esta potencia de cálculo para el uso en aplicaciones convencionales.

La importancia de la posibilidad de ejecutar cálculos mediante la GPU reside en su alta capacidad de ejecución de procesos en paralelo.

De forma habitual, la CPU está formada por un número reducido de núcleos, típicamente de 1 a 4, que disponen de un elevado número de instrucciones diseñadas para desarrollar de forma eficiente tareas de elevada complejidad. Sin embargo, esta arquitectura presenta dificultades para la ejecución de procesos con múltiples hilos.

Frente a esta filosofía de diseño, las tarjetas gráficas disponen de un elevado número de procesadores, típicamente del orden de 256 a 512, de construcción considerablemente más simple que la CPU. Adicionalmente se dispone de una serie de mecanismos, en especial relativos a la gestión de memoria, diseñados para favorecer la ejecución de un elevado número de hilos de forma paralela.

Esta se diferencia se ilustra en los siguientes esquemas que representan, de forma simplificada, ambas filosofías de arquitectura.

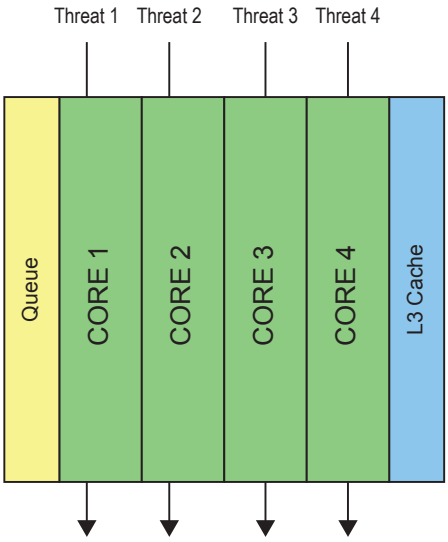


Imagen 5. Esquema CPU

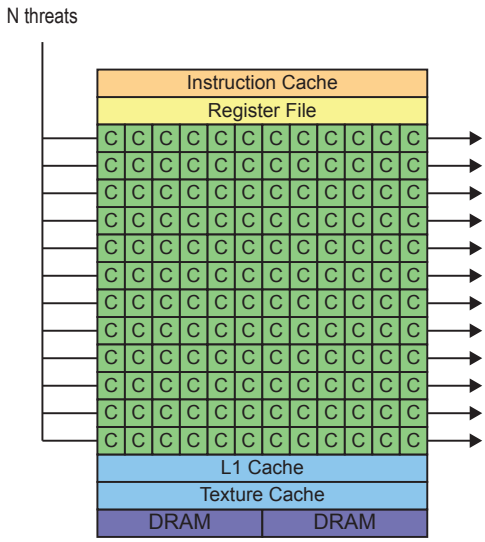


Imagen 6. Esquema GPU

El empleo de CUDA permite disponer de las ventajas de ambas arquitecturas, empleando la CPU para las instrucciones de control del programa, a la vez que se encuentra disponible la elevada potencia de cálculo de la GPU.

La implementación en CUDA resulta de gran interés en aplicaciones científicas y de ingeniería, donde se realiza un uso intensivo de cálculos. Su uso está especialmente indicado aplicaciones que realicen cálculos que requieran la ejecución de un gran número de operaciones susceptibles de ser paralelizables, como la manipulación de grandes cantidades de datos o la ejecución de operaciones con matrices.

Estas características convierten CUDA en una tecnología de gran interés para la implementación de algoritmos dentro del ámbito de la visión, donde el empleo de CUDA puede suponer una aceleración de varios órdenes de magnitud.

1.4 Trabajos realizados

Durante el desarrollo del proyecto se implementado una numerosa cantidad de clases, programadas en lenguaje C++. En todo momento se ha pretendido dotar a todos los componentes desarrollados de la versatilidad suficiente para ser fácilmente reutilizables, buscando una utilidad real más allá de la realización del presente proyecto. El flujo de trabajos se sintetiza en el siguiente esquema:

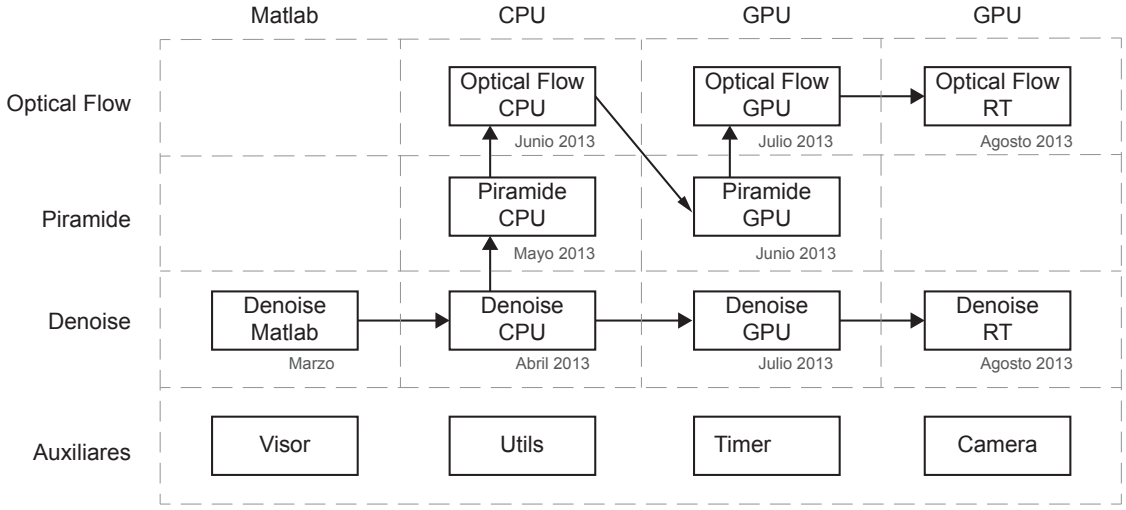


Imagen 7. Flujo y calendario de trabajos realizados

En primer lugar se ha implementado el algoritmo Denoise en Matlab. Esto ha permitido familiarizarse con las ecuaciones y la base matemática del método. Posteriormente se ha implementado la Clase Denoise en C++ para su ejecución en CPU, lo que permitió familiarizarse con los algoritmos y estructuras empleados.

Una vez verificado el correcto funcionamiento de la Clase Denoise se procede a realizar la implementación de la Clase Piramide, componente fundamental para la realización de la resolución "coarse to fine" del Optical Flow, aspecto que será explicado con detalle posteriormente. A continuación se realiza la implementación completa de la Clase Optical Flow para su ejecución en la CPU.

Posteriormente, se portan las distintas clases, Clase Denoise, Clase Pirámide, y Clase Optical Flow, para su ejecución en GPU. Finalmente, se optimiza las Clases Denoise y Optical Flow GPU, hasta conseguir su funcionamiento en tiempo real con imágenes adquiridas mediante una webcam.

Adicionalmente se han desarrollado diversas clases auxiliares necesarias para el correcto desarrollo de los programas realizados. La Clase Visor, que muestra los resultados en la interfaz mediante el empleo de la librería OpenGL. La Clase Utils, que contiene herramientas para conversión de imágenes y debug de las restantes clases. La Clase Timer, que implementa un cronómetro para medición de resultados. Por último, la Clase Camera, que realiza la captura de imágenes en un hilo paralelo no bloqueante usando QThreads, de forma que no interfiera en la ejecución del hilo principal.

1.5 Consideraciones adicionales

1.5.1 Equipo empleado

La implementación de las distintas clases que componen el presente proyecto, así como la elaboración de los diferentes experimentos y resultados obtenidos, se han realizado en un ordenador con la siguiente configuración:

- Placa base: Gigabyte X48 socket 775.
- Procesador: Inter Quad Core Q6700 2,66 Ghz.
- Memoria RAM: 8 GB DDR2.
- Tarjeta Video: GTX 650 con 2GB DDR5.
- Camara: Logitech C170 con captura de vídeo 1024 x 768 píxeles a 30 fps.
- SO: Ubuntu 12.04.

1.5.2 Igualdad de condiciones

Con objeto de que la comparación entre los tiempos obtenidos en la implementación en CPU y GPU refleje los mejores resultados que pueden ser obtenidos con cada arquitectura y, por tanto, obtener una comparación justa, el código se ha optimizado con los recursos disponibles en cada una de ellas. Esto implica que cada implementación presenta diferencias en el código, con objeto de aprovechar las características y particularidades disponibles en las distintas arquitecturas.

En el caso del CPU se dispone de los siguientes elementos,

- Manejo de punteros.
- Optimización del empleo de memoria dinámica.
- Optimización de las estructuras de flujo de control.
- Empleo de librería OpenCV.

En la implementación en GPU se cuenta con los siguientes recursos:

- Optimización de las funciones de memoria en CUDA.
- Optimización de cache y memoria compartida.
- Optimización de paralelización de procesos.
- Empleo de texturas.

1.5.3 Parámetros de simulación

Cada uno de los experimentos ejecutados para la obtención de resultados se ha realizado mediante 10 simulaciones individuales, siendo el valor mostrado la mediana de los 10 resultados obtenidos.

1.5.4 Estructura de los resultados

Para cada uno de las aplicaciones realizadas, Denoise y Optical Flow, se presentarán el planteamiento tras el algoritmo, la clase realizada y, como resultados, la comparación de tiempo de ejecución entre CPU y GPU, la influencia de los parámetros en el tiempo de ejecución, la eficacia en la resolución del problema, y combinación de parámetros que permite conseguir su ejecución en tiempo real.

2 Fundamentos matemáticos

Los fundamentos matemáticos en los que se basa el presente proyecto están basados en la Tesis doctoral de Thomas Pock de la Universidad de Graz de enero de 2008 [TP08], que posteriormente serían ampliados en un artículo publicado en mayo de 2010, realizado por el mismo autor en colaboración con Antonin Cambolle [TA10].

Existe un marcado interés dentro del ámbito de la visión por computador en el estudio de los métodos variacionales debido a que su planteamiento y capacidad de modelado se ajusta con elevado éxito a un amplio tipo de problemas encontrado de forma habitual en visión por computador, tales como denoising, inpainting, deblurring, optical flow, segmentación y reconstrucción 3D, entre otros.

Las técnicas y algoritmos presentados en estos artículos presuponen un conocimiento de las herramientas y fundamentos matemáticos en los que se basan. Desafortunadamente no se ha encontrado un compendio o libro en que explique de forma unificada estos conceptos. Por tanto se considera interesante proporcionar en este proyecto una explicación intuitiva sobre la base la base matemática y el funcionamiento de estos métodos. Una explicación extendida de los algoritmos se presenta en el Apéndice Matemático adjunto al presente documento.

Los métodos variacionales aplicados a imágenes consisten, de forma general, en el estudio de la resolución de familias de ecuaciones del tipo:

$$\min_u \left\{ \underbrace{\int_\Omega F(\nabla u)}_{\text{Término regularizador}} + \underbrace{\lambda \int_\Omega G(u)}_{\text{Término fidelidad}} \right\}$$

El interés que subyace a la resolución de este tipo de problemas de minimización reside en una suposición lógica de que las imágenes poseen una relativa suavidad. Por tanto, resulta interesante minimizar la variación en las mismas.

La ecuación presentada dispone de dos términos diferenciados,

- El término regularizador penaliza la presencia de variaciones dentro de la imagen. Por tanto, favorece la obtención de soluciones suaves.
- El término de fidelidad, o data term, aporta las condiciones de entrada, penaliza las soluciones que se desvían de las mismas.

La relación entre ambos comportamientos, el carácter suave impuesto por el término regularizador, y el ajuste a los datos de entrada impuesto por el término de fidelidad, se regula a través del peso del parámetro lambda

El método propuesto presenta, como se ha comentado, importantes aplicaciones dentro de un amplio rango de problemas típicos en visión por computadora. La diferencia entre su aplicación a una u otro caso viene dada por la forma adoptada por las funciones F y G, que deben ser particularizadas para cada aplicación en concreto.

Tomando como ejemplo para la explicación su aplicación al filtrado de ruido, por ser el caso más sencillo, aplicando una regularización Tikhonov el problema variacional general pasa a estar expresado de expresado de la siguiente forma:

$$\min_u \left\{ \int_\Omega (\nabla u)^2 d\Omega + \lambda \int_\Omega (u-f)^2 d\Omega \right\}$$

En 1992 los autores Rudin, Osher y Fatemi [LO92], estudian la sustitución de la norma cuadrática del término regularizador por la norma L1, con lo que el primer término se convierte en la **variación total** |∇u| de la función a minimizar, dando lugar al método TV-ROF (Total Variation Rudi-Osher-Fatemi).

$$\min_u \left\{ \int_\Omega |\nabla u| + \lambda \int_\Omega (u-f)^2 d\Omega \right\}$$

Finalmente, los métodos TV-L1, sustituyen ambos términos por la norma L1.

$$\min_u \left\{ \int_\Omega |\nabla u| + \lambda \int_\Omega |u-f| d\Omega \right\}$$

El interés tras la sustitución de la norma cuadrática en el data term por la norma L1 puede entenderse como analogía con los métodos de ajuste robusto en estadística.

En cuanto a la variación total, la siguiente figura ilustra el significado de |∇u| en los métodos variacionales. Se observa que la variación total de la función escalón (azul), es idéntica a la variación total de la función dentada (rojo). En general, todas las funciones monótonas con idénticos extremos presentarán la misma variación total. Por el contrario, la norma cuadrática del gradiente de la función penalizaría más la gráfica escalón, que la gráfica dentada y a su vez que la función lineal por lo que no permite la existencia de saltos bruscos

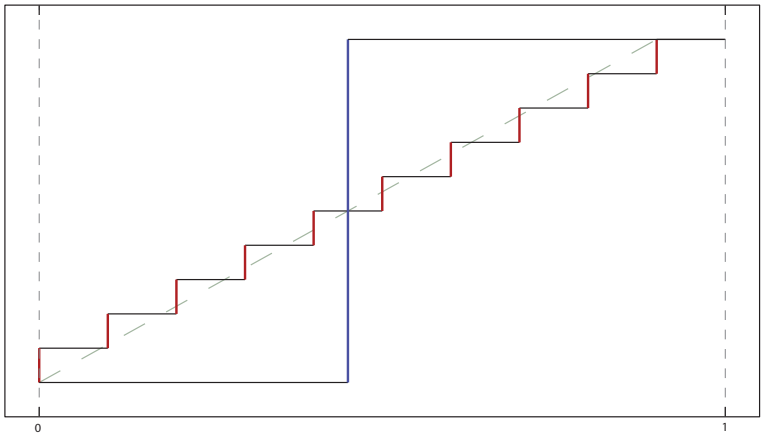


Imagen 8. Efecto de la norma L1 en métodos variacionales

Por tanto, el principal interés tras la sustitución de la norma cuadrática en el término regularizador en problemas variacionales en imágenes es permitir la existencia de zonas suaves sin penalizar en exceso, y por tanto conservando, los contornos.

Las siguientes imágenes, obtenidas del artículo [A01], comparan los resultados obtenidos en la aplicación de los distintos algoritmos en la eliminación de ruido.

En primer lugar comparamos Tikhonov con TV-ROF para imágenes a las que se les ha añadido ruido independiente gaussiano en cada pixel.

Original

Imagen con ruido

Resultado con Tikhonov

Resultado con TV-ROF

Se observa como el modelo TV-ROF es capaz de reconstruir con mayor precisión la imagen original de la imagen, manteniendo los contornos, mientras que el modelo Tikhonov presenta un difuminado general de la imagen.

A continuación se compara el comportamiento del TV-ROF frente al TV-L1 para el caso en que la imagen, además de ruido gaussiano, tiene ruido de sal y pimienta es decir datos espurios.

Original

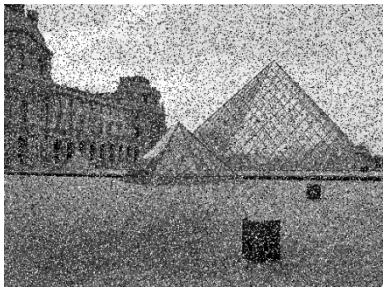


Imagen con ruido

Resultado con TV-ROF

Resultado con TV-L1

El método TV-L1, al tener una norma absoluta en el término de fidelidad, presenta mayor inmunidad a la presencia de espurios, a la vez que el regularizador sigue preservando los bordes de la imagen. El método TV-ROF se ve más afectado por espurios dado que no es robusto, obteniendo una imagen con menor precisión.

Pese a sus prestaciones superiores, la sustitución de las normas cuadráticas por el valor absoluto tiene como consecuencia el incremento de la dificultad en la resolución debido a la introducción de funciones no diferenciables. Para solventar este problema se emplea un algoritmo denominado Primal Dual, que permite hallar el mínimo absoluto de problemas del tipo que nos ocupan.

Retomando el caso general, su formulación discretizada resulta,

$$\min_u F(Ku) + G(u)$$

donde ambas funciones F y G son convexas, aunque no necesariamente diferenciables (por ejemplo, en el caso de usar valor absolutos en una o ambas de ellas) y K es la versión discreta del operador gradiente.

