UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

VISUALIZACIÓN DEL CÁLCULO DEL FLUJO ÓPTICO BASADO EN LA TRANSFORMADA DE HERMITE ROTADA SOBRE ECOCARDIOGAFÍAS FETALES

TESIS
QUE PARA OPTAR POR EL GRADO DE
MAESTRÍA EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

PRESENTA:
FABIOLA ARELI PACHECO ARTEAGA

TUTOR:
DR. BORIS ESCALANTE RAMÍREZ
FACULTAD DE INGENIERÍA

CIUDAD UNIVERSITARIA, CD. MX., JUNIO 2018

Agradecimientos

A mi familia por todo el amor, el cariño y la dedicación que me han brindado a lo largo de la vida.

A mis amigos por el apoyo y comprensión que siempre me han mostrado.

A la Universidad Nacional Autónoma de México (UNAM) y su Posgrado en Ciencia e Ingeniería de la Computación (PCIC) por los conocimientos y las oportunidades que me dieron.

Al Dr. Boris Escalante Ramírez por su orientación como tutor y profesor.

Al Dr. Fabian Torres Robles por su ayuda y asesoría en el desarrollo de este proyecto.

Al Laboratorio Avanzado de Procesamiento de Imágenes (LaPI) por todas las experiencias que me compartieron.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por la beca que me otorgó durante mis estudios para poder dedicarme a tiempo completo.

Al proyecto PAPIIT IN116917.

Al proyecto SECITI 110/2015.

Resumen

La presente tesis muestra el desarrollo de métodos numéricos y computacionales que implementan algoritmos de análisis del corazón en imágenes de ultrasonido fetal. Estos métodos incluyen la implementación de un algoritmo rápido de la Transformada de Hermite y de un algoritmo complejo de estimación de movimiento, los cálculos del modo M en ultrasonido y de los coeficientes rotados de la Transformada de Hermite mediante los coeficientes resultantes del algoritmo rápido de la Transformada de Hermite y sus derivadas. Todo esto se aplica para generar una interfaz gráfica acorde a las necesidades del potencial uso de estas herramientas como asistencia al diagnóstico médico. El lenguaje de programación usado es C++ ayudándose con las librerías de ITK y VTK, y utilizando Visual Studio 2008, Qt y CMake.

La ecocardiografía visualizada en la interfaz principal es una imagen 4D con la que el usuario, mediante su manipulación, obtiene una secuencia ordenada de imágenes 2D a la que se le pueden obtener un modo M y un flujo óptico a través de un algoritmo desarrollado en el Laboratorio Avanzado de Procesamiento de Imágenes (LaPI) de la UNAM, que usa coeficientes de Hermite rotados, sus derivadas y una restricción de regiones homogéneas sobre los coeficientes. En la secuencia mostrada para el flujo óptico, se pueden seleccionar manualmente contornos específicos para que únicamente se observe sobre ellos. Los valores sobre estos contornos son utilizados para generar una gráfica que muestra como cambia ahí el flujo óptico.

CONTENIDO

Capítulo 1
Introducción

La presente tesis muestra el desarrollo de métodos numéricos y computacionales que implementan el algoritmo de estimación de movimiento desarrollado por Moya-Albor [1] y Vargas-Quintero [2] y de un algoritmo rápido de la Transformada de Hermite.

Para generar los distintos elementos del funcional de la estimación de movimiento, se calculan los coeficientes de Hermite rotados y sus derivadas a partir de los obtenidos del algoritmo rápido de la Transformada de Hermite. Una vez obtenidos los elementos de la estimación, se acomodan en matrices y vectores para aplicar el método de Sobrerrelajación Sucesiva (SOR), y así obtener su resultado y poder visualizarlo como flechas superpuestas en sus ecocardiografías correspondientes. Aquí se puede segmentar un contorno de interés para observar su flujo óptico correspondiente y generar una gráfica a partir sus valores asociados.

Además de la estimación de movimiento, se calcula el modo M en ultrasonido a partir de una secuencia de imágenes ecocardiográficas 2D con vista de cuatro cámaras, las mismas con las que se calcula la estimación de movimiento, para ser integrados en una interfaz gráfica junto con la estimación de movimiento.

1.1. Planteamiento del Problema

El Instituto Nacional de Perinatología está desarrollando un proyecto colaborativo para generar un sistema experto para el apoyo en la evaluación, clasificación y asignación de riesgo en fetos con alteraciones en el crecimiento. En la parte de análisis cardíaco fetal de este proyecto se necesitan herramientas para que un médico pueda visualizar el flujo óptico junto con el modo M de una secuencia de imágenes ecocardiográficas 2D. El desarrollo de la interfaz debe se para PC's con sistema operativo Windows y estar hecho en lenguaje C++, ocupando las librerías ITK y VTK.

De forma indirecta para realizar estas herramientas se tiene que desarrollar de forma particular el algoritmo rápido para la obtención de derivadas de orden múltiple de Hashimoto y Sklansky [3] de forma particular para obtener las derivadas hasta tercer orden en 2D, y proponer un método para encontrar la estimación de movimiento con base en el algoritmo propuesto por Moya-Albor [1] y Vargas-Quintero [2].Y como elementos complementarios hacer los cálculos para obtener los coeficientes de Hermite rotados, las derivadas de los coeficientes rotados y el modo M de ultrasonido.

1.2. Importancia del Problema

El Instituto Nacional de Perinatología actualmente no cuenta con un sistema como el que se propone y las herramientas de las que dispone están dispersas en distintos Softwares que no contienen información respecto a los métodos y algoritmos que utilizan.