A continuación, para facilitar la explicación, supongamos el modelo ROF, donde F es no diferenciable. La base del método es usar la transformación de Legendre-Fenchel para sustituir las funciones no diferenciables por una expresión equivalente. De forma general, la transformación de Legendre-Fenchel tiene la siguiente expresión

$$F^*(p) = \sup_x < x, p > -F(x)$$

Una versión intuitiva del concepto es considerar que <p,x> es la formulación de un hiperplano que pasa por el origen, cuya orientación está determinada por el vector p. De esta forma, F*(b) puede ser interpretado como la ordenada en el origen que debe tener un hiperplano de orientación p para quedar siempre por debajo de F(x).

Esto permite interpretar F(x) como la envolvente definida por hiperplanos afines denominada conjugada convexa [SB04] como se ilustra en el siguiente esquema.

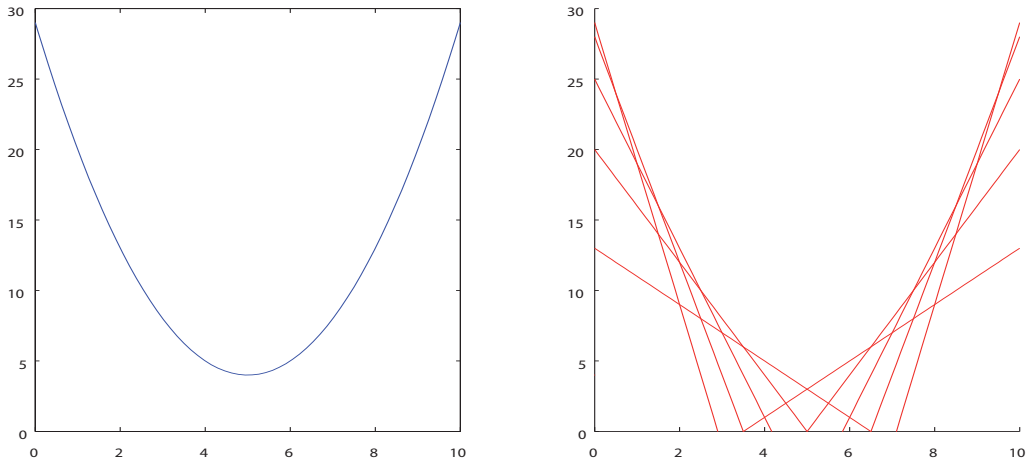


Imagen 9. Interpretación de la conjugada convexa.

Se hace notar que la definición de la misma depende fuertemente de las derivadas de F(x). La conjugada convexa tiene importantes propiedades y aplicaciones dentro del campo de los problemas de optimización. Por ejemplo, la biconjugada de cualquier función es una función convexa, que representa el menor conjunto convexo que envuelve a la función original, denominada "Convex Hull". De esta forma, la biconjugada de una función F coincide con si misma si y solo si F es convexa y semicontinua inferiormente.

La aplicación del método a la norma L1 resulta en la siguiente expresión,

$$F^*(p) = \begin{cases} 0 & |p| \leq 1 \\ \infty & |p| > 1 \end{cases}$$

cuya gráfica es la siguiente,

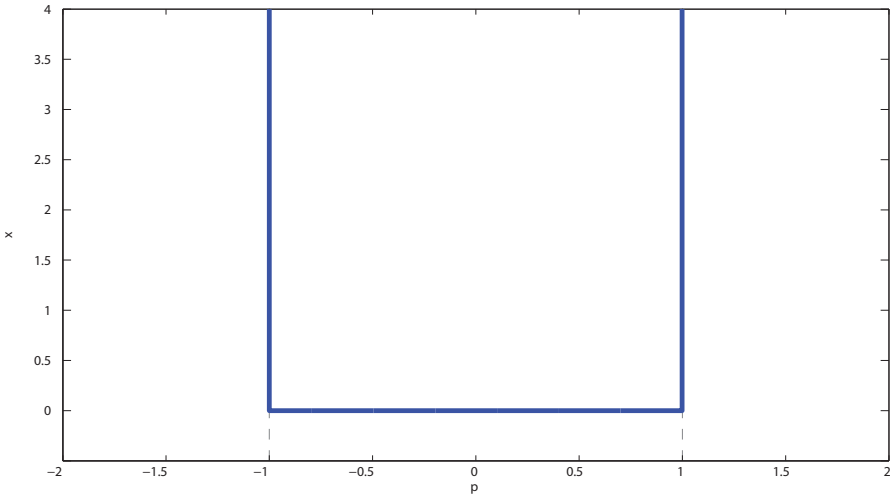


Imagen 10. Formulación dual de la función valor absoluto

Para ilustrar el resultado obtenido, así como la dependencia con la visión intuitiva anterior de la conjugada convexa, consideremos la siguiente imagen,

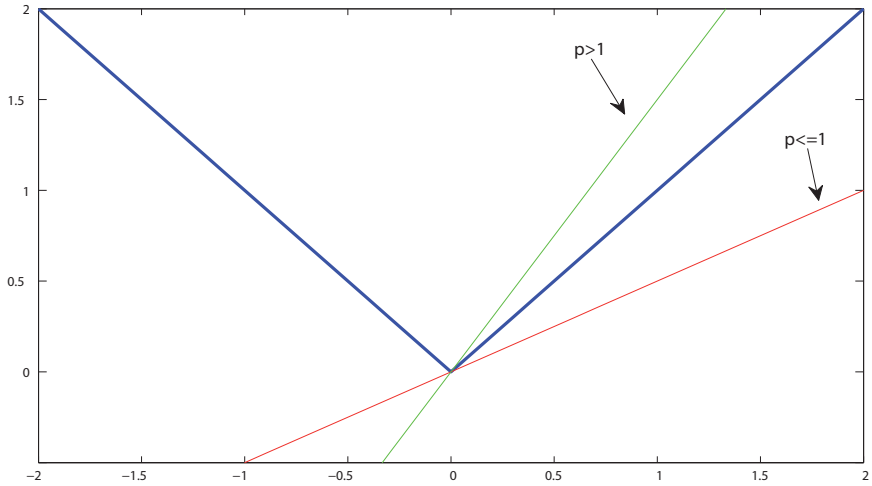


Imagen 11. Interpretación de la conjugada convexa del valor absoluto.

La función valor absoluto está compuesta por únicamente dos pendientes, de valores -1 y 1. Para direcciones p inferiores a -1 o superiores a 1, cualquier hiperplano resultante quedará por encima de |x| al tender x a infinito, y la diferencia entre la función y el hiperplano tenderá a infinito, por lo que F*(x) resulta infinito. Únicamente para direcciones tales que |p|≤1 el hiperplano resultante es un subestimador de F(x), siendo el máximo valor de la ordenada del mismo, y por tanto F*(x), el origen 0.

La aplicación del método a la norma L1 resulta en la siguiente identidad:

$$|\nabla u| = max\{p \cdot \nabla u : ||p|| \leq 1\}$$

De esta forma se sustituye la función norma L1 por una función derivable. Sin embargo se añade la dificultad de introduce una variable dual, y de una etapa adicional de maximización con restricciones |p|<=1.

Para la resolución del problema general ampliado, tras la sustitución de la norma L1 por su formulación dual equivalente, Thomas Pock propone la resolución mediante el método del Proximal Map [RF70] el cual se sintetiza en el siguiente esquema,

$$\begin{cases} y^{n+1} & = & (I + \sigma\partial F^*)^{-1}(y^n + \sigma K\hat{x}^n) \\ x^{n+1} & = & (I + \tau\partial G)^{-1}(y^n + \tau K^* y^{n+1}) \\ \hat{x}^{n+1} & = & x^{n+1} + \theta(x^{n+1} - x^n) \end{cases}$$

donde las expresiones del tipo $(I + \tau\partial F)^{-1}$ precisamente suponen el Proximal Map, cuya definición formal se recoge en la siguiente expresión,

$$x = (I + \tau\partial F)^{-1}(y) = \arg\min_x \left\{ \frac{||x - y||^2}{2\tau} + F(x) \right\}$$

En particular, para el problema de Denoising el Proximal Map admite una interpretación sencilla como un algoritmo de gradiente ascendente (dual) / descendente (primal), en los que el gradiente se evalúa en el punto posterior, mientras que se garantiza que las restricciones cumplen (|p|<=1).

Por otro lado, si bien en un problema genérico de optimización no se puede garantizar encontrar el mínimo absoluto, para la clase de problemas con estructura que estamos considerando, en particular para F y G convexas, la metodología presentada permite encontrar el mínimo absoluto.

La razón de emplear el método Primal-Dual frente a otros de orden superior, como Newton Rapson, es que su implementación resulta más eficiente y, especialmente, que resulta susceptible de ser ejecutada en paralelo. Por tanto aunque el orden de convergencia del método es inferior, la eficiencia de la implementación y la posibilidad de posterior aceleración mediante su ejecución en GPU compensan con creces el mayor número de pasos necesarios para alcanzar la solución.

3 Denoise

3.1 Planteamiento

Particularizando el caso general para el filtrado de ruido en imágenes, como se ha adelantado con anterioridad al explicar los fundamentos matemáticos, en primer lugar se presenta el método TV-ROF cuya expresión es la siguiente:

$$\min_u \left\{ \int_\Omega |\nabla u| + \lambda \int_\Omega (u-f)^2 d\Omega \right\}$$

Donde,

- u es la solución al método, es decir, la imagen tras el filtrado de ruido.
- g es la imagen introducida en el algoritmo, es decir, la imagen que se desea filtrar.
- El término regulador tiene la forma habitual en los métodos de variación total.
- El término de fidelidad resulta la norma cuadrática de la diferencia entre la imagen filtrada u, y la imagen original con ruido f.

Intuitivamente, el problema de minimización tras el filtrado de ruido se encuentra determinado por la relación entre ambos términos. El término regularizador intenta imponer una suavidad en la imagen, mientras que el término de fidelidad intenta ajustar la imagen a la imagen original introducida en el algoritmo.

La transición entre ambos comportamientos se controla mediante el peso del parámetro lambda. Si lambda toma un valor bajo, la imagen estará formada gradientes suaves, pero se habrá perdido todo el detalle de la imagen. Si lambda toma valores altos, la solución se ajustará a la imagen original, pero la eliminación de ruido será inexistente.

Por su parte, el método TV-L1 resulta similar, sustituyendo la norma cuadrática en el término por regulador por la robusta norma L1.

$$\min_u \left\{ \int_\Omega |\nabla u| + \lambda \int_\Omega |u-f| d\Omega \right\}$$

Ambos modelos son susceptibles del empleo de una sustitución como la presentada anteriormente mediante el empleo de variables duales, que puede ser resuelto con mediante una aproximación Primal Dual comentado con anterioridad.

Los detalles del algoritmo se presentan en Apéndice A Desarrollo Matemático, en el apartado 4.

3.2 Clase Denoise

La Clase Denoise implementa los métodos ROF y TV-L1 para el filtrado de ruido. Pone a disposición del desarrollador métodos públicos para las siguientes tareas.

- Inicializar el objeto a un tamaño de imagen determinado.
- Introducir imagen a filtrar
- Ejecutar el algoritmo de filtrado.
- Obtener la imagen filtrada.

La siguiente pantalla muestra el entorno de pruebas realizado para la Clase Denoise. El mismo permite cargar una imagen y realizar el proceso de eliminación de ruido. Los visores muestran la imagen antes y después del filtrado. Por su parte, los Sliders permiten variar la influencia del parámetro lambda y el número de iteraciones.

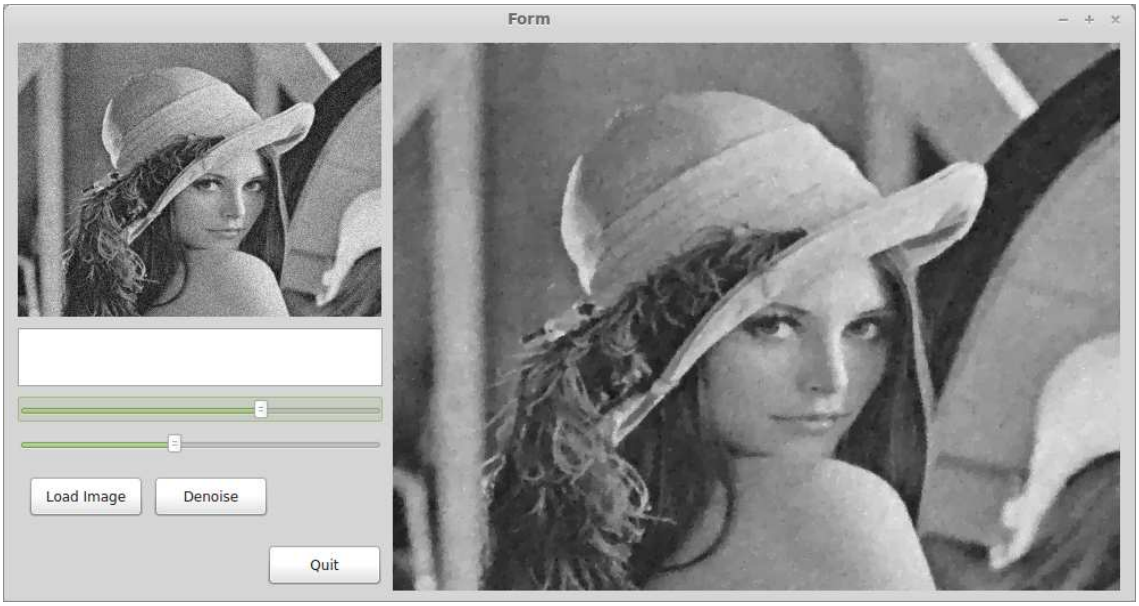
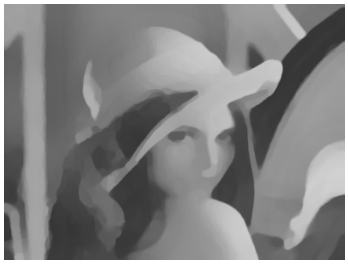


Imagen 12. Entorno de pruebas Clase Denoise

Las siguientes imágenes muestran el resultado de variar el parámetro lambda, que controla la transición entre término regularizador y término fidelidad.



Lambda 2

Lambda 8

Lambda 20

3.3 Resultados

3.3.1 Comparación entre CPU y GPU

La siguiente tabla recoge los tiempos de ejecución en milisegundos de las distintas etapas de la aplicación, para un tamaño de imagen fijo de 640x480 pixels y con 40 iteraciones de cálculo de Denoise.

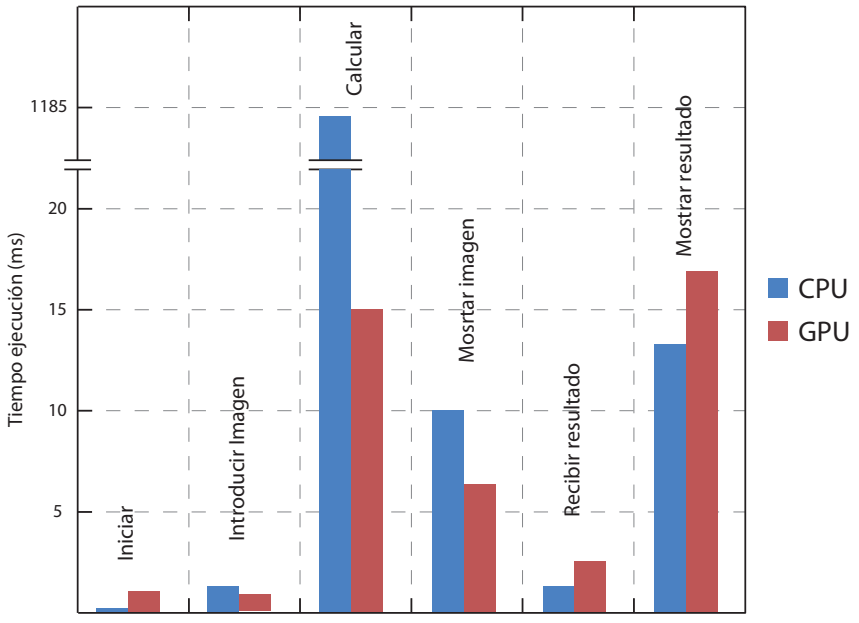
	Individual		Secuencial	
Concepto	**CPU**	**GPU**	**CPU**	**GPU**
Iniciar Denoise	0,114	1,000	-	-
Introducir imagen	1,370	0,919	1,370	0,919
Mostrar imagen	17,368	15,068	-	-
Calcular Denoise	1184,58	6,338	1184,58	6,338
Resultado recibido	1,155	2,571	1,155	2,571
Mostrar resultado	13,341	6,624	-	-
Total	**1217,928**	**42,827**	**1187,105**	**9,828**
Relación CPU/GPU		**28,44**		**120,79**

Tabla 1. Comparación tiempos (ms) entre CPU y GPU

Se observa que la implementación en GPU es 28,44 veces más rápida para el cálculo de una única imagen, incluyendo las etapas de inicialización de los objetos Denoise. Se hace notar que este resultado se muy penalizado por la etapa de mostrar ambas imágenes, que es una etapa relativamente lenta y que no puede ser acelerada mediante CPU, estando limitada por la librería OpenGL.

En caso de ejecución continua, para una secuencia de imágenes, el cálculo resulta 120,79 veces más rápido. En particular, la etapa específica de cálculo del Denoise resulta 186,90 veces más rápida en la implementación realizada en GPU.

Estos valores se muestran en la siguiente tabla. Se hace notar que la columna de la fase de cálculo en CPU, muy superior a las demás, no se representa a escala.

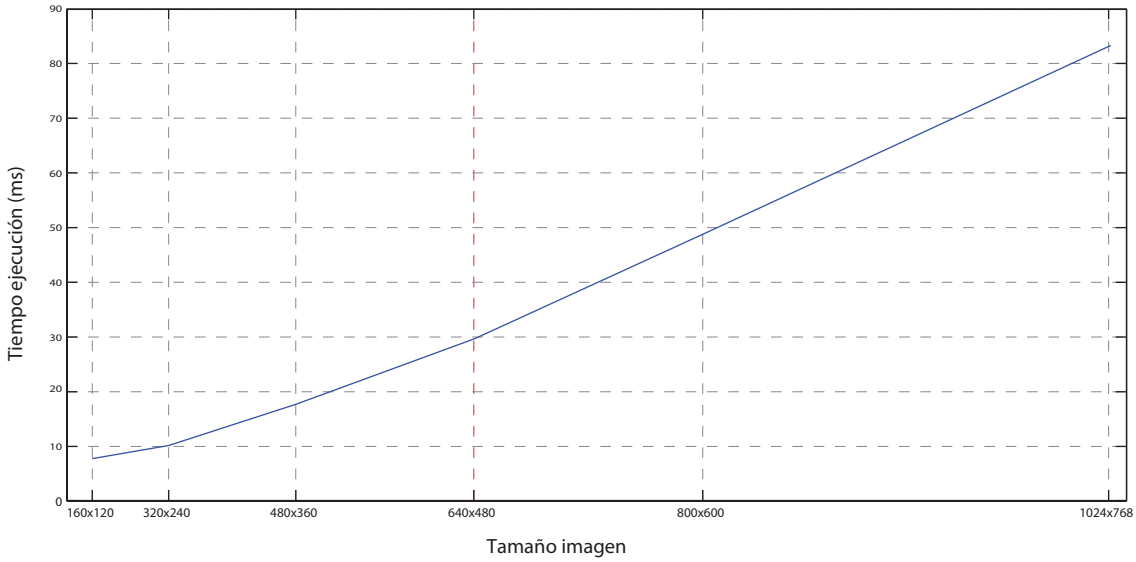


Gráfica 1. Comparación tiempos (ms) entre CPU y GPU

3.3.2 Influencia de los parámetros

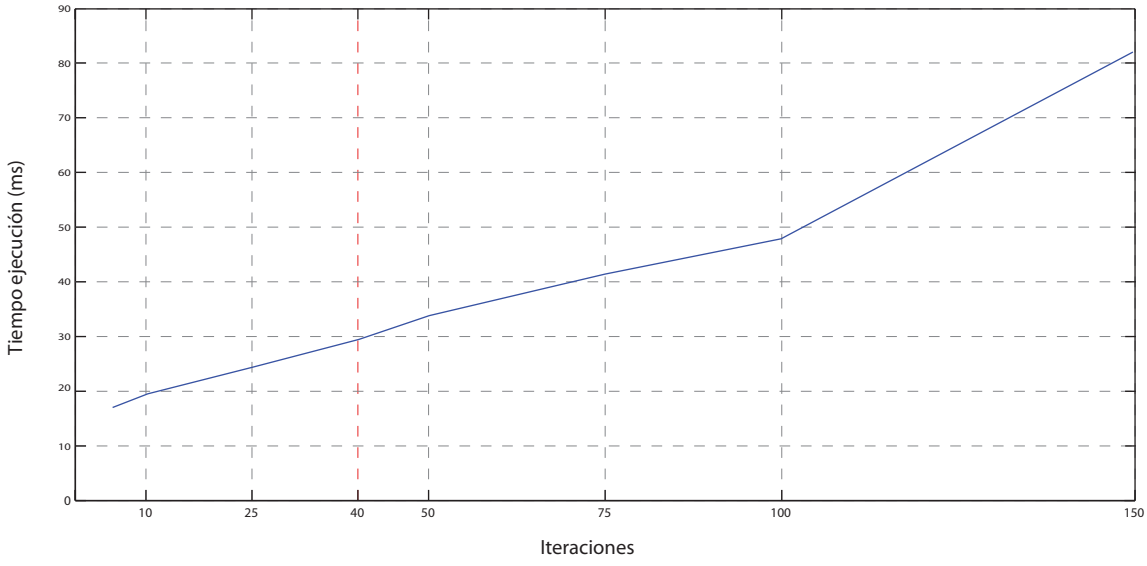
A continuación se diseña una serie de experimentos para caracterizar la influencia de los distintos parámetros que intervienen en el algoritmo en el tiempo de ejecución global del mismo. Los tiempos representados muestran el proceso completo, incluyendo captura de imágenes, cálculo de Denoise, y visualización de resultados.

En primer lugar se determina la relación con el tamaño de la imagen, medida en número de pixels.



Gráfica 2.Relación entre tiempo (ms) y tamaño de imagen.

A continuación se determina la relación con el número de iteraciones realizada en el proceso de eliminación de ruido.



Gráfica 3.Relación entre tiempo (ms) e iteraciones.

3.3.3 Relación señal ruido SNR

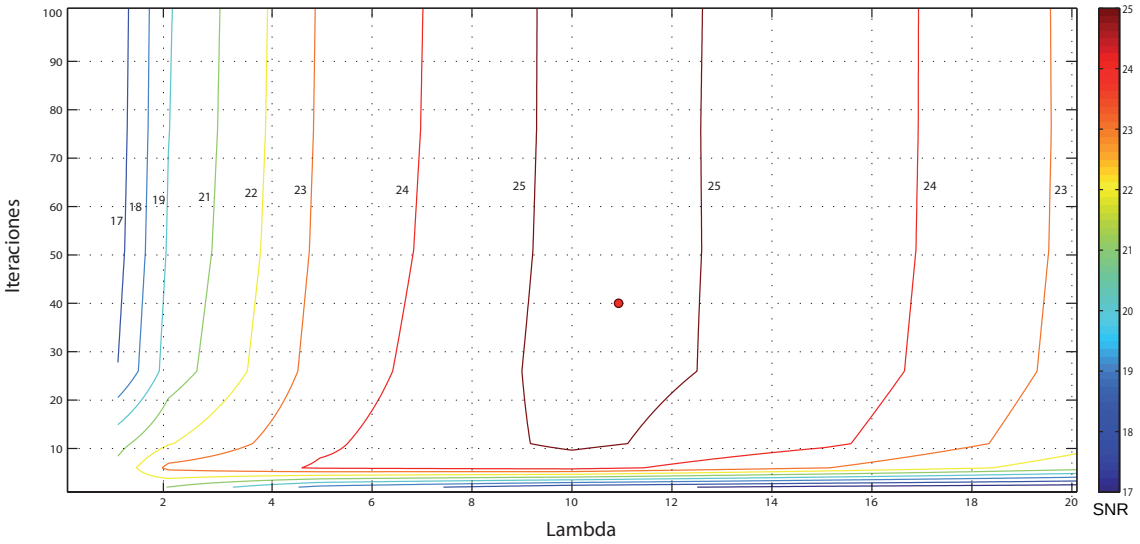
Se desea determinar la eficacia en la ejecución del filtrado de la imagen, y la influencia de los parámetros en la misma. Para ello se realiza el filtrado de ruido a una imagen de muestra normalizada a valores de gris entre 0 (negro) y 1 (blanco), a la que se ha añadido de forma intencionada una señal de ruido de distribución Gaussiana de media 0 y varianza 0,1.

La eficacia en el filtrado se caracterizará mediante la Relación Señal Ruido SNR (Signal Noise Ratio), calculada mediante la siguiente expresión

$$\mathrm{SNR}_{db} = 10 \cdot \log_{10}\left(\frac{A_{signal}}{A_{noise}}\right)^2 = 20 \cdot \log_{10}\left(\frac{A_{signal}}{A_{noise}}\right)$$

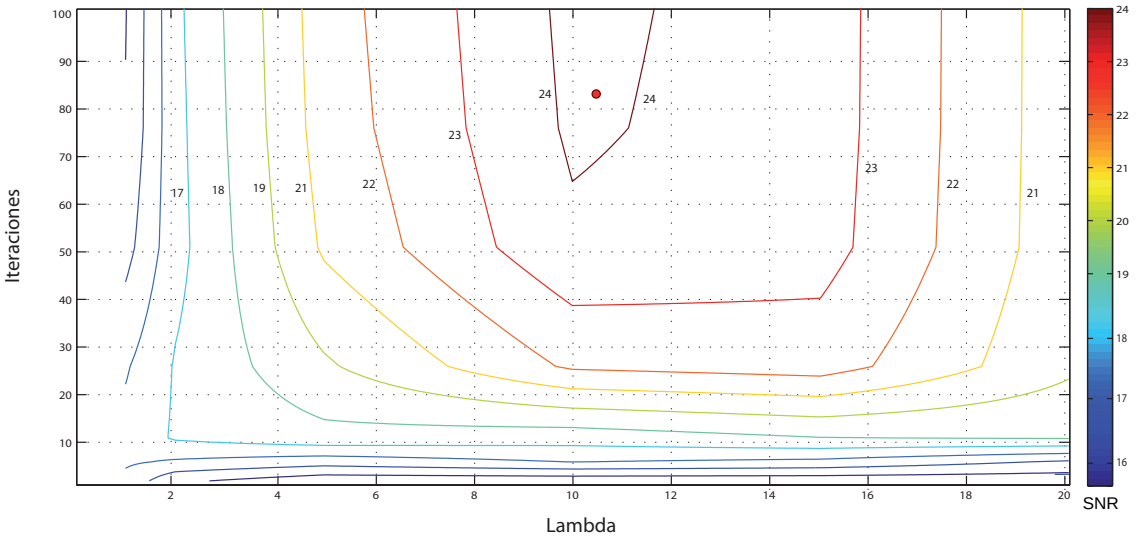
siendo A la media cuadrática (RMS), Señal la imagen original previa a añadir el ruido, y Noise la diferencia entre la imagen filtrada y la imagen original.

Los resultados para el algoritmo ROF se muestran en la siguiente gráfica,



Gráfica 4. SRN algoritmo ROF

La siguiente gráfica muestra los resultados para el algoritmo TV-L1.



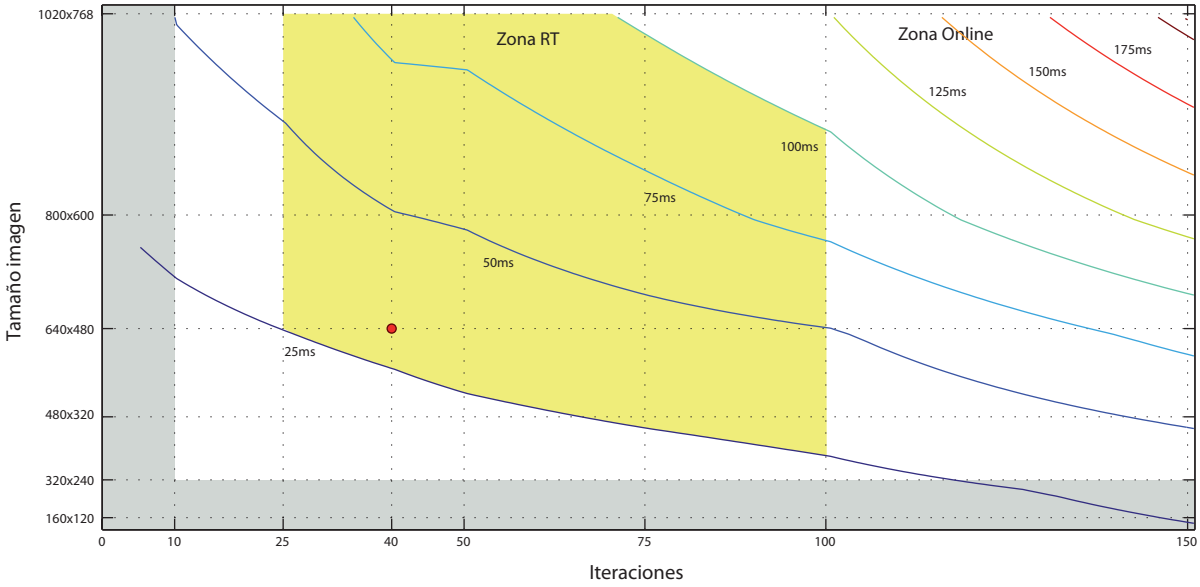
Gráfica 5. SRN algoritmo TV-L1

3.3.4 Zona tiempo real

Finalmente, se desea caracterizar la relación entre tiempo, tamaño de imagen e iteraciones, para determinar una zona dentro de la cual la se consigue la ejecución en tiempo real. A estos efectos, se considera tiempo real a velocidades de refresco de 10-20 frames por segundo, lo que implica tiempos de ejecución de 50 a 100 ms por frame.

Para ello se diseña y ejecuta una serie muti variable de experimentos. Se ha optado por este método, en lugar de la interpolación a partir de los resultados previos, para eliminar la posible interferencia entre los efectos de ambos parámetros.

Los resultados obtenidos se muestran en la siguiente tabla, donde se ha sombreado en amarillo la zona de tiempo real.



Gráfica 6. Zona tiempo real Denoise

Se comprueba que con la implementación realizada se consigue tiempo real en un amplio rango de resoluciones, incluso a 1024x768 pixels. La ejecución con menos de 20 iteraciones en el método resulta desaconsejable dado que supone una merma en la calidad del filtrado, sin conseguir una mejora sustancial de la velocidad de refresco. Igualmente, la ejecución de más de 100 iteraciones resulta innecesaria, dado que resulta inapreciable la mejora de la solución obtenida.

Por tanto se considera que con la implementación realizada y el equipo empleado el punto óptimo de funcionamiento es 640 x 480 a 30 fps con 40 iteraciones por nivel, lo que permite elevadas tasas de refresco y una alta calidad en el resultado conseguido.

No obstante se hace notar, nuevamente, que la mayor parte del tiempo de ejecución se emplea en adquisición mediante webcam y en la muestra de resultados. Una aplicación que actuara sin estos procesos, por ejemplo realizando el filtrado de una secuencia de video, conseguiría tasas considerablemente superiores.

Igualmente la ejecución en un equipo con una tarjeta gráfica superior aumentaría de forma significativa la velocidad de ejecución.

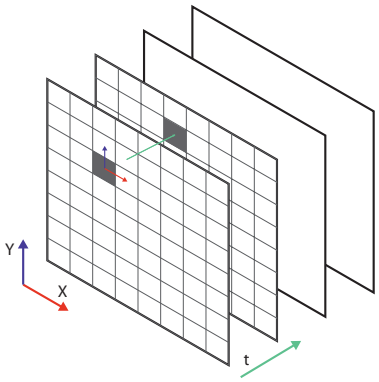
4 Optical Flow

4.1 Planteamiento

La particularización del método general para la resolución del Optical Flow no resulta tan inmediata e intuitiva como para el caso del filtrado de ruido.

En primer lugar, consideremos que una secuencia de imágenes como una función que codifica la intensidad de cada pixel mediante una expresión que depende tanto de las coordenadas espaciales como del tiempo.

$$Imagen = I(x(t), t)$$



La restricción típica del Optical Flow es la siguiente:

$$\frac{d}{dt} I(x(t), t) = 0$$

Intuitivamente, la capacidad de seguimiento tras la restricción del Optical Flow reside en la suposición de que un pixel puede desplazarse temporal o espacialmente entre frames, pero sin modificar su intensidad durante el proceso.

Operando matemáticamente, proceso que se detalla en Apéndice Matemático, la restricción puede expresarse mediante la siguiente expresión,

$$\rho(v) = I_t + (\nabla I)^T (v - v_0)$$

Donde v es un campo vectorial de dos componentes que contiene estimación de movimiento, y v_0 las condiciones iníciales para el cálculo.

No obstante, la condición de Optical Flow resulta extremadamente restrictiva, ya que presupone que la intensidad de los pixels permanece constante en todo momento, lo cual no se cumple de forma habitual. Por tanto, se amplía añadiendo un término adicional que recoge las variaciones temporales en la iluminación, cuyo peso se controla mediante el parámetro beta.

$$\rho(u, v) = I_t + (\nabla I)^T (v - v_0) + \beta u$$

Donde u es un campo escalar que contiene la variación de la iluminación.

Por tanto el método general particularizado a la expresión del Optical Flow resulta,

$$\min_v \int_\Omega |Du| + \int_\Omega |Dv| + \lambda ||\rho(u, v)||$$

que puede ser resuelto mediante un algoritmo Primal-Dual.