En el análisis cardíaco fetal en imágenes de ultrasonido, la visualización del flujo óptico y del modo M son herramientas que ayudan al médico a agilizar la realización del diagnóstico y detección de anomalías. Por lo que se le puede dar una adecuada atención al paciente mientras le da la posibilidad al médico de analizar y evaluar más casos, además de optimizar la utilización de los recursos disponibles.

Dando seguimiento al desarrollo de la estimación de movimiento hecha por Moya-Albor [1] y Vargas-Quintero [2], se requiere de una implementación que el médico pueda utilizar. Con lo cual, en este trabajo se propone una manera de llevar a cabo la estimación de movimiento antes mencionada mediante la implementación de métodos numéricos y computacionales, además del cálculo del modo M en ultrasonidos para programarlos en lenguaje C++ con ayuda de las librerías de ITK y VTK.

1.3. Contribución

La interfaz desarrollada en esta tesis se integrará como parte del sistema experto propuesto por el Instituto Nacional de Perinatología y le brindará al médico herramientas útiles para el análisis y evaluación del movimiento cardíaco fetal. De forma independiente, los filtros de ITK para obtener los coeficientes de Hermite, los coeficientes de Hermite rotados y la estimación de movimiento pueden ser ocupadas para la programación de otros métodos de este sistema o algún otro que también trabaje con ITK.

Con base en el algoritmo rápido para la obtención de derivadas de orden múltiple de Hashimoto y Sklansky [3] se genera un método para obtener los coeficientes de Hermite de tercer orden en 2D del mismo tamaño que las imágenes de entrada que se procesan. A partir de éstos coeficientes, se calculan los coeficientes rotados y las derivadas de los coeficientes rotados que se ocupan en la estimación de movimiento.

La estimación de movimiento ocupada implica la utilización de un sistema de resolución de ecuaciones, aquí se ha decidido usar el método SOR para lo cual se propone un forma de implementarlo mediante acomodo de los elementos de la

estimación en una matriz y dos vectores.

Para calcular los coeficientes de Hermite y la estimación de movimiento se crean dos filtros de ITK, uno para los coeficientes y otro para la estimación que depende del filtro de coeficientes. Estos se utilizan como cualquier otro método propio de ITK.

1.4. Descripción de la Tesis

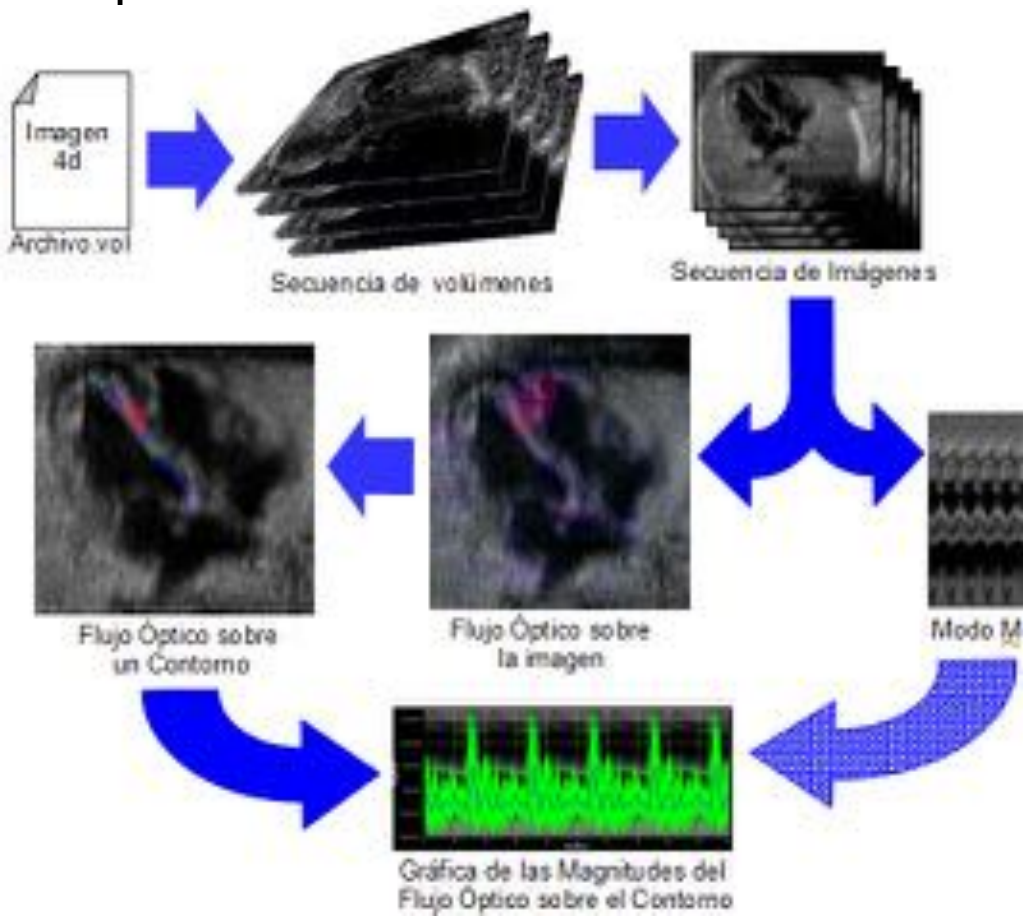


Figura 1.1. Diagrama global del funcionamiento del sistema desarrollado.

El proceso que se sigue en la interfaz se muestra en la Figura 1.1. y consiste en recibir un archivo que contiene una ecocardiografía 4D con vista de cuatro cámaras para desplegarse como una secuencia ordenada de volúmenes de la cual se puede seleccionar un corte para generar una secuencia de imágenes con el mismo ordenamiento que su volumen correspondiente. Con la secuencia tomada, se tiene la opción de obtener el modo M y/o el flujo óptico. El flujo óptico se observa en toda la imagen con opción a visualizarlo en un contorno realizado

por el usuario, en cuyo caso se puede visualizar una gráfica con la magnitud de cada vector seleccionado y su promedio a lo largo de la secuencia que puede o no tener el modo M de fondo dependiendo de si se calculó antes de la estimación de movimiento.