Los detalles del algoritmo se presentan en Apéndice A Desarrollo Matemático, en el apartado 4.

Sin embargo, la resolución del Optical Flow presenta una importante complicación adicional. Al ser un problema fuertemente no lineal y no convexo, no se garantiza la convergencia del método para grandes desplazamientos en la imagen.

Para ilustrar esta situación se presenta la siguiente figura, donde se representa una función arbitraria con numerosos mínimos locales y un mínimo global.

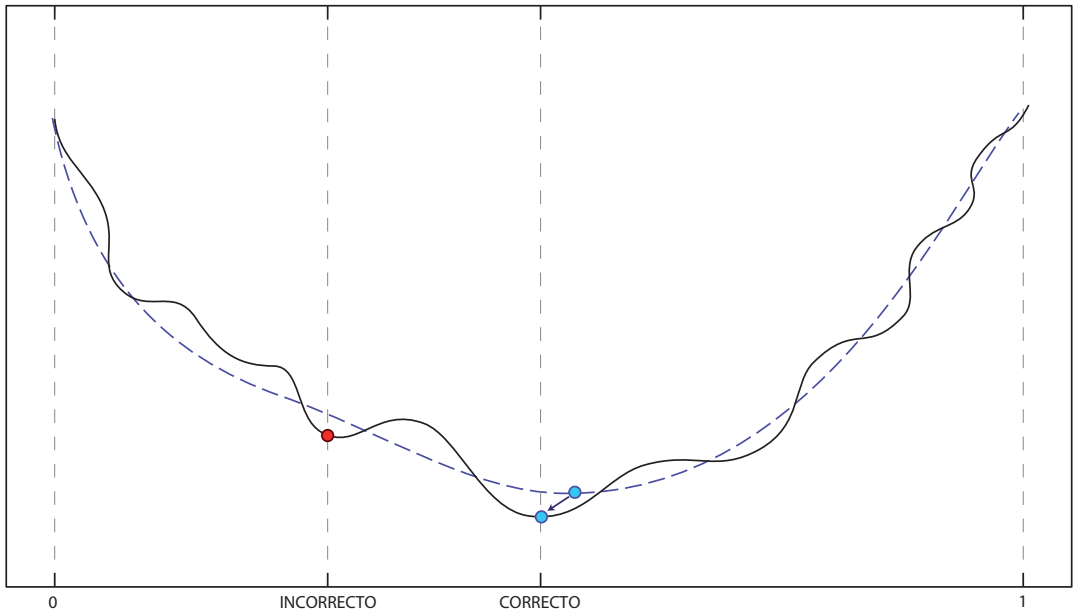


Imagen 13.Esquema mínimo local

Ante unas condiciones iniciales arbitrarias, típicamente una matriz nula, la solución puede alcanzar un mínimo local representado por el punto rojo. Sin embargo si suavizamos la función, representada mediante la línea discontinua, y tomando la solución a esta función suavizada como condiciones iniciales para el siguiente paso, se mejora sustancialmente la convergencia del método hacia un mínimo global.

La forma de adoptar esta metodología al problema del Optical Flow de forma que mejoremos la convergencia del algoritmo se hace a través de una aproximación del tipo “coarse to fine”. Para ello se genera una secuencia de imágenes reduciendo el tamaño entre niveles a la mitad, hasta que el lado menor sea 16 pixels. Posteriormente se resuelve el Optical Flow para cada una de las imágenes, empezando por las de menor tamaño. En cada nivel, por supuesto, se realiza el número de iteraciones establecido. Finalmente la solución obtenida en cada nivel se expande y se toma como condición inicial para el siguiente nivel.

De esta forma se consigue, de forma intuitiva, eliminar el detalle y las altas frecuencias de las imágenes de entrada al algoritmo. El cálculo de los niveles superiores caracterizan el movimiento de grandes zonas de la imagen (bajas frecuencias) y cada nivel posterior añade detalle progresivamente a la solución. De esta forma, si bien sigue sin estar garantizada la convergencia, se favorece la misma de forma considerable, en especial ante la presencia de grandes desplazamientos.

4.2 Clase pirámide

La Clase Pirámide es la encargada de calcular y almacenar las imágenes y soluciones empleadas en la aproximación "coarse to fine" del Optical Flow. Se dispone de métodos para introducir u obtener imágenes en cualquier nivel, así como rellenar los niveles de la pirámide en sentido ascendente o descendente.

En los pasos ascendentes cada pixel se calcula como media de los cuatro pixeles inferiores con objeto de eliminar, o al menos reducir, la aparición de fenómenos de Aliasing. La propagación en sentido descendente se realiza simplemente mediante el vecino más cercano. También se ha estudiado el empleo de interpolación bilineal o bicúbica para ambos procesos. Estas soluciones han sido descartadas dado que no se observa una mejora de la convergencia, frente a la reducción en la velocidad de ejecución que suponen.

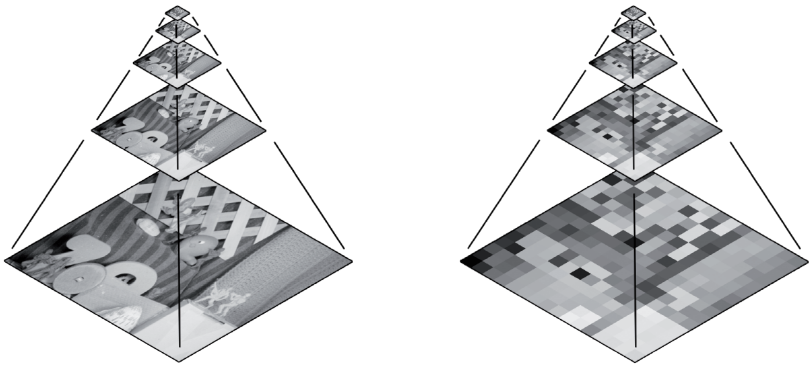


Imagen 14.Esquema funcionamiento Clase Pirámide

La siguiente pantalla muestra el entorno de pruebas realizado para la Clase Pirámide. El mismo carga una imagen en un objeto pirámide, que se rellena en sentido ascendente. El nivel superior se copia a una segunda pirámide y se rellena en sentido descendente. Los sliders permiten desplazarse entre niveles de ambas pirámides.

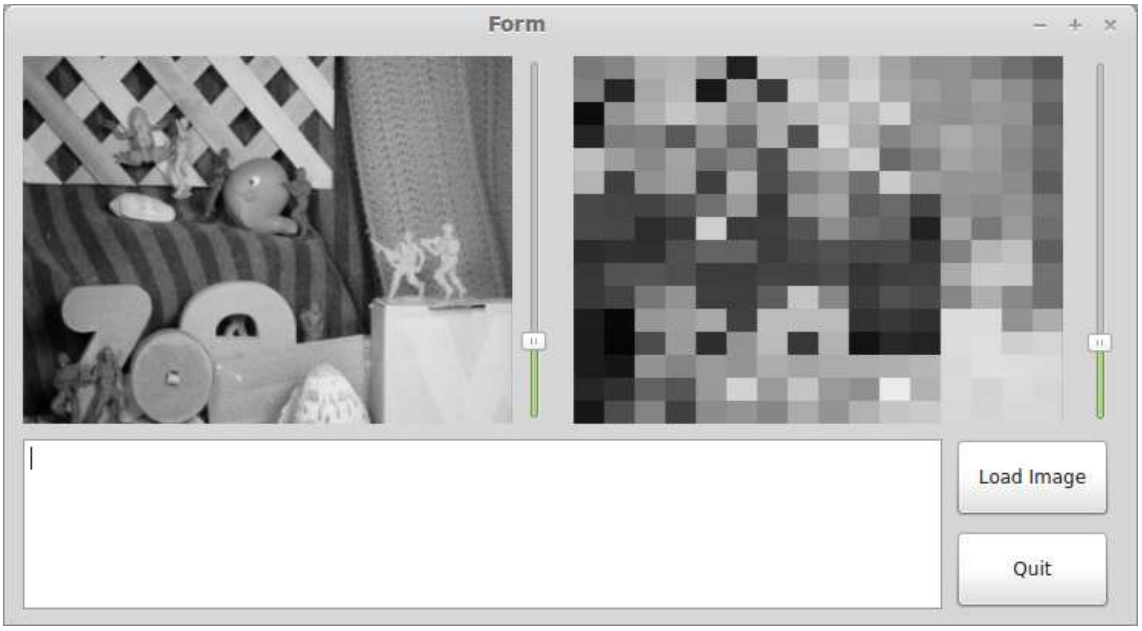


Imagen 15. Entorno de pruebas Clase Pirámide.

4.3 Clase Optical Flow

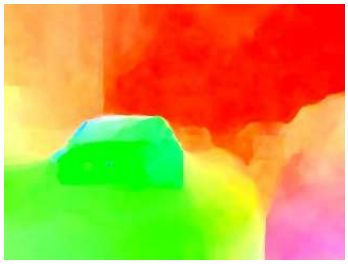
La Clase Optical Flow implementa el algoritmo de estimación de movimiento. Dispone de métodos para inicializar el objeto a un tamaño de imagen determinado, introducir ambas imágenes de entrada, ejecutar el cálculo del Optical Flow y obtener la representación de la velocidad y el cambio en iluminación.

La siguiente pantalla muestra el entorno de pruebas realizado para la Clase Optical Flow, donde se cargan dos imágenes y se calcula la estimación del movimiento. Los slider permiten variar los parámetros lambda, beta, theta y número de iteraciones.

Imagen 16.Entorno de pruebas Optical Flow

Las imágenes que se muestran en la parte inferior son el campo de velocidad calculado y el resultado del cambio de iluminación. La velocidad se codifica por colores, donde la dirección del movimiento determina el color y la magnitud del mismo la saturación. Esta representación, que resulta habitual, ya se había introducido al comienzo del presente documento. Ambos resultados están normalizados entre sus valores mínimos y máximo para permitir una visualización adecuada.

Las siguientes imágenes muestran el resultado de variar el parámetro lambda, que controla la transición entre término regularizador y término fidelidad.



Lambda 4

Lambda 10

Lambda 20

4.4 Esquema de funcionamiento

La siguiente figura muestra el esquema de funcionamiento del algoritmo, en especial la aproximación “coarse to fine”, y el papel de la Clase Pirámide en el mismo.

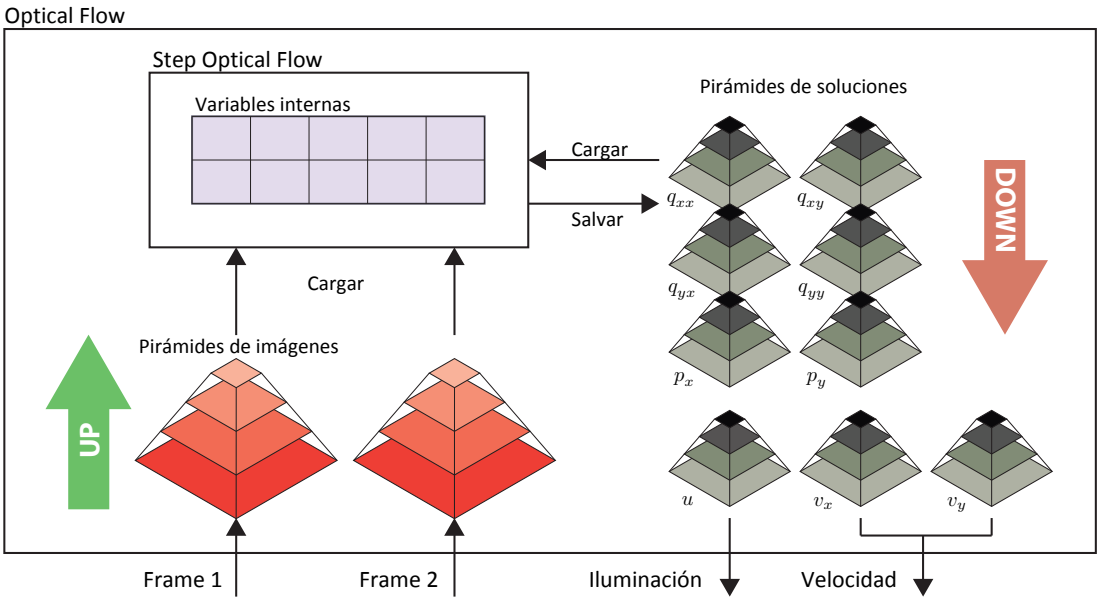


Imagen 17. Esquema de funcionamiento Optical Flow

Durante la fase de inicialización se generan las pirámides necesarias para almacenar las dos imágenes de entrada y todas las soluciones. Estas estructuras se crean al principio, de forma que durante la ejecución continua no sea necesario la reserva de memoria dinámica adicional, lo cual es un proceso costoso.

Al introducir las imágenes de entrada en la base de ambas pirámides, estas se rellenan en sentido ascendente y se inicializa el nivel superior de las pirámides de soluciones a cero. Posteriormente, empezando por el nivel superior y de forma iterativa, se cargan las variables del nivel actual desde las distintas pirámides y se realiza el cálculo del Optical Flow con el número de iteraciones establecido para el cálculo de cada nivel. Finalmente se guardan los resultados en las pirámides, propagando sus valores, dejando la estructura lista para el cálculo del siguiente nivel.

Al finalizar del algoritmo los resultados están disponibles en la parte inferior de las pirámides, si bien resta codificados y normalizados para poder ser visualizados.

4.5 Resultados

4.5.1 Comparación entre CPU y GPU

La siguiente tabla recoge los tiempos de ejecución en milisegundos de las distintas etapas de la aplicación para un tamaño de imagen fijo 640x480 pixels y con 50 iteraciones de cálculo de Optical Flow en cada nivel de pirámide.

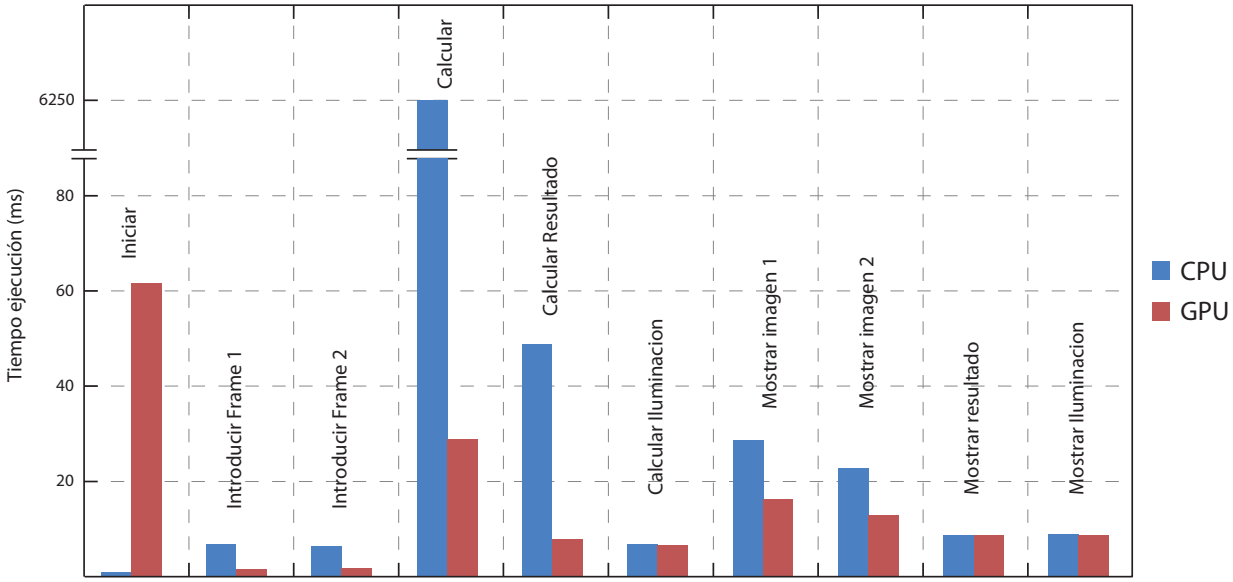
	Individual		Secuencial	
Concepto	**CPU**	**GPU**	**CPU**	**GPU**
Iniciar Optical Flow	0,202	62,371	-	-
Introducir Frame 1	6,972	1,528	6,972	1,528
Introducir Frame 2	6,447	1,766	6,447	1,766
Calculo recibido	6250,590	29,267	6250,590	29,267
Resultado recibido	49,459	7,957	62,745	8,557
Iluminación recibida	6,826	6,624	20,287	6,624
Mostrar imagen 1	28,763	16,229	-	-
Mostrar imagen 2	22,820	12,878	-	-
Mostrar resultado	8,768	8,737	-	-
Mostrar iluminación	8,968	8,884	-	-
Total	**6389,613**	**93,87**	**6347,041**	**47,742**
Relación CPU/GPU		**68,07**		**132,94**

Tabla 2. Comparación tiempos (ms) entre CPU y GPU

Se observa que la implementación en GPU es 68,07 veces más rápida para el cálculo entre dos imágenes individuales, incluyendo etapas de inicialización.

En caso de ejecución continua para una secuencia de imágenes, el cálculo resulta 132,94 veces más rápido. En particular, la etapa específica de cálculo del Optical Flow resulta 213,57 veces más rápida en la implementación realizada en GPU.

Estos valores se muestran en la siguiente tabla. Se hace notar que la columna de la fase de cálculo en CPU, muy superior a las demás, no se representa a escala.

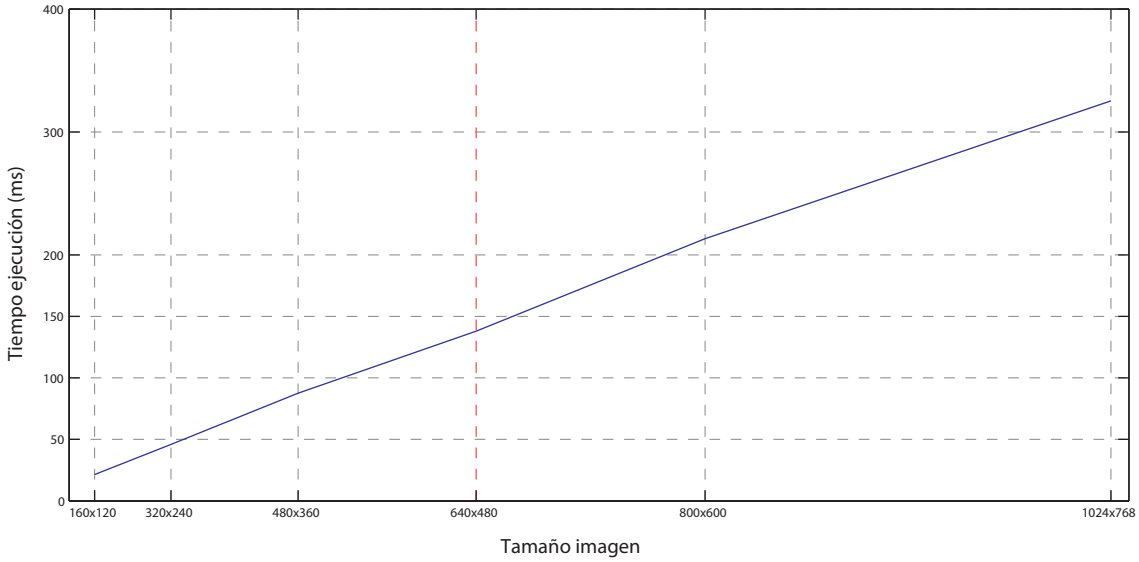


Gráfica 7. Comparación tiempos (ms) entre CPU y GPU

4.5.2 Influencia de los parámetros

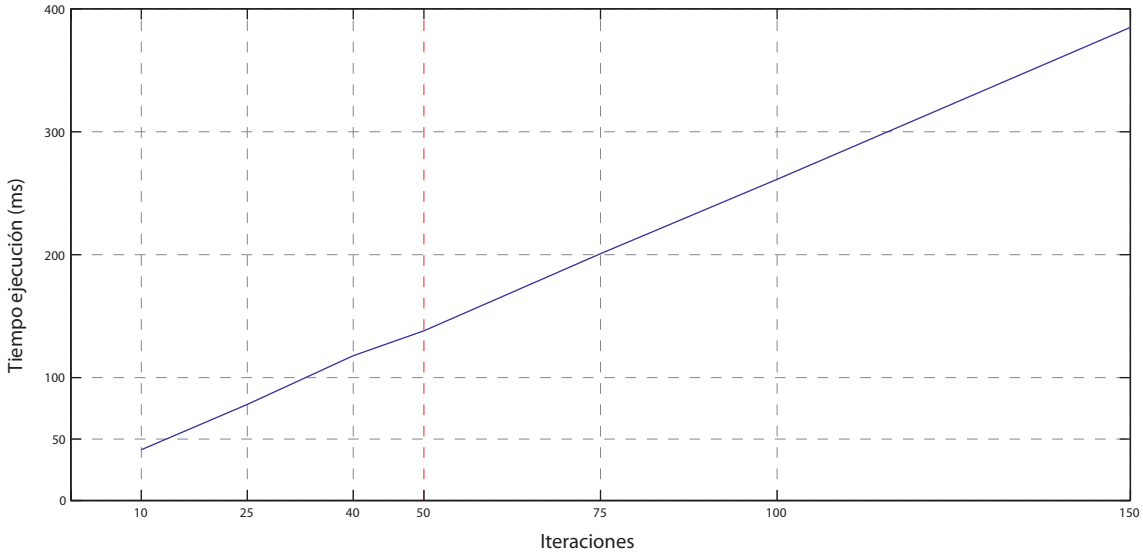
A continuación se diseña una serie de experimentos para caracterizar la influencia de los distintos parámetros que intervienen en el algoritmo en el tiempo de ejecución global del mismo. Los tiempos representados muestran el proceso completo, incluyendo captura de imágenes, cálculo Optical Flow, y visualización de resultados.

En primer lugar se determina la relación con el tamaño de la imagen, medida en número de pixels. Se observa una relación fuertemente lineal entre ambas variables.



Gráfica 8.Relación entre tiempo (ms) y tamaño de imagen.

A continuación se determina la relación entre el número de iteraciones realizada por nivel de pirámide. Nuevamente se observa una relación fuertemente lineal entre ambas variables.



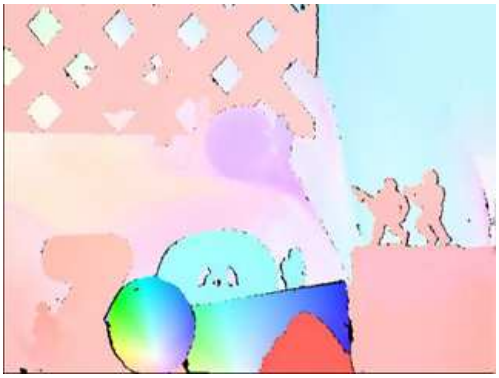
Gráfica 9.Relación entre tiempo (ms) e iteraciones.

4.6 Comparación con Ground Truth

Para determinar la eficacia en la estimación de movimiento mediante el algoritmo realizado, se dispone de una serie de muestras que incluyen ambas imágenes de entrada y la solución exacta de la estimación de movimiento, que denominaremos Ground Truth. A continuación se muestra las imágenes y resultados obtenidos para una de las muestras empleadas,

Frame 1

Frame 2



Ground Truth

Resultado con Lambda 5

Resultado con Lambda 20

Resultado con Lambda 40

Imagen 18. Comparación resultados con Grounded Image

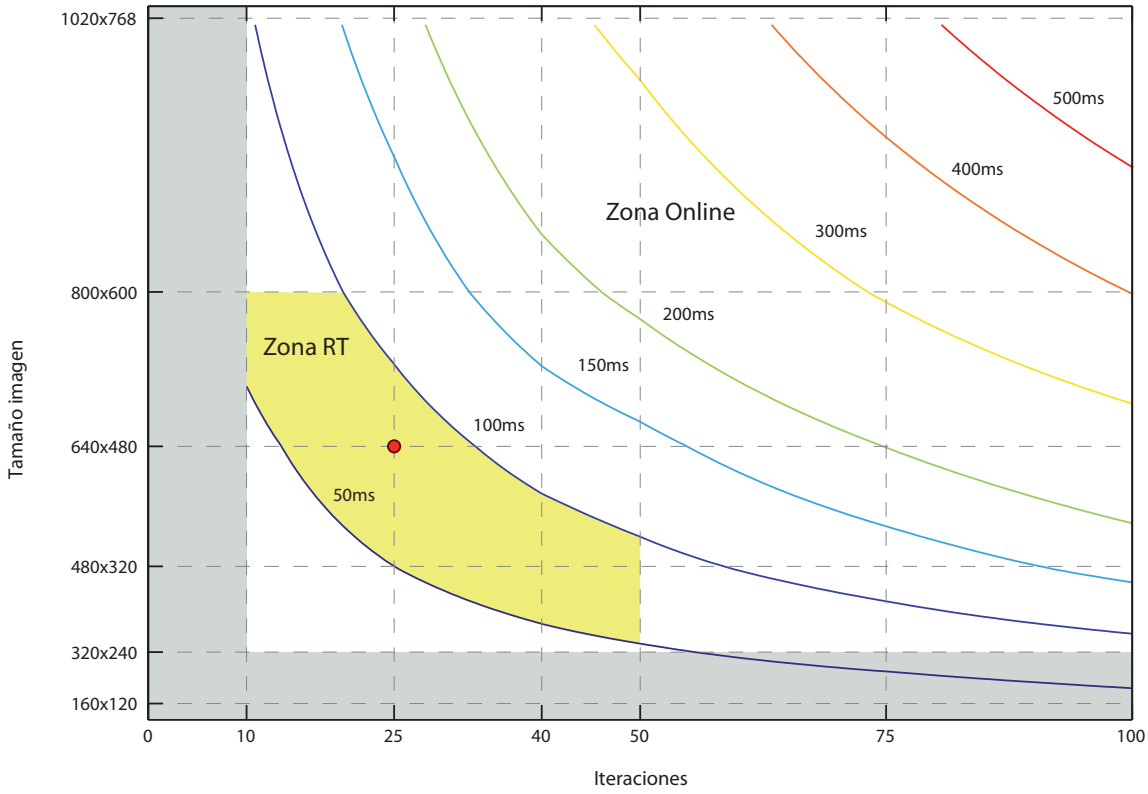
Se observa que con lambda 15 a 25 se obtienen resultados altamente ajustados a la solución Ground Truth proporcionada. Las pequeñas diferencias restantes resultan de limitaciones propias del algoritmo, y no de la implementación realizada. Por tanto no pueden ser eliminadas con los métodos empleados en el presente proyecto.

4.6.1 Zona tiempo real

Finalmente, se desea caracterizar la relación entre tiempo, tamaño de imagen e iteraciones, para determinar una zona dentro de la cual la se consigue la ejecución en tiempo real. A estos efectos, se considera tiempo real a velocidades de refresco de 10-20 frames por segundo, lo que implica tiempos de ejecución de 50 a 100 ms por frame.

Para ello se diseña y ejecuta una serie muti variable de experimentos. Se ha optado por este método, en lugar de la interpolación a partir de los resultados previos, para eliminar la posible interferencia entre los efectos de ambos parámetros.

Los resultados obtenidos se muestran en la siguiente tabla, donde se ha sombreado en amarillo la zona de tiempo real.



Gráfica 10. Zona tiempo real Optical Flow

Se comprueba que con menos de 10 iteraciones el método presenta problemas importantes de convergencia. Por otro lado, la ejecución de más 50 iteraciones resulta innecesaria dado que no se observa una mejora sustancial de la solución obtenida.

Adicionalmente se observa la obtención de mejores resultados fijando un número de iteraciones relativamente bajo, pero que permita conseguir elevadas tasa de refresco. Esto permite la aparición de menores desplazamientos entre cuadros lo que favorece la convergencia, más que aumentar el número de iteraciones.

Con estos datos, la implementación realizada y el equipo empleado, se considera que el punto óptimo es 640 x 480 a 15 fps, con 25 iteraciones por nivel.

La ejecución en un equipo con una tarjeta superior o una implementación con un número menor de visualizadores aumentaría aún más la velocidad de ejecución.

5 Conclusiones

Durante el desarrollo del presente proyecto se han conseguido los siguientes objetivos y resultados:

- Se ha realizado la implementación de Denoise, en Matlab, en CPU y en GPU.
- Se ha realizado la implementación del modelo TV-ROF y TV-L1 de Denoise, tanto en CPU como en GPU.
- Se ha realizado la implementación del Optical Flow tanto en CPU como en GPU.
- Se ha optimizado ambas aplicaciones en GPU hasta conseguir la ejecución en tiempo real mediante imágenes con una webcam.
- Para ambas aplicaciones se ha caracterizado la relación entre tiempo y tamaño de la imagen, y tiempo entre número de iteraciones.
- Para ambas aplicaciones se ha determinado la eficacia de los métodos empleados. En el caso de Denoise mediante el cálculo del SNR, y en el caso del Optical Flow mediante comparación con la solución Ground Truth.
- Para ambas aplicaciones se ha obtenido una región de RT, como representación de la relación entre ambos parámetros para que la ejecución se realice en tiempo real.
- Se han generado las herramientas y clases suficientes que permitan la reutilización del trabajo realizado en futuros proyectos y aplicaciones.

Por tanto, se consideran correctamente conseguidos la totalidad de objetivos que motivan el presente proyecto.

5.1 Trabajos futuros

Movidos por el éxito en la consecución de los objetivos se considera interesante presentar los siguientes trabajos como ejemplos de futuras aplicaciones o ampliaciones que resultarían interesantes de realizar en próximos proyectos:

- Extender los algoritmos presentados para el tratamiento de imágenes en color.
- Modificar las aplicaciones para su ejecución remota, permitiendo adquirir las imágenes con un dispositivo móvil, mientras que el cálculo se realiza en un equipo independiente de elevada potencia, tal como el cluster HERMES del I3A.
- Combinar los algoritmos presentados con un sistema detección de blobs que permita reducir o eliminar el problema de falta de textura.
- Extender el algoritmo para Scene Flow [AW11], que consiste en obtener la velocidad de cada punto 3D de la escena mediante visión estereoscópica.
- Emplear las clases implementadas en una aplicación práctica, tal como un videojuego, el sistema de posicionamiento de un robot, o un sistema de detección de obstáculos para vehículos en movimiento.

APÉNDICE A.
DESARROLLO MATEMÁTICO

1 Planteamiento del método

1.1 Caso general

En un caso general, la aplicación de los métodos variacionales consisten en la resolución de funciones de la familia

$$\min_u \left\{ \int_\Omega F(\nabla u) + \lambda \int_\Omega G(u) \right\} \tag{1}$$

Donde el término regularizador $F(\nabla u)$, impone condiciones de suavidad en la imagen, mientras que el término de fidelidad $G(u)$, o data term, ajusta la solución obtenida a las condiciones de entrada. La transición entre ambos comportamientos se controla mediante el parámetro lambda. El método presenta dentro un ampli rango de problemas típicos en vision por computador. La diferencia entre su aplicación a uno u otro caso depende de la forma adoptada por las funciones F y G que deben ser particularizadas para cada aplicación en concreto.

1.2 Modelo Tikhonov

Con objeto de simplificar la presentación del método general, y por sencillez, se ilustrará el modelo particularizado para el caso de filtrado de ruido. En su formulación más sencilla, aplicando la regularización de Tikhonov, se obtiene la siguiente expresión.

$$\min_u \left\{ \int_\Omega (\nabla u)^2 d\Omega + \lambda \int_\Omega (f-u)^2 d\Omega \right\}) \tag{2}$$

Denotaremos a esta ecuación como el modelo Tikhonov, donde el término regularizador tiene la forma $(\nabla u)^2$ y el término fidelidad resulta $(u-f)^2$.

1.3 Modelo TV-ROF

Los metodos de variación total en visión presentan una mejora del método anterior. Su formulación general resulta de la aplicación como regularizador de la variación total de la señal o imagen TV(u) definida por

$$TV(u) = \int_\Omega |\nabla u| d\Omega \tag{3}$$

Los primeros en introducir los métodos de variación total a los problemas de visión fueron Rudi, Osher y Fatemi (ROF) en su artículo sobre la eliminación de ruido preservando los contornos [LO92]. El modelo es capaz de eliminar ruido y otros detalles de pequeña escala indeseados, mientras que preserva las discontinuidedes pronuciadas, tales comos los contornos.

En su formulación original, el modelo TV-ROF es definido como el problema de optimización con restricciones

$$\min_u \left\{ \int_\Omega |\nabla u| d\Omega \right\} \qquad \text{s.t.} \qquad \int_\Omega (u-f)^2 d\Omega = \sigma^2 \tag{4}$$

Posteriormente Chambolle demostró que el problema original no convexo del modelo TV-ROF podía ser convertido en un problema de minimización convexo reemplazando la igualdad $\int_\Omega (u-f)^2 d\Omega = \sigma^2$ por la desigualdad $\int_\Omega (u-f)^2 d\Omega \leq \sigma^2$, con lo que el modelo original podia ser transformado en el modelo sin restricciones usando el multiplicador de Lagrange lambda.

$$\min_u \left\{ \int_\Omega |\nabla u| + \lambda \int_\Omega (u-f)^2 d\Omega \right\} \tag{5}$$

El interés tras la sustitución de la norma cuadrática en el término regularizador en problemas variacionales en imágenes es permitir la existencia de zonas suaves, a la vez que mantiene las discontinuidades tales como los contornos.

1.4 Modelo TV-L1

De forma similar al modelo TV-ROF, el modelo TV-L1 es definido como el problema variacional

$$\min_u \left\{ \int_\Omega |\nabla u| + \lambda \int_\Omega |u-f| d\Omega \right\} \tag{6}$$

La diferencia con el modelo ROF es que la norma cuadratica L2 del termino de fidelidad es reemplazada por la noma L1. Sin embargo, mientras que el modelo ROF es un problema de minimización estrictamente convexo, el modelo TV-L1 no es estrictamente convexo. Esto significa que, de forma general, no existe un único minimizador global.