Parte de la interfaz inicial con la que ya cuenta el Instituto Nacional de Perinatología, tiene una función para leer volúmenes 3D. Con base en ésta se genera otra que lee volúmenes 4D y los guarda en el formato de imágenes de ITK para que con sus métodos pueda ser desplegado en la interfaz.

La selección de corte consiste en que el usuario, en alguno de los volúmenes mostrados, selecciona la mejor vista de cuatro cámaras. En cada volumen se extrae la imagen correspondiente a la vista seleccionada y se genera la secuencia de imágenes con la que se trabajara. Dándole una línea de selección, se toman los valores correspondientes a cada imagen y se organiza para forma el modo M.

El flujo óptico que se ocupa es una combinación de métodos diferenciales global y local de flujo óptico [4] sobre los Coeficientes de Hermite rotados [1] con penalización de zonas homogéneas [2]. La visualización que se despliega es el resultado de la estimación de movimiento entre la imagen actual y la siguiente.

El coeficiente cero obtiene una versión suavizada de la imagen, los coeficientes de primer orden, bordes y los de segundo orden, cruces por cero. Una de las características principales de los coeficientes rotados es que resaltan características unidimensionales de las imágenes. Por lo anterior, para la estimación de movimiento se utilizan los coeficientes rotados de Hermite hasta orden dos.

Para ahorrar la implementación de un método numérico que aproxime las derivadas de los coeficientes rotados de Hermite se pueden ocupar los coeficientes de Hermite de un orden superior para generar las derivadas correspondientes, por lo que se obtienen los coeficientes hasta el tercer orden para posteriormente realizar las derivadas de los coeficientes rotados de segundo orden. Con la intención de agilizar la obtención los coeficientes de Hermite de orden tres se aplica un algoritmo basado en la estimación de derivadas parciales descrita en [3].

Ya que la estimación de movimiento se obtiene a partir de un funcional, se emplea el método de Sobrerrelajación Sucesiva (SOR) para obtener su solución. El resultado se despliega como flechas de los colores que van desde el rojo para las magnitudes más grandes hasta el azul para las más pequeñas. Aquí se puede segmentar un contorno para mostrar exclusivamente las flechas seleccionadas con sus magnitudes asociadas. Con éstas se puede generar una gráfica que

muestra como cada magnitud varía a lo largo del tiempo y su promedio.

Insight Segmentation and Registration Toolkit (ITK) y Visualization Toolkit (VTK) son un conjunto de herramientas de software multiplataforma de código abierto para el análisis y procesamiento de imágenes, realización de gráficos 3D, modelado, renderizado, visualización científica y visualización de información [5], [6]. Por estas características es que se especificó que serían las principales librerías auxiliares en el desarrollo de la interfaz.

El presente documento está dispuesto de la siguiente manera:

- Capítulo 1. Introducción: Describe de manera global el contenido de la tesis y los procesos que se implementaron para desarrollar la interfaz.
-
- Capítulo 2. Planteamiento del problema: Define el contexto y el problema que se quiere solucionar, así como los requisitos especificados para ello.

- Capítulo 3. Ecografía: Presenta los conceptos e importancia de la ecocardiografía fetal y del modo M.

- Capítulo 4. Transformada de Hermite: Desarrolla el marco teórico en el que se basa la Transformada de Hermite y la Transformada de Hermite rotada, además del algoritmo rápido para la obtención de derivadas de orden múltiple.

- Capítulo 5. Flujo Óptico: Contiene información respecto a los métodos diferenciales de estimación de movimiento, enfocándose principalmente la combinación de métodos diferenciales local y global que se combinaron y adaptaron para ocupar a los coeficientes de Hermite rotados en lugar de la imagen. También incluye la restricción de regiones homogéneas para los coeficientes de Hermite que se integró en la estimación.

- Capítulo 6. Desarrollo: Describe la implementación efectuada para obtener el flujo óptico, el modo M y la visualización a través de la creación de una interfaz gráfica.

- Capítulo 7. Pruebas y Resultados: Muestra como quedó la interfaz y manipula el número de iteraciones y escalas en el flujo óptico para comparar su funcionamiento.

- Capítulo 8. Conclusiones: Muestra que fue lo que se consiguió de la realización de este trabajo y que es lo que la interfaz requiere que se le agregue o mejore.

Capítulo 2
Planteamiento del problema

Se necesita la medición automatizada de parámetros biométricos y biofísicos en fetos para clasificar el riesgo específico paciente-patología y proporcionar una correcta asignación de factores de riesgo. De esta manera se podrá tomar la mejor decisión clínica y la posterior aplicación de maniobras preventivas, así como la correcta asignación del nivel de atención que le corresponde. Permitiendo mejorar la aplicación de los recursos y disminuir la dependencia de operadores especializados en la evaluación fetal.

Con lo cual, el Instituto Nacional de Perinatología (INPer) se plantea la realización de un sistema experto para el apoyo en la evaluación, clasificación y asignación de riesgo en fetos con alteraciones en el crecimiento. Las imágenes, volúmenes ultrasonográficos y trazos de electrocardiografía materno-fetal captados por personal técnico se enviaran mediante Internet para ser analizados y evaluados de forma remota por personal especializado tanto en el área de conocimiento como en los sistemas en desarrollo.

El proyecto completo es desarrollado en conjunto por:

- Laboratorio de Neuroimagenología, CBI, UAM-Iztapalapa.
- Laboratorio de Fisiología humana y Laboratorio de Fenómenos Fisiológicos Perinatales, CBI, CBS, UAM-Iztapalapa.
- Laboratorio de Imagenología Biomédica, CCADET, UNAM.
- Laboratorio Avanzado de Procesamiento de Imágenes, Facultad de Ingeniería, UNAM.
- Departamento de Medicina y Cirugía Fetal, Instituto Nacional de Perinatología.

Uno de los módulos contenidos es el de análisis automático de movimiento de corazón fetal en imágenes de ultrasonido, el cual requiere del desarrollo y validación de un algoritmo multiescala que permita estimar el movimiento de estructuras cardíacas a través del cálculo de flujo óptico en secuencias de imágenes médicas del corazón, y que permitan hacer observaciones del comportamiento mecánico del corazón. Para esto, el Laboratorio Avanzado de Procesamiento de Imágenes desarrolló métodos de segmentación y estimación de movimiento en imágenes cardíacas de ultrasonido [1], [2] mediante la utilización de la Transformada de Hermite.

Dando seguimiento a los métodos creados y cubriendo parte de la necesidad de herramientas para la evaluación de la función cardíaca del módulo, se necesita

una interfaz gráfica con la cual el operador pueda interactuar para ver y manipular los resultados de los algoritmos para realizar su evaluación. El desarrollo de la interfaz debe estar programado en el lenguaje C++ con las librerías ITK y VTK, para funcionar sobre el sistema operativo Windows.

2.1. Objetivo general

Implementar algoritmos de análisis del corazón en imágenes de ultrasonido fetal mediante una interfaz gráfica para PC's con Sistema Operativo Windows, programada en lenguaje C++ con ayuda de las librerías ITK y VTK, compilada y editada con el software Microsoft Visual Studio y QT para la creación e interacción con las ventanas de despliegue. Que incluya la obtención del modo M, la realización del algoritmo rápido para obtener obtener los coeficientes de Hermite y sus rotaciones hasta orden 3 en dos dimensiones para poder implementar el algoritmo de estimación de movimiento propuesto por Vargas-Quintero [2] el cual toma la estimación de movimiento propuesta por Moya-Albor [1] y le aplica una restricción de regiones homogéneas, mediante métodos numéricos. El modo M y el flujo óptico deben poder visualizarse en conjunto.

La entrada de la interfaz es un volumen ecocardiográfico 4D al que se le puede seleccionar una secuencia de cortes y de esta obtener el modo M y la estimación de movimiento. La estimación de movimiento se visualiza mediante flechas sobre la secuencia de cortes, en dicha visualización el usuario puede seleccionar un contorno de interés para visualizar su flujo óptico correspondiente y desplegar una gráfica respecto a esos valores.

El desarrollo generado se acoplara posteriormente a la interfaz del proyecto completo del Instituto Nacional de Perinatología como parte del módulo de análisis automático del movimiento de corazón fetal, en donde será complementado y validado para ser usado por un médico como una herramienta para optimizar el diagnóstico y evaluación de anomalías presentadas en ecocardiografías fetales.

La interfaz debe contar con distintos métodos y herramientas para:

- Leer ecocardiografías fetales 4D para:
  - Manipular las ecocardiografías fetales 4D.
  - Seleccionar un corte por el usuario, y con este:
    - Obtener y manipular el modo M de las secuencias de imágenes pertenecientes al corte seleccionado.
      - Invertir niveles de gris.
      - Seleccionar un periodo a analizar.
      - Aumentar el número de ciclos observados.

- Obtener la estimación de movimiento.
  - Visualizar el Flujo óptico, resultado de la estimación de movimiento en:
    - Toda la imagen
    - Un contorno definido por el usuario.
      - Graficar la estimación de movimiento del contorno sobrepuesta al modo M.

Capítulo 3
Ecografía

Las ondas de ultrasonido fueron descubiertas en el año 1700 por el biólogo italiano Lazzaro Spallanzani al observar a los murciélagos. A finales de la segunda Guerra Mundial se desarrollan los primeros prototipos de diagnóstico médico usando estas ondas. En la década de 1950 se acepta al ultrasonido como instrumento de diagnóstico médico [7].

La frecuencia del ultrasonido es superior a los 20 Khz aunque en medicina, para diagnóstico clínico, frecuentemente se ocupa un rango de 2 a 30 Mhz. El procedimiento consiste en una onda de ultrasonido que atraviesa un tejido y cuyos haces ultrasónicos rebotan hacia la fuente receptora (transductor), esto es llamado "eco", los ecos recibidos son transformados en píxeles cuyo brillo dependerá de la intensidad del eco. La matriz que se forma es llamada "ecografía" [8], en la Figura 3.1. se ve como se forma una ecografía a partir de la intensidad de los ecos.