Sin embargo, el modelo TV-L1 presenta una mejora importante del comportamiento. Por un lado, es más efectivo que el modelo ROF para la eliminación de ruido impulsional, tal como señales de ruido y pimienta. Por otro lado, es invariante al contraste, una cualidad que resulta útil para la detección de contornos y la eliminación de ruido.

2 Método Primal Dual

Previamente a poder desarrollar de forma adecuada los métodos de resolución empleados en el presente proyecto, se hace necesario recordar ciertos conceptos y definiciones propios de los problemas de minimización de funcionales no convexos.

En general, muchos problemas de estimacion se pueden plantear como la minimización de una cierta función potencial $F : \mathbb{R}^n \to \mathbb{R}$. Cuando F es una funcion convexa es posible encontrar un minimizador global de F y en consecuencia, la solucion óptima del problema en cuestion.

Decimos que un conjunto $U \subset \mathbb{R}^p$ es convexo si $\forall x, y \in U$

$$\lambda \in [0,1], \lambda x + (1-\lambda)y \in U \tag{7}$$

Por otro lado, dado $U \subset \mathbb{R}^p$ y $f : U \to (-\infty, \infty]$ llamamos epigráfica de una función f a

$$epi(f) = \{(x,\mu) : x \in, f(x) \leq \mu\} \subset \mathbb{R}^{p+1} \tag{8}$$

Por último se define una función convexa. Dado $U \subset \mathbb{R}^p$ y $f : U \to (-\infty, \infty]$ una función. Decimos que f es convexa si su epigráfica $epi(f)$ es un conjunto convexo o, de forma equivalente,

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y) \qquad \forall x, y \in X, \forall \alpha \in [0,1] \tag{9}$$

Si el potencial convexo a minimizar es diferenciable, es posible utilizar metodos de optimización basados en descenso de gradiente para encontrar un minimizador global. Sin embargo, las funciones potenciales empleadas en el presente proyecto no son diferenciables. Quisieramos extender el concepto de gradiente a funciones no son diferenciables para posteriormente emplear esta extensión como punto de partida para generalizar los métodos de optimización basados en el cálculo del gradiente de la funcion objetivo.

La propiedad del gradiente que nos va a interesar es la siguiente:

Sea $f : \mathbb{R}^p \to (-\infty, \infty]$ una función convexa diferenciable en $x_0 \in \mathbb{R}^p$. Entonces la función $g(x) = f(x_0) + \nabla f(x_0)^T(x - x_0)$ es un subestimador global de f en $\mathbb{R}^p$. Es decir, $g(x) \leq f(x) \forall x \in \mathbb{R}^p$ (la linea tangente a f en x_0 queda por debajo de f).

La observación importante es que cuando f no es diferenciable, existen muchas funciones afines g que son subestimadores globales de f. En ese caso, vamos a considerar todas las funciones afines que cumplen dicha condición.

Una propiedad interesante de las funciones convexas es que si h es una función afín que subestima globlamente a f, entonces la epigráfica de f está completamente contenida en la epigráfica de h. Entonces tenemos el siguiente teorema.

Sea $f : \mathbb{R}^p \to (-\infty, \infty]$ una función convexa. Sea F^* el conjunto de todas las funciones afines que subestiman globalmente a f. Entonces la epigráfica de la función f es la intersección de las epigráficas de todas las funciones afines en F^*

$$epi(f) = \bigcap_{h \in F^*} epi(h) \tag{10}$$

La forma intuitiva de interpretar este problema es considerar una nueva forma de considerar una función. De forma habitual, consideramos una función f como el conjunto de puntos $(x, f(x))$, lo que hacemos de forma habitual. EL teorema nos indica que podemos pensar en f como un conjunto de hiperplanos que envuelven y definen a f a través de su epigráfica.

Profundizando en este concepto, consideremos una función afín $h : \mathbb{R}^p \to \mathbb{R}$ caracterizada por el par (p, μ) donde $p \in \mathbb{R}^p$ y $\mu \in \mathbb{R}$, la función afín puede escribirse como $h(x) =< x, p > -\mu$ Entonces, en virtud del teorema anterior

$$\begin{aligned} F^* &= \{(p, \mu) \in \mathbb{R}^{p+1} : \mu \geq f^*(p)\} = epi(f^*) \\ F^* &= \{(p, \mu) \in \mathbb{R}^{p+1} : \mu \geq sup < p, x > -f(x)) \end{aligned} \tag{11}$$

Por tanto, definimos la conjugada convexa de una función $f : \mathbb{R}^p \to (-\infty, \infty]$ como la función $f^* : \mathbb{R}^p \to (-\infty, \infty]$ tal que

$$f^*(p) = \sup_{x \in \mathbb{R}^p} < b, x > -f(x) \tag{12}$$

A su vez, definimos la función indicadora $C \subset \mathbb{R}^p$ un conjunto. Definimos la función indicadora de C como la función $\delta_C : \mathbb{R}^p \to (-\infty, \infty]$ dada por

$$\delta_C(w) = \begin{cases} 0 & \text{si } w \in C \\ +\infty & \text{si } w \notin C \end{cases}$$

Por otro lado, definimos la norma dual. Sea $\Omega : \mathbb{R}^p \to (-\infty, \infty]$ una norma. Definimos la norma dual asociada a Ω^* como de ω, entonces

$$\Omega^*(z) = \max_{w \in \mathbb{R}^p : \Omega(w) \leq 1} < z, w > \tag{13}$$

Finalmente plantearemos el la Dualidad del Problema. Sea f^* y Ω^* respectivamente la conjugada convexa de f y la norma dual de Ω, entonces

$$\max_{z \in \mathbb{R}^p : \Omega^*(z) \leq \lambda} -f^* \leq \min_{w \in \mathbb{R}^p} f(w) + \lambda \Omega(w) \tag{14}$$

Aplicando los resultados anteriores anteriores a la norma TV, la formulación dual queda expresada de la siguiente forma

$$|\nabla u| = max \{p \cdot \nabla u : ||p|| \leq 1\} \tag{15}$$

3 Discretización

Nuestro objetivo es la obtención de métodos matemáticos que puedan ser implementados en un ordenador [TA10]. Por tanto, resulta necesario discretizar las ecuaciones de cada método. En primer lugar, se define una imagen digital como una rejilla rectangular cartesiana de dos dimensiones, definida como

$$\{(x_i, y_j) = (i \cdot \Delta x, j \cdot \Delta y | 1 \leq i \leq m, 1 \leq j \leq n\} \tag{16}$$

donde (x_i, y_j) son las localizaciones discretas en la rejilla, x e y denotan los subintervalos de la rejilla, y m y n, respectivamente, ancho y alto de la imagen.

3.1 Discretización caso general

El caso general planteado puede ser discretizado como

$$\min_{x \in X} F(Kx) + G(x) \tag{17}$$

o el correspondiente problema dual

$$\max_{y \in Y} -(G^*(-K^*y) + F^*(y)) \tag{18}$$

Sea X, Y dos espacios vectorales de dimensión finita dotados de un producto interior $\langle \cdot, \cdot \rangle$ y una norma $|| \cdot || = \langle \cdot, \cdot \rangle^{\frac{1}{2}}$. El operador $K : X \to Y$ es un operador lineal continuo inducido por la norma

$$||K|| = \max \{||Kx|| : x \in X \text{ con } ||x|| \leq 1\}) \tag{19}$$

$$K = \min_{x \in X} \max_{y \in Y} \langle Kx, y \rangle + G(x) + F^*(y) \tag{20}$$

Asumiendo que el problema tiene al menos una solución ($\hat{x}, \hat{y} \in XxY$ que por tanto satisfacen

$$\begin{aligned} K\hat{x} &\in \partial F^*(\hat{y}), \\ -(K^*\hat{y}) &\in \partial G(\hat{x}) \end{aligned} \tag{21}$$

donde δF^* y δG son subgradientes de las funciones convexas F^* y G. En lo siguiente asumiremos que F y G son "sencillas", en el sentido de que puede ser resuelto el operador "proximal map"definido a traves de

$$x = (I + \tau \partial F)^{-1}(y) = \arg\min_x \left\{ \frac{||x - y||^2}{2\tau} + F(x) \right\} \tag{22}$$

$$\left\{ \begin{array}{lcl} y^{n+1} & = & (I + \sigma \partial F^*)^{-1}(y^n + \sigma K \hat{x}^n) \\ x^{n+1} & = & (I + \tau \partial G)^{-1}(y^n + \tau K^* y^{n+1}) \\ \hat{x}^{n+1} & = & x^{n+1} + \theta(x^{n+1} - x^n) \end{array} \right.$$

3.2 Discretización operadores

De igual forma, con caracter previo a la definición de los algoritmos discretos que permiten resolver el modelo ROF y TV-L^1, es necesario definir de forma apropiada los operadores diferenciales,

$$\nabla u = \left(\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}\right)^T, \qquad |\nabla u| = \sqrt{\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2} \tag{23}$$

En su formulación discreta, el operador gradiente está definido como

$$(\nabla u)_{i,j} = (\delta_x^+ u_{i,j}, \delta_y^+ u_{i,j})^T \qquad |\nabla u| = \sqrt{(\delta_x^+ u_{i,j})^2 + (\delta_y^+ u_{i,j})^2} \tag{24}$$

donde las derivadas se aproximan usando diferencias hacia delante

$$\begin{aligned} \delta_x^+ u_{i,j} &= \frac{u_{i+1,j} - u_{i,j}}{\Delta x} \\ \delta_y^+ u_{i,j} &= \frac{u_{i,j+1} - u_{i,j}}{\Delta y} \end{aligned} \tag{25}$$

Por su parte, la divergencia de un campo vectorial $p = (p_1, p_2)^T$

$$\nabla \cdot p = \mathrm{div} p = \frac{\partial p^1}{\partial x} + \frac{\partial p^2}{\partial y} \tag{26}$$

El operador divergencia en su configuración discreta está definido como

$$\mathrm{div} p_{i,j} = \delta_x^- p_{i,j}^1 + \delta_y^- p_{i,j}^2 \tag{27}$$

En su formulación continua, el operador divergencia es adjunto al operador diferencial mediante $-(\nabla p)u = p \cdot \nabla u$. En orden de satisfacer también este requerimiento en su formulación continua se hace necesario aproximar las derivadas del operador divergencia discreto mediante diferencias hacia atras.

$$\begin{aligned} \delta_x^- p_{i,j}^1 &= \frac{p_{i,j}^1 - p_{i-1,j}^1}{\Delta x} \\ \delta_y^- p_{i,j}^2 &= \frac{p_{i,j}^2 - p_{i,j-1}^2}{\Delta y} \end{aligned} \tag{28}$$

Donde, sin pérdida de generalidad, podemos asumis a efectos de simplicidad que $x = y = 1$

Por supuesto, es posible emplear sistemas de discretización más sofisticados, resultando en operadores diferenciales más precisos. No obstante, para los problemas prácticos asociados con la visión por ordenador, las discretizaciones empleadas resultan suficientes.

4 Denoise

4.1 Planteamiento

El planteamiento general para los métodos variacionales aplicados a problemas dentro filtrado de ruido en imagenes, partirmos del planteamiento general establecido por Rudin, Osher y Fatemi en el modelo ROF, definido mediante

$$\min_x \int_\Omega |Du| + \frac{\lambda}{2}||u-g||_2^2 \tag{29}$$

donde $\Omega \subset \mathbb{R}^d$ es el dominio d-dimensional de la imagen, $u \in L^1(\Omega)$ es la solución deseada y $g \in L^1(\Omega)$ es la imagen con ruido introducida. El parámetro λ es usado para determinar el peso entre regularización y fidelidad. El término $\int_\Omega |Du|$ es la variación total de la función u, donde Du denota la diferenciación, correctamente definida para funciones discontinuas. Para funciones suficientemente suaves u, se reduce al consabido término $\int_\Omega |\nabla u| dx$.

4.2 Discretización

Usando la configuración discreta introducida con interioridad, y estableciendo las dimensiones $d = 2$), el modelo discreto ROF, que llamaremos primal ROF, está dado por

$$h^2 \min_{u \in X} ||\nabla u||_1 + \frac{\lambda}{2}||u-g||_2^2 \tag{30}$$

donde $u, g \in X$ son, respectivamente, la solución y la imagen con ruido. La norma $||u||_2^2 = \langle u, u \rangle_X$ denota la norma cuadrática L^2 estandar X y $||\nabla u||_1$ la versión discreta de la norma TV isotrópica definida como

$$||\nabla u||_1 = \sum_{i,j} |(\nabla u)_{i,j}|, \qquad |(\nabla u)_{i,j}| = \sqrt{((\nabla u)_{i,j}^1)^2 + ((\nabla u)_{i,j}^2)^2}$$

Mediante comparación con el caso general resulta inmediato observar que $F(\nabla u) = ||\nabla u||_1$ y $G(u) = \frac{\lambda}{2}||u-g||_2^2$. En adelante obviaremos el término h^2 dado que únicamente afecta un escalado de la energía interno, sin variar la solución.

Por otro lado. formulación primal-dual del modelo ROF está dada por

$$\min_{u \in X} \max_{p \in Y} - \langle u - divp \rangle_X + \frac{\lambda}{2}||u-g||_2^2 - \delta_P(p) \tag{31}$$

donde $p \in Y$ es la variable dual. El conjunto convexo P está dado por

$$P = \{p \in Y : ||p||_\infty \leq 1\} \tag{32}$$

y $||p||_\infty$ denota la norma máxima discreta definida como

$$||p||_\infty = \max_{i,j} |p_{i,j}|, \qquad |p_{i,j}| = \sqrt{(p_{i,j}^1)^2 + (p_{i,j}^2)^2} \tag{33}$$

La función δP indica la función indicadora del conjunto P que está definido como

$$\delta_P(p) = \begin{cases} 0 & \text{si } p \in P \\ +\infty & \text{si } p \notin P \end{cases}$$

Adicionalmente, el problema ROF primal ROF y el problema ROF primal-dual son equivalentes al problema ROF dual

$$\max_{p \in Y} - \left(\frac{1}{2\lambda} + ||\text{div } p||_2^2 + \langle g, \text{div } p \rangle_X + \delta_P(p) \right) \tag{34}$$

Resta por resolver los operadores $(I+\sigma\delta F^\star)^{-1}$ y $(I+\tau\delta G)^{-1}$. Identificando términos con la formulación general se observa que $F^\star(p) = \delta_P(p)$ y $G(u) = \frac{\lambda}{2}||u-g||_2^2$. Dado que $F^\star$ es la función idnicadora del conjunto convexo, el operador se reduce a la proyección Euclidea en bolas L^2

$$p = (I + \sigma\delta F^\star)^{-1}(\hat{p}) \Longleftrightarrow p_{i,j} = \frac{\hat{p}_{i,j}}{\max(1, |\hat{p}_{i,j}|)} \tag{35}$$

La resolución del operador respecto a G está dada por

$$u = (I + \tau\delta G)^{-1}(\hat{u}) \Longleftrightarrow u_{i,j} = \frac{\hat{p}_{i,j} + \tau\lambda g_{i,j}}{1 + \tau\lambda} \tag{36}$$

En el caso del modelo TV-L^1, obtenido mediante una variación del modelo ROF reemplazando la norma cuadrática L^2 en el término de fidelidad por la robusta norma L^1.

$$\min_x \int_\Omega |Du| + \frac{\lambda}{2}||u-g||_1 \tag{37}$$

La versión discreta está dada por

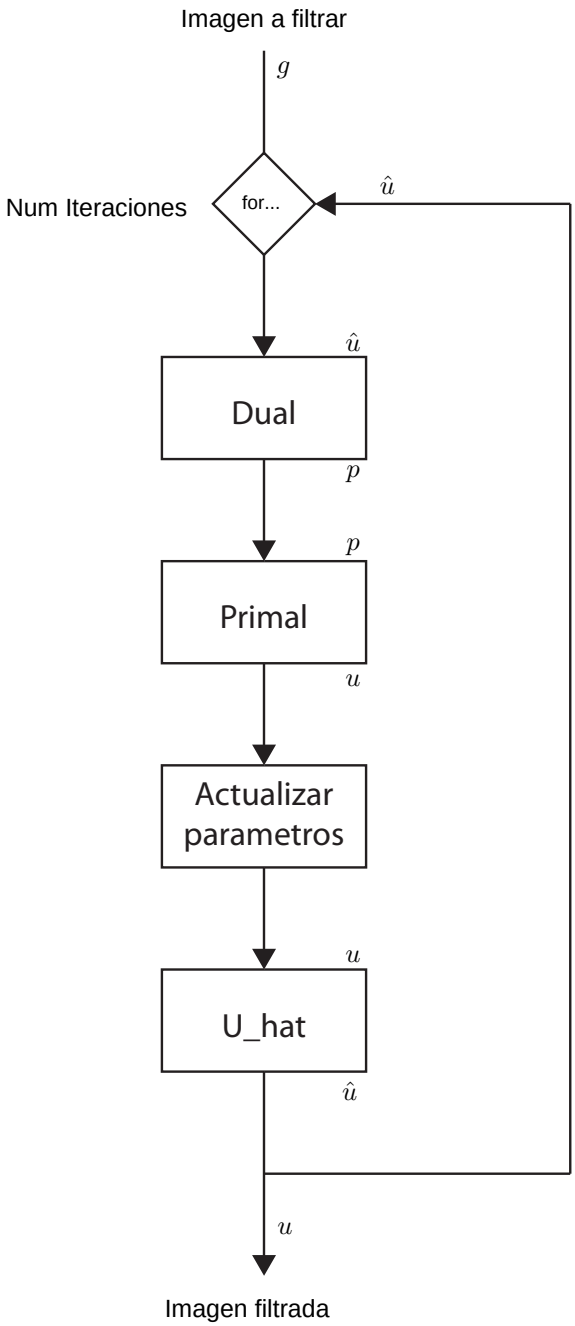
$$\min_{u \in X} ||\nabla u||_1 + \lambda||u-g||_1 \tag{38}$$

Donde la resolución de respecto a p es idéntica al caso ROF, y la resolución del operador respecto a G viene dada por

$$u = (I + \tau\delta G)^{-1}(\hat{u}) \Longleftrightarrow u_{i,j} = \begin{cases} \hat{u}_{i,j} - \tau\lambda & \text{si } \hat{u}_{i,j} - \tau\lambda > \tau\lambda \\ \hat{u}_{i,j} + \tau\lambda & \text{si } \hat{u}_{i,j} - \tau\lambda < -\tau\lambda \\ g_{i,j} & \text{si } |\hat{u}_{i,j} - \tau\lambda| \leq \tau\lambda \end{cases}$$

4.3 Esquema algoritmo

El siguiente esquema muestra el diagrama de flujo para la Clase Denoise, donde cada una de las etapas se detallan posteriormente.