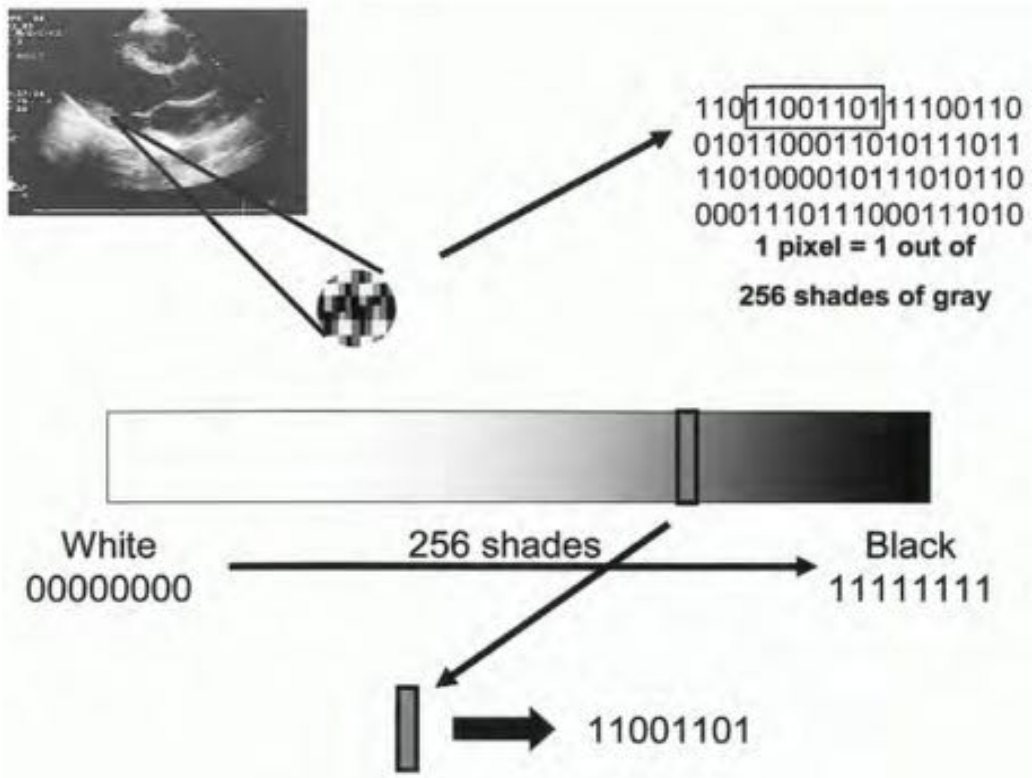


Figura 3.1. Conversión de los ecos en píxeles con 256 niveles de gris para formar una ecografía [9].

3.1. Ecocardiografía

En 1950, el investigador Alemán W. D. Keidel trató de determinar volúmenes cardíacos transmitiendo ondas ultrasónicas a través del corazón y tomando el efecto del ultrasonido del otro lado del pecho. En 1953 el Dr. Helmut Hertz de Suecia obtuvo un ultrasonoscopio comercial y junto con el cardiólogo Dr. Inge Edler lo empezaron a usar para examinar el corazón, esta colaboración es el inicio de la ecocardiografía clínica [9].

3.1.1. Modalidades

Existen tres modalidades principales de visualización de ecocardiografías, las cuales se muestran en la Figura 3.2. y consisten en:

- El modo A (o de Amplitud) es una representación gráfica de la señal mostrando la amplitud a distintas profundidades.
- El modo M (o de Movimiento) es una representación gráfica de la señal de una estructura en movimiento en la cual la amplitud es el eje vertical y el tiempo y la profundidad son el eje horizontal.
- El modo B (o de Brillo) es una representación pictórica de la suma de los ecos en diferentes direcciones (axial, lateral) [8].

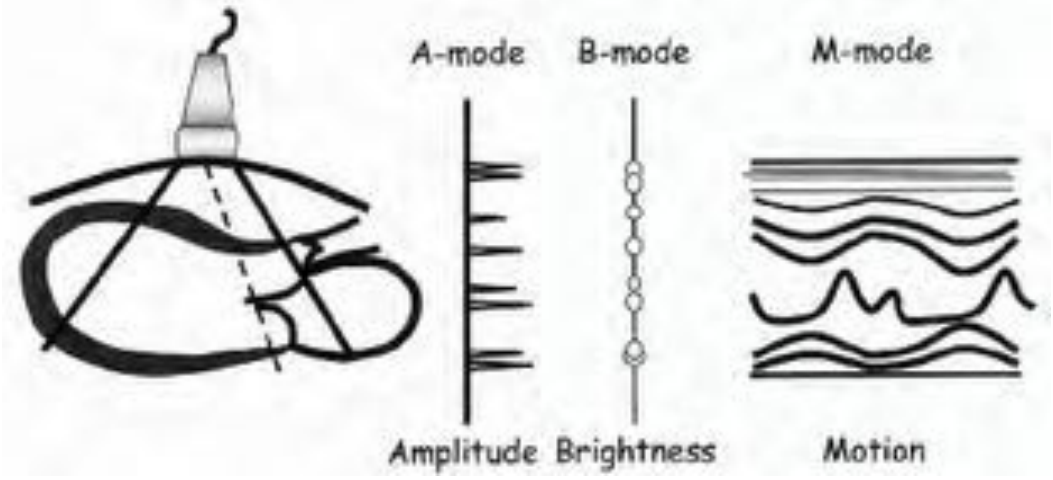


Figura 3.2. Modalidades principales de despliegue (A, B y M) [9].

3.1.1.1. Modo M

Actualmente el modo M es la columna vertebral de la ecocardiografía clínica. Se coloca el transductor sobre diferentes ventanas acústicas de la pared torácica, pudiendo registrar imágenes unidimensionales de estructuras cardíacas e

inferencias estructurales, dimensionales y funcionales como se muestra en la Figura 3.3. El modo M se crea recopilando la información gráfica correspondiente a un único haz en una secuencia de tiempo determinado [9].

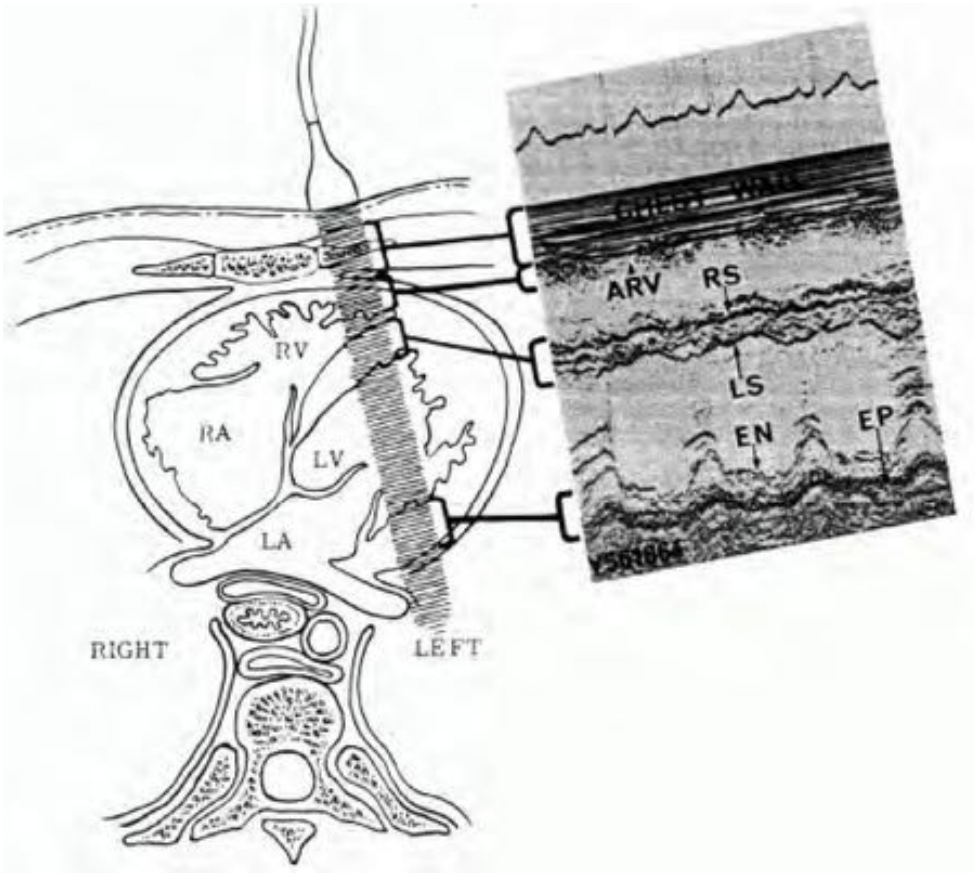


Figura 3.3. Relación entre el transductor y las estructuras del pecho en el modo M donde ARV es la pared ventricular derecha anterior, EN es el endocardio ventricular izquierdo posterior, EP es el epicardio ventricular izquierdo posterior, LS es el septo izquierdo y RD es el septo derecho [9].

3.1.2. Ecocardiografía Fetal

3.1.2.1. Fisiología Cardíaca Fetal [10], [11]

La oxigenación de la sangre fetal ocurre en la placenta, la cual está intercomunicada con su sistema circulatorio. Una vez oxigenada, regresa al corazón del feto mediante el sistema venoso umbilical llevando sangre llena de oxígeno.

El corazón fetal comienza a desarrollarse como un simple tubo que se contrae con cuatro segmentos: el segmento sinoauricular (origen de la aurícula), el ventrículo primitivo (origen del ventrículo izquierdo), el bulbo cardíaco (origen del ventrículo derecho), y el conotruncus (el origen de la arteria pulmonar y la aorta). Alrededor del día 26 de vida embrionaria, la contracción del corazón puede ser visualizada por ultrasonido.

Los órganos inferiores del feto son irrigados por los dos ventrículos del corazón por lo que se suele considerar la salida cardíaca fetal como la salida total del corazón, esto corresponde a la salida ventricular combinada. A diferencia de un adulto, los ventrículos izquierdo y derecho del feto no bombean en serie por lo que el ventrículo derecho expulsa aproximadamente dos tercios de la salida cardíaca fetal mientras que el izquierdo sólo un poco más de un tercio. La saturación de oxígeno en el ventrículo derecho es menor que en el izquierdo.

Factores como el estado de sueño, la actividad electrocortical y la actividad uterina, afectan transitoriamente a la circulación fetal.

3.1.2.2. Malformaciones Cardíacas Fetales [12], [11]

En la vida embrionaria y fetal pueden ocurrir variaciones fenotípicas que resulten en la muerte o en discapacidad grave, a estas variaciones se les denominan como anomalías congénitas mayores. La estimación de la frecuencia de las principales anomalías evaluadas en el ultrasonido anteparto son inexactas debido a las limitaciones de los métodos de imagen. Sin embargo, la incidencia general de anomalías mayores se considera que está en el rango de 2% a 3% de todos los embarazos.

Algunos factores que influyen en la detección de anomalías fetales mediante el ultrasonido son:

- Tipo de anomalía fetal.
- Edad gestacional.
- Factores de riesgo.
- Tipo y calidad de imagen