4.4 Ecuaciones algoritmo

A continuación se recogen las ecuaciones que integran cada una de las etapas de la Clase Denoise

Update Dual

Calculo de la variable Dual p

$$p = p + \sigma \cdot \nabla \hat{u} \tag{39}$$

Normalización aplicando la restricción $|p| \leq 1$

$$\begin{aligned} p_{norm} &= \sqrt{p_x^2 + p_y^2} \\ p &= p/max(1, p_{norm}) \end{aligned} \tag{40}$$

Update Primal

Calculo de la variable Primal $\hat{u}$

$$u_{ant} = u \tag{41}$$

$$\hat{u} = u + \tau \nabla \cdot p \tag{42}$$

En el caso de aplicar el modelo TV-ROF aplicar la ecuación

$$u = \frac{\hat{u} + \tau \cdot \lambda \cdot g}{1 + \tau \cdot \lambda} \tag{43}$$

Para el modelo TV-L1 la resolución de respecto a p sustituir la expresión anterior por

$$u = \begin{cases} \hat{u} - \tau\lambda & \text{si } \hat{u} - \tau\lambda \quad > \tau\lambda \\ \hat{u} + \tau\lambda & \text{si } \hat{u} - \tau\lambda \quad < -\tau\lambda \\ g & \text{si } |\hat{u} - \tau\lambda| \quad \leq \tau\lambda \end{cases}$$

Update parameters

Actualización de los parámetros para siguiente iteración

$$\theta = \frac{1}{\sqrt{1 + 2\gamma\tau}} \tag{44}$$

$$\tau = \tau \cdot \theta \tag{45}$$

$$\sigma = \sigma / \theta \tag{46}$$

Calculate Uhat

Calculo de $\hat{u}$

$$\hat{u} = u + \theta \cdot (u - u_{ant}) \tag{47}$$

4.5 Ecuaciones algoritmo discretizadas

La discretización detallada de las ecuaciones que integran la formulación de las distintas etapas de la Clase Denoise son las siguientes:

Update Dual

Calculo de la variable Dual $p_{i,j}$

$$\begin{aligned} p_{x(i,j)} &= p_{x(i,j)} + \sigma(u_{i+1,j} - u_{i,j}) \\ p_{y(i,j)} &= p_{y(i,j)} + \sigma(u_{i,j+1} - u_{i,j}) \end{aligned} \tag{48}$$

Normalización aplicando la restricción $p_{i,j} \leq 1$

$$\begin{aligned} p_{norm(i,j)} &= \sqrt{p_{x(i,j)}^2 + p_{y(i,j)}^2} \\ p(i,j) &= p(i,j)/max(1, p_{norm(i,j)}) \end{aligned} \tag{49}$$

Update Primal

Calculo de la variable Primal $u_{(i,j)}$

$$\begin{aligned} u_{ant(i,j)} &= u_{(i,j)} \\ \hat{u}_{(i,j)} &= u_{(i,j)} + \tau[p_{(i,j)} - p_{(i-1,j)} + p_{(i,j)} - p_{(i,j-1)}] \end{aligned} \tag{50}$$

En el caso de aplicar el modelo TV-ROF aplicar la ecuación

$$u_{(i,j)} = \frac{\hat{u}_{(i,j)} + \tau \cdot \lambda \cdot g_{(i,j)}}{1 + \tau \cdot \lambda} \tag{51}$$

Donde la resolución de respecto la P es idéntica al caso ROF, y la resolución del operador respecto a G viene dada por

$$u_{i,j} = \begin{cases} \hat{u}_{i,j} - \tau\lambda & \text{si } \hat{u}_{i,j} - \tau\lambda \quad > \tau\lambda \\ \hat{u}_{i,j} + \tau\lambda & \text{si } \hat{u}_{i,j} - \tau\lambda \quad < -\tau\lambda \\ g_{i,j} & \text{si } |\hat{u}_{i,j} - \tau\lambda| \quad \leq \tau\lambda \end{cases}$$

Update parameters

Actualización de los parámetros para siguiente iteración

$$\begin{aligned} \theta &= \tfrac{1}{\sqrt{1+2\gamma\tau}} \\ \tau &= \tau \cdot \theta \\ \sigma &= \sigma/\theta \end{aligned} \tag{52}$$

Calculate Uhat

Calculo de $\hat{u}_{(i,j)}$

$$\hat{u}_{(i,j)} = u_{(i,j)} + \theta \cdot \left(u_{(i,j)} - u_{ant(i,j)}\right) \tag{53}$$

5 Optical Flow

5.1 Planteamiento

El planteamiento general para los métodos variacionales aplicados a la estimación de movimiento está dando por

$$\min_{v} \int_{\Omega} |Dv| + \lambda ||\rho(v)|| \tag{54}$$

donde $v = (v_1, v_2)^T : \Omega \to \mathbb{R}^2$ es el campo vectorial que recoge las componentes estimadas de velocidad, y $\rho(v)$ es la restricción tradicional del Optical Flow (OFC). El parámetro λ es nuevamente empelada para definir la transición entre regularización y fidelidad. Para obtener la expresión de la restrición OFC consideremos en primer lugar la imagen como una función que codifica la intensidad de cada pixel de la imagen como una función del espacio y el tiempo.

$$Imagen = I(x(t), t) \tag{55}$$

La restricción tradicional del Optical Flow queda dado por la siguiente expresión,

$$\frac{d}{dt} I(x(t), t) = 0 \tag{56}$$

Esta restricción es obtenida asumiento que las intensidades de los pixeles permanenecen invariantes en el tiempo. Realizando la expansión del término diferencial de la restrición OFC resulta

$$\frac{d}{dt} I(x(t), t) = \frac{dI}{dX_u} \frac{dX_u}{dt} + \frac{dI}{dX_v} \frac{dX_v}{dt} \tag{57}$$

Definiendo los siguientes operadores

$$\nabla I = \left[\frac{dI}{dX_u}, \frac{dI}{dX_u} \right]^T \tag{58}$$

$$\left[\frac{dX_u}{t}, \frac{dX_u}{t} \right] = (v - v_0) \tag{59}$$

La restricción OFC puede ser reescrita mediante la siguiente expresión

$$I_t + \nabla I (v - v_0) = 0 \tag{60}$$

Por lo que el término de fidelidad buscado resulta

$$\rho(v) = I_t + (\nabla I)^T (v - v_0) \tag{61}$$

donde $v = (v_1, v_2)^T : \Omega \to \mathbb{R}^2$ es el campo vectorial que recoge las componentes estimadas de velocidad, I_t es la variación temporal de la secuencia de la secuencia de imágenes, ∇I es el gradiente espacial de la imagen y v_0 es una condición inicial dada.

En situaciones prácticas, no obstante, es altamente improbable debido a los cambios en la iluminación que los intensidades de los pixels permanezcan constantes en el tiempo. Esto motiva la siguiente mejora en la restricción OFC, que de forma explícita modeliza la variación de la iluminación en términos de una función aditiva u

$$\rho(u, v) = I_t + (\nabla I)^T (v - v_0) + \beta u \tag{62}$$

La función $u : \Omega \to \mathbb{R}$ es presumiblemente suave, y por tanto puede ser también regularizada mediante métodos de variación total. El parámetro β controla la influencia del término de iluminación. El modelo de estimación de movimiento mejorado está dado por

$$\min_v \int_\Omega |Du| + \int_\Omega |Dv| + \lambda ||\rho(u, v)|| \tag{63}$$

No obstante, la restricción OFC es únicamente válida para pequeños desplazamientos $v - v_0$. En orden de extender el método para grandes desplazamientos, todo el método debe ser integrado dentro de una aproximación tipo çoarse-fine", que re-estime iterativamente las condiciones iniciales v_0 entre los distintos niveles que integren el algoritmo.

5.2 Discretización

Tras la discretización se obteniene la siguiente formulación dual para el modelo de estimación de movimiento

$$\min_{u \in X, v \in Y} ||\nabla u||_1 + ||\nabla v||_1 + \lambda ||\rho(u, v)||_1 \tag{64}$$

donde la versión discreta de la restricción OFC mejorada está dada por

$$\rho(u_{i,j}, v_{i,j}) = (I_t)_{i,j} + (\nabla I)^T_{i,j} (v_{i,j} - v_{0(i,j)}) + \beta u_{i,j} \tag{65}$$

El gradiente vectorial $(\nabla v) = (\nabla v_1, \nabla v_2)$ existe en el espacio $Z = Y \times Y$ equipado con el producto escalar

$$\langle q, r \rangle_Z = \sum_{i,j} q^1_{i,j} r^1_{i,j} + q^2_{i,j} r^2_{i,j} + q^3_{i,j} r^3_{i,j} + q^4_{i,j} r^4_{i,j} \tag{66}$$

$$q = (q^1, q^2, q^3, q^4) \in Z \tag{67}$$

$$r = (r^1, r^2, r^3, r^4) \in Z \tag{68}$$

y la norma

$$||\nabla v||_1 = \sum_{i,j} |\nabla v_{i,j}| \tag{69}$$

$$|\nabla v_{i,j}| = \sqrt{((\nabla v_1)^1_{i,j})^2 + ((\nabla v_1)^2_{i,j})^2 + ((\nabla v_2)^1_{i,j})^2 + ((\nabla v_2)^2_{i,j})^2} \tag{70}$$

La formulación del problema primal se obtiene como

$$\min_{u\in X, v\in Y} \max_{p\in Y, q\in Z} \langle \nabla u, p\rangle_Y + \langle \nabla v, q\rangle_Z + \lambda||\rho(u,v)|| - \delta_P(p) - \delta_Q(q) \qquad (71)$$

donde los conjuntos convexos P y Q están definido como

$$P = \{p \in Y : ||p||_\infty \leq 1\}$$

$$Q = \{q \in Z : ||q||_\infty \leq 1\}$$

y $||q||_\infty$ es la norma máxima discreta definida en Z como

$$||q||_\infty = \max_{i,j} |q_{i,j}, |q_{i,j}| = \sqrt{(q^1_{i,j})^2 + (q^2_{i,j})^2 + (q^3_{i,j})^2 + (q^4_{i,j})^2}$$

Identificando términos con el caso general resulta $G(u,v) = \lambda||\rho(u,v)||_1$ y $F^\star(p,q) = \delta_P(p) + \delta_Q(q)$.

La resolución del operador con respecto a $F^\star(p,q)$ está nuevamente dado por la proyección simple dentro de bolas L^2.

$$(p,q) = (I + \sigma\delta F^\star)^{-1}(\hat{p},\hat{q})$$

$$p_{i,j} = \frac{\hat{p}_{i,j}}{\text{máx}(1, |\hat{p}_{i,j}|)} \qquad\qquad q_{i,j} = \frac{\hat{q}_{i,j}}{\text{máx}(1, |\hat{q}_{i,j}|)}$$

La resolución para $G(u,v)$ está dada por

$$(I + \tau\partial G)^{-1}(\hat{u},\hat{v}) = \lambda|\rho(u,\theta)| \qquad (72)$$

$$arg\min_{u,v} = \frac{\|u-\hat{u}\|^2}{2\tau} + \frac{\|v-\hat{v}\|^2}{2\tau} + G(u,v) \qquad (73)$$

Resulta conveniente definir $a = [\nabla I, \beta]^T$ y $b = [v - v_0, u]^T$, con lo que la restricción OFC puede ser reescrita en la forma más compacta

$$\rho(u,v) = I_t + \nabla I^T(v - v_0) + \beta u \Longrightarrow \rho(b) = I_t + a^T b \qquad (74)$$

Por lo que la resolución del operador $G(u,v)$ puede ser expresado mediante

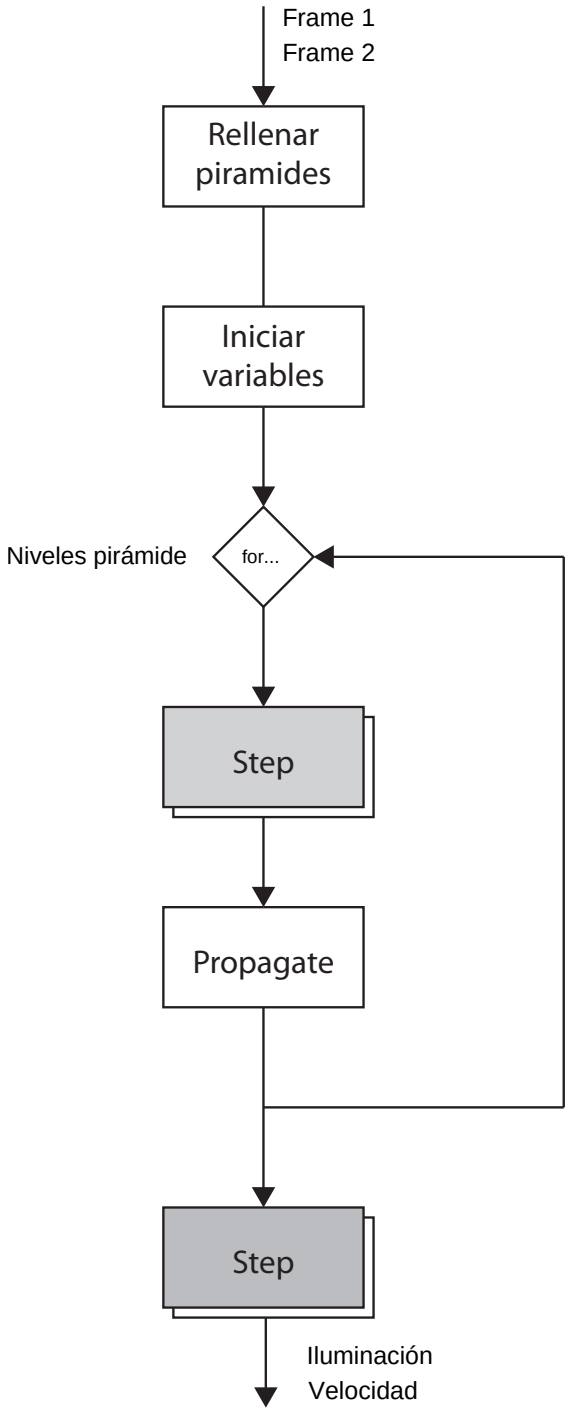
$$arg\min_{b} = \frac{\|b-\hat{b}\|^2}{2\tau} + \lambda|\rho(b)| \qquad (75)$$

Considerando la versión discreta dada por $a_{i,j} = (\beta, (\nabla I)_{i,j})$ y $|a|^2_{i,j} = \beta^2 + |\nabla I|^2_{i,j}$, la solución al operador está entonces definida por