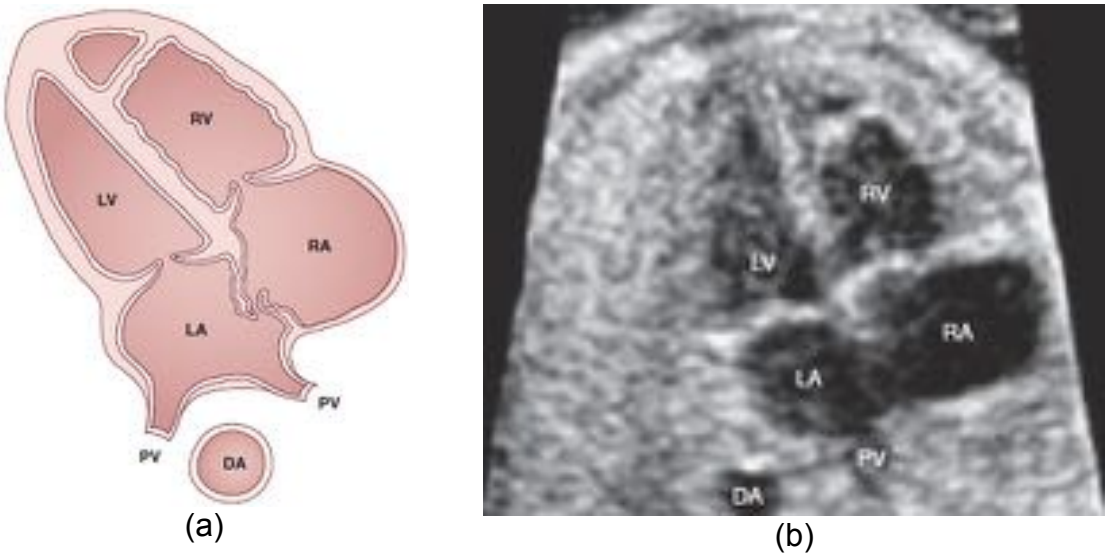
Aproximadamente el 1% de todos los bebés que nacen vivos presentan una anomalía cardíaca estructural, esta tasa aumenta al incluir abortos y bebés que nacen muertos. La distribución de las anomalías cardíacas no es igual entre los diversos defectos estructurales. El defecto septo ventricular es la lesión más frecuente y en la mayoría de los casos, se desconoce la causa exacta de la malformación del corazón fetal.

La mayoría de las arritmias fetales se resuelven por sí solas y no necesitan tratamiento. El tratar a un feto por arritmia debe hacerse exclusivamente después de comparar los beneficios con las posibles complicaciones maternas y fetales y después de discutirlo con la madre. El diagnóstico se hace empíricamente mediante la observación del ritmo con imágenes 2D y puede confirmarse mediante un análisis del modo M.

La ecocardiografía en modo M permite la visualización simultánea de las contracciones auricular y ventricular cuando, guiada por una imagen 2D, la línea de muestreo en modo M se dirige a través de la aurícula y el ventrículo.

3.1.2.3. Vista de cuatro cámaras [11]

La detección de enfermedades del corazón fetal es uno de los aspectos más desafiantes y menos exitoso de la ecografía fetal. La cardiopatía congénita es la malformación congénita más común. Aunque la mayoría de los estudios sugieren una incidencia de aproximadamente 8 de 1000 bebés nacidos vivos, esta cifra incluye muchas formas de enfermedad que nunca requerirán atención médica o quirúrgica. Su incidencia durante el primer y segundo trimestre es considerablemente mayor que la incidencia a término. Es necesario mejorar en la detección y diagnóstico prenatal de la cardiopatía congénita ya que esto disminuiría la mortalidad en fetos y neonatos.



(a) (b)

Figura 3.4. Vista de cuatro cámaras donde PV es la vena pulmonar, LA es la aurícula izquierda, DA es la aorta descendente, LV es el ventrículo izquierdo, RA es la aurícula derecha y RV es el ventrículo derecho. (a) Esquema. (b) Ecocardiografía [11].

Poco después de desarrollar la imagen cardíaca fetal bidimensional, los investigadores propusieron la visión de cuatro cámaras o cavidades, como la mostrada en la Figura 3.4., como un enfoque eficaz de la proyección fetal para detectar la cardiopatía congénita. Hasta la fecha, la visión de cuatro cámaras se ha mantenido como el estándar para la evaluación de las ecocardiografías fetales en los embarazos de bajo riesgo. Para mejorar aún más la detección, otros han sugerido el flujo a color junto con la visión de cuatro cámaras pero esta técnica requiere conocimientos especializados y equipo ya que requiere procesar la detección de movimiento para representar el flujo sanguíneo.

La vista de cuatro cámaras proporciona la mejor perspectiva individual para evaluar la estructura ventricular. Ambos ventrículos deben extenderse hasta el ápice cardíaco, y ambos deben apretar simétricamente durante la sístole. Permite realizar una evaluación cuantitativa de la función sistólica y diastólica ventricular si se sospecha una anormalidad.

Más recientemente, varios investigadores han sugerido que la obtención de imágenes en 3D, en la que los volúmenes pueden adquirirse en cuestión de segundos y posteriormente revisarse interactivamente, permitiendo la visualización de prácticamente cualquier plano, puede facilitar y mejorar la evaluación de los tractos en la visión de cuatro cámaras. Sin embargo, la resolución con imágenes en 3D permanece baja, y actualmente se requiere una considerable experiencia para evaluar volúmenes de datos, a pesar de una gran cantidad de algoritmos y técnicas dirigidas a facilitar y mejorar el análisis de la imágenes 3D.

Las imágenes 3D pueden facilitar la detección prenatal de la cardiopatía congénita permitiendo la evaluación virtual o automatizada de todo el corazón fetal después de una adquisición simple y corta. Cuando se desea, los conjuntos de datos pueden transmitirse electrónicamente a expertos en ubicaciones remotas, donde ellos pueden realizar un examen virtual como si estuvieran escaneando al propio paciente. Por lo que el procesamiento de un volumen completo de datos de una imagen 3D es menos dependiente del operador y ocupa poco tiempo en su evaluación.

Capítulo 4
Transformada de Hermite [13]

Varias aplicaciones de procesamiento de imágenes y de visión computacional necesitan datos de las imágenes puedan interpretarse con patrones visuales significativos. Generalmente esto implica determinar las relaciones espaciotemporales de los datos originales para procesarlos de manera local.

Para realizar el procesamiento local usualmente se usa una función ventana, comúnmente cuadrada, con la que se multiplica la imagen un número suficiente de veces para describir por completo a la imagen. Esto puede repetirse con distintos tamaños de ventana.