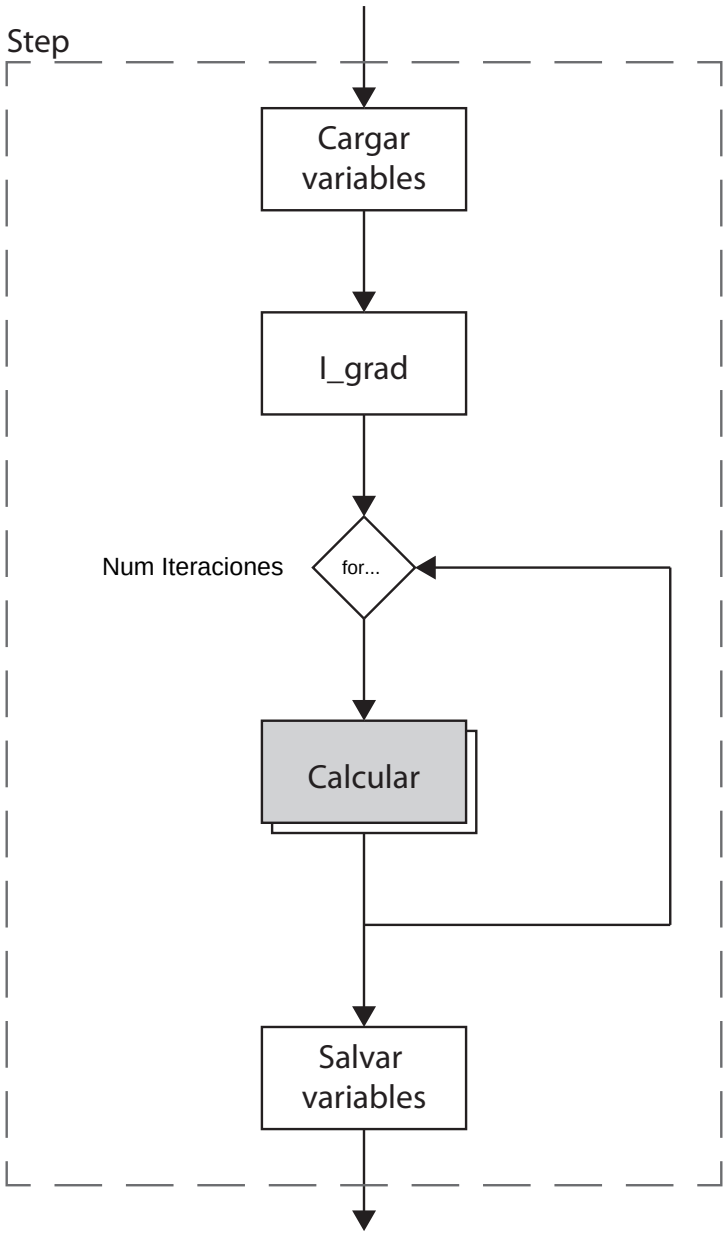
$$(u_{i,j}, v_{i,j}) = (\hat{u}_{i,j}, \hat{v}_{i,j}) + \begin{cases} \tau\lambda a_{i,j} & \text{si } \rho(\hat{u}_{i,j}, \hat{v}_{i,j}) < -\tau\lambda|a|^2_{i,j} \\ -\tau\lambda a_{i,j} & \text{si } \rho(\hat{u}_{i,j}, \hat{v}_{i,j}) > -\tau\lambda|a|^2_{i,j} \\ -\rho(\hat{u}_{i,j}, \hat{v}_{i,j})a_{i,j}/|a|^2_{i,j} & \text{si } |\rho(\hat{u}_{i,j}, \hat{v}_{i,j})| \leq -\tau\lambda|a|^2_{i,j} \end{cases}$$

5.3 Esquema algoritmo

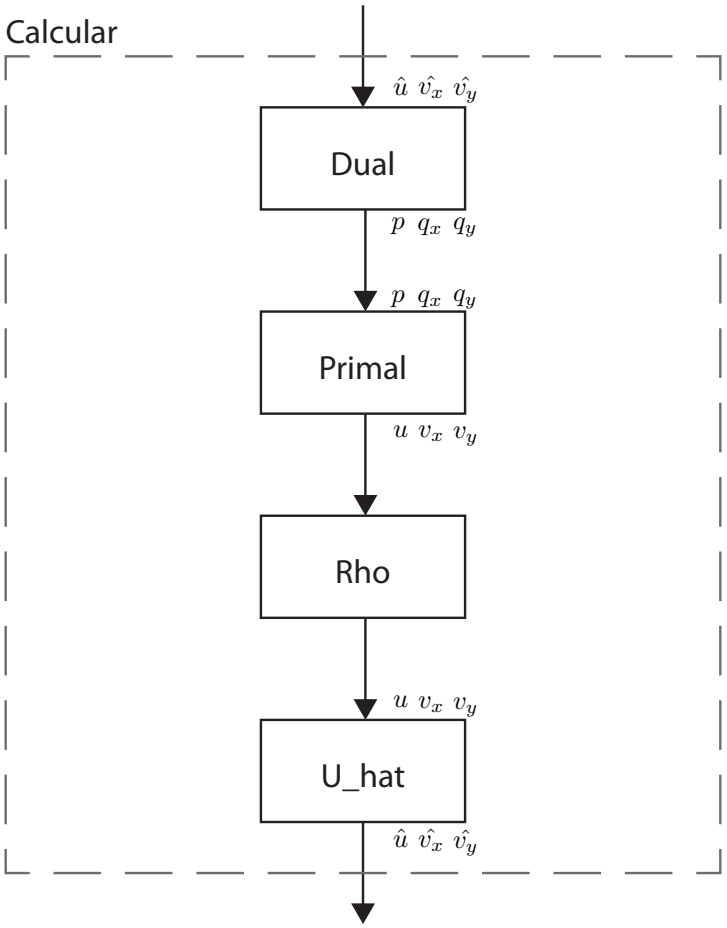
Los siguientes esquemas muestran el diagrama de flujo de la Clase Optical Flow. En primer lugar se rellenan las pirámides que contienen las imágenes de entrada y las soluciones del algoritmo y se inicializan las variables necesarias para realizar el cálculo. A continuación se implementa la aproximación çoarse to fine"del algoritmo. Para todos los niveles de las pirámides, empezando por la parte superior, se realiza el Optical Flow y se propagan las soluciones para el nivel siguiente. Al final se realiza el último paso, dejando las soluciones en la parte inferior de los objetos Pirámide.



La etapa Step se detalla en el siguiente esquema. En cada paso de cálculo se cargan las variables desde los distintos objetos Pirámide que contiene el objeto Optical Flow. A continuación se realiza el proceso de cálculo de I_{grad} que realiza el cálculo de I_t, I_x, I_y, variables necesarias para el cálculo de rho. Estas variables se obtienen mediante la diferencia del frame 2, y el frame 1 deformado según el campo de velocidad en cada paso del Optical Flow (proceso de wrapping), obtenido mediante interpolación lineal. Posteriormente se realiza el cálculo del paso, aplicando el número de iteraciones definidas para cada nivel del Optical Flow. Por último se salvan las variables en las pirámides.



Finalmente, la etapa de cálculo se detalla en el siguiente esquema.



5.4 Ecuaciones algoritmo

A continuación se recogen las ecuaciones que integran cada una de las etapas de la Clase Optical Flow:

Update Dual

Cálculo de las variables Duales p, q_x, q_y

$$\begin{aligned} p &= p + \sigma \cdot \nabla \hat{u} \\ q_x &= q_x + \sigma \cdot \nabla \hat{v}_x \\ q_y &= q_y + \sigma \cdot \nabla \hat{v}_y \end{aligned} \tag{76}$$

Normalización aplicando la restricción $|p| \leq 1$

$$\begin{aligned} p_{norm} &= \sqrt{p_x^2 + p_y^2} \\ p &= p/max(1, p_{norm}) \end{aligned} \tag{77}$$

Normalización aplicando la restricción $|q| \leq 1$

$$\begin{aligned} q_x &= q_x + \sigma \cdot \nabla \hat{v}_x \\ q_y &= q_y + \sigma \cdot \nabla \hat{v}_y \end{aligned} \tag{78}$$

$$\begin{aligned} q_{norm} &= \sqrt{q_{xx}^2 + q_{xy}^2 + q_{yx}^2 + q_{yy}^2} \\ q &= q./max(1, q_{norm}) \end{aligned} \tag{79}$$

Update Primal

Cálculo de las variables Primales $\hat{u}$, $\hat{v_x}$, $\hat{v_y}$

$$\begin{aligned} u_{ant} &= u \\ v_{x_{ant}} &= v_x \\ v_{y_{ant}} &= v_y \end{aligned} \tag{80}$$

$$\begin{aligned} \hat{u} &= u + \tau \nabla \cdot p \\ \hat{v}_x &= v_x + \tau \nabla \cdot q_x \\ \hat{v}_y &= v_y + \tau \nabla \cdot q_y \end{aligned} \tag{81}$$

Update Rho

Cálculo de $\rho(u, v)$

$$\rho(\hat{u}, \hat{v}) = I_t + \nabla I^T (\hat{v} - v_0) + \beta \hat{u} \tag{82}$$

$$|a|^2 = \beta^2 + |\nabla I|^2 \tag{83}$$

$$(u,v) = (\hat{u}, \hat{v}) + \begin{cases} \tau\lambda a & \text{si } \rho(\hat{u}, \hat{v}) < -\tau\lambda|a|^2 \\ -\tau\lambda a & \text{si } \rho(\hat{u}, \hat{v}) > -\tau\lambda|a|^2 \\ -\rho(\hat{u}, \hat{v})a/|a|^2 & \text{si } |\rho(\hat{u}, \hat{v})| \leq -\tau\lambda|a|^2 \end{cases}$$

Calculate Uhat

Cálculo de $\hat{u}$, $\hat{v}$

$$\begin{aligned} \hat{u} &= u + \theta \cdot (u - u_{ant}) \\ \hat{v} &= v + \theta \cdot (v - v_{ant}) \end{aligned} \tag{84}$$

5.5 Ecuaciones algoritmo discretizadas

La discretización detallada de las ecuaciones que integran la formulación de las distintas etapas de la Clase Optical Flow son las siguientes:

Igrad

Cálculo de $\nabla I^2_{(i,j)}$, donde $I_{x(i,j)}$, $I_{y(i,j)}$,$I_{t(i,j)}$ son conocidos, y calculados por wrapping del frame 1 en el frame 2 a partir del campo de velocidades actual, aplicando interpolación lineal.

$$\nabla I^2_{(i,j)} = max(1e-09, I^2_{x(i,j)} + I^2_{y(i,j)} + \beta^2) \tag{85}$$

Update Dual

Cálculo de las variables Duales $p_{(i,j)}$, $q_{x(i,j)}$, $q_{y(i,j)}$

$$\begin{aligned} p_{x(i,j)} &= p_{x(i,j)} + \sigma(u_{i+1,j} - u_{i,j}) \\ p_{y(i,j)} &= p_{y(i,j)} + \sigma(u_{i,j+1} - u_{i,j}) \\ q_{xx(i,j)} &= q_{xx(i,j)} + \sigma(\hat{v}_{x(i+1,j)} - \hat{v}_{x(i,j)}) \\ q_{xy(i,j)} &= q_{xy(i,j)} + \sigma(\hat{v}_{x(i,j+1)} - \hat{v}_{x(i,j)}) \\ q_{yx(i,j)} &= q_{yx(i,j)} + \sigma(\hat{v}_{y(i+1,j)} - \hat{v}_{y(i,j)}) \\ q_{yy(i,j)} &= q_{yy(i,j)} + \sigma(\hat{v}_{y(i,j+1)} - \hat{v}_{y(i,j)}) \end{aligned} \tag{86}$$

Normalización aplicando la restricción $|p_{(i,j)}| \leq 1$

$$\begin{aligned} p_{norm(i,j)} &= \sqrt{p^2_{x(i,j)} + p^2_{y(i,j)}} \\ p_{(i,j)} &= p_{(i,j)}/max(1, p_{norm(i,j)}) \end{aligned} \tag{87}$$

Normalización aplicando la restricción $|q_{(i,j)}| \leq 1$

$$\begin{aligned} q_{norm} &= \sqrt{q^2_{xx} + q^2_{xy} + q^2_{yx} + q^2_{yy}} \\ q_{(i,j)} &= q_{(i,j)}/max(1, q_{norm(i,j)}) \end{aligned} \tag{88}$$

Update Primal

Cálculo de las variables Primales $\hat{u}_{(i,j)}$, $\hat{v_x}_{(i,j)}$, $\hat{v_y}_{(i,j)}$

$$\begin{aligned} u_{ant(i,j)} &= u_{(i,j)} \\ v_{x_{ant}(i,j)} &= v_{x(i,j)} \\ v_{y_{ant}(i,j)} &= v_{y(i,j)} \end{aligned} \tag{89}$$

$$\begin{aligned} \hat{u}_{(i,j)} &= u_{(i,j)} + \tau[p_{(i,j)} - p_{(i-1,j)} + p_{(i,j)} - p_{(i,j-1)}] \\ \hat{v}_{x(i,j)} &= v_{x(i,j)} + \tau[q_{x(i,j)} - q_{x(i-1,j)} + q_{x(i,j)} - q_{x(i,j-1)}] \\ \hat{v}_{y(i,j)} &= v_{y(i,j)} + \tau[q_{y(i,j)} - q_{y(i-1,j)} + q_{y(i,j)} - q_{y(i,j-1)}] \end{aligned} \tag{90}$$

Update rho

Cálculo de $\rho(u,v)_{(i,j)}$

$$\rho_{(i,j)} = I_{t(i,j)} + (v_{x(i,j)} - v_{x0(i,j)}) \cdot I_{x(i,j)} + (v_{y(i,j)} - v_{y0(i,j)}) \cdot I_{y(i,j)} + \beta \cdot u_{(i,j)} \quad (91)$$

Si $\rho_{(i,j)} < -\tau\lambda(\nabla I^2)_{(i,j)}$

$$\begin{aligned} u_{(i,j)} &= u_{(i,j)} + \tau\lambda\beta \\ v_{x(i,j)} &= v_{x(i,j)} + \tau\lambda I_{x(i,j)} \\ v_{y(i,j)} &= v_{y(i,j)} + \tau\lambda I_{y(i,j)} \end{aligned} \quad (92)$$

Si $\rho_{(i,j)} > +\tau\lambda(\nabla I^2)_{(i,j)}$

$$\begin{aligned} u_{(i,j)} &= u_{(i,j)} - \tau\lambda\beta \\ v_{x(i,j)} &= v_{x(i,j)} - \tau\lambda I_{x(i,j)} \\ v_{y(i,j)} &= v_{y(i,j)} - \tau\lambda I_{y(i,j)} \end{aligned} \quad (93)$$

Si $|\rho_{(i,j)}| <= \tau\lambda(\nabla I^2)_{(i,j)}$

$$\begin{aligned} u_{(i,j)} &= u_{(i,j)} - \rho_{(i,j)}\beta/(\nabla I^2)_{(i,j)} \\ v_{x(i,j)} &= v_{x(i,j)} - \rho_{(i,j)} I_{x(i,j)}/(\nabla I^2)_{(i,j)} \\ v_{y(i,j)} &= v_{y(i,j)} - \rho_{(i,j)} I_{y(i,j)}/(\nabla I^2)_{(i,j)} \end{aligned} \quad (94)$$

Calculate Uhat

Cálculo de $\hat{u}_{(i,j)}$, $\hat{v}_{x(i,j)}$, $\hat{v}_{y(i,j)}$

$$\begin{aligned} \hat{u}_{(i,j)} &= u_{(i,j)} + \theta \cdot \left(u_{(i,j)} - u_{ant}\right) \\ \hat{v}_{x(i,j)} &= v_{x(i,j)} + \theta \cdot \left(v_{x(i,j)} - v_{x_{ant}(i,j)}\right) \\ \hat{v}_{y(i,j)} &= v_{y(i,j)} + \theta \cdot \left(v_{y(i,j)} - v_{y_{ant}(i,j)}\right) \end{aligned} \quad (95)$$

Bibliografía

[TP08] T. Pock. “Fast Total Variation for Computer Vision,”. 2008.

[TA10] A. Chambolle and T. Pock, “A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging,” Journal of Mathematical Imaging and Vision, vol. 40, no. 1, pp. 120–145, May 2010.

[LO92] L. I. Rudin, S. Osher, and E. Fatemi, “Nonlinear total variation based noise removal algorithms,” in Proc. of the 11th annual Int. Conf. of the Center for Nonlinear Studies on Experimental mathematics : computational issues in nonlinear science. Elsevier North-Holland, Inc., 1992, pp. 259–268. [Online]. Available: http://dl.acm.org/citation.cfm?id=142269.142312

[AW13] A. Wedel, T. Brox, T. Vaudrey, C. Rabe, U. Franke, D. Cremers, Stereoscopic Scene Flow Computation for 3D Motion Understanding, In International Journal of Computer Vision, volume 95, 2011.

[SB04] Stephen Boyd and Lieven Vandenberghe. Convex Optimization. Cambridge University Press, March 2004.

[RF70] Rockafellar, R. T. (1970). Convex Analysis, volume 28 of Princeton Mathematical Series, No. 28. Princeton University Press.

[CB13] NVidia (July 2013). CUDA C Best Practices Guide - DG-05603-001 v5.5. Technical report.

[CP13] NVidia (July 2013). CUDA C Programming Guide - PG-02829-001 v5.5. Technical report.