En cada posición de la ventana se realizan pasos específicos de procesamiento, estos se definen dependiendo de los patrones que se buscan específicamente, lo que equivale a seleccionar los patrones visuales que son más relevantes a priori. Debido a la dificultad de elección en base a elementos puramente teóricos se utiliza como referencia el sistema de visión humana.

Las funciones ventana que satisfacen todas las condiciones de ortogonalidad tienen dos desventajas. La primera es que generalmente son mucho más grandes que el espacio entre ventanas, y la segunda es que a menudo tiene un anillado considerable y el suavizado es menor que el de las funciones deducidas a partir del traslape de los campos receptivos en la perceptivos de la visión humana.

Para obtener un mayor suavizado en regiones homogéneas, se necesita ignorar la condición de ortogonalidad entre las funciones base de ventanas conjuntas. Por ejemplo, las expansiones de Gabor, donde se usan ventanas de suavizado (Gaussianas). Aunque, no es necesario ignorar la condición de ortogonalidad entre la funciones base pertenecientes a una ventana. La Transformada de Hermite difiere de las expansiones de Gabor en este aspecto. Usar ventanas que se traslapan entre si con funciones Gaussianas produce un efecto de suavizado, dicho efecto es más natural y menos molesto para el humano que el producido por ventanas cuadradas sin traslape.

El tamaño (o escala) de la ventana es un parámetro importante y seleccionarlo es un problema fundamental. Por un lado, tiene que ser suficientemente grande para permitir una alta reducción de datos. Por otro, la complejidad del análisis dentro de cada ventana se incrementa rápidamente con el tamaño de ventana. Hay dos posibles aproximaciones a este problema. Uno, se puede seleccionar una ventana de tamaño fijo y realizar una análisis dentro de cada ventana que sea suficientemente compleja para incluir patrones visuales de interés. Otro, limita la

complejidad del análisis que se ejecuta en cada ventana, y subsecuentemente determinar en tamaño de ventana necesario para describir localmente a la imagen con suficiente exactitud. Así, en vez de restringir el proceso a una única escala, se repite el mismo procedimiento en múltiples escalas y luego se utilizan las salidas de la etapa de procesamiento para elegir el tamaño óptimo.

En las técnicas de procesamiento local, la descripción de la imagen por patrones elegidos a priori seguida del uso de métodos estadísticos para modelar las dependencias restantes, usualmente tiene mejor resultado que el uso exclusivo de aproximaciones estadísticas.

Usando patrones con distintas orientaciones se aproxima más al sistema de visión humana, y esto se incluye en este desarrollo. En visión computacional, el problema de procesamiento de imágenes se aborda desde el punto de vista de la interpretación. Como resultado, es de principal interés encontrar características como bordes y líneas, lo que involucra el uso de la primera y segunda derivadas junto con un filtro paso bajas. Las derivadas de Gaussiana pueden modelar operaciones filtro de la visión humana con la misma precisión que los ampliamente usados filtros de Gabor con la ventaja de usar menos parámetros.

4.1. Transformada Polinomial [13]

La Transformada Polinomial es una técnica en la que una señal es aproximada localmente por polinomios, el análisis involucra dos pasos:

1. La señal original $L(x)$ se localiza en varias posiciones multiplicándose por una ventana $V(x)$. Una descripción completa de señal requiere que el proceso de localización se repita con la ventana en suficientes posiciones y considerando un espacio equidistante entre ventanas. Para encontrar la función ventana $V(x)$ se construye la función de pesos con un periodo T :

$$W(x)=\sum_k V(x-kT) \tag{4.1}$$

con $W(x)$ diferente de 0 para toda x se tiene:

$$L(x)=\frac{1}{W(x)}\sum_k L(x)\cdot V(x-kT) \tag{4.2}$$

Lo que garantiza que las señales localizadas mediante $L(x)\cdot V(x-kT)$, para cada posición de kT , contengan suficiente información de la señal original.

2. Aproximar parte de la señal dentro de la ventana $V(x-kT)$ por un polinomio. Como funciones base para la expansión polinómica, se toman los polinomios $G_n(x)$ con orden $[G_n(x)]=n$, ortonormales respecto a $V^2(x)$, esto es:

$$\int_{-\infty}^{+\infty} V^2(x)\,G_m(x)\,G_n(x)\,dx=\delta_{mn} \quad , \tag{4.3}$$

donde δ_{mn} denota la función Kronecker, y los polinomios ortonormales para una función ventana arbitraria $V^2(x)$ están dados por:

$$G_n(x)=\frac{1}{\sqrt{M_{n-1}M_n}}\begin{vmatrix} c_0 & c_1 & \dots & c_n \\ c_1 & c_2 & \dots & c_{n+1} \\ \vdots & \vdots & & \vdots \\ c_{n-1} & c_n & \dots & c_{2n-1} \\ 1 & x & \cdots & x^n \end{vmatrix} \tag{4.4}$$

teniendo el determinante M_n definido por:

$$M_n=\left|c_{i+j}\right|_{i,\,j=0,\dots,n} \quad \text{con} \quad M_{-1}=1 \tag{4.5}$$

y el momento c de orden n por:

$$c_n=\int_{-\infty}^{+\infty} x^n\,V^2(x)\,dx \quad . \tag{4.6}$$

Si $V^2(x)$ es par se pueden obtener las expresiones explícitas para los polinomios ortonormales, los cuales, hasta orden 3, son:

$$G_0(x)=\frac{1}{\sqrt{c_0}} \ , \ G_1(x)=\frac{x}{\sqrt{c_2}} \ , \ G_2(x)=\frac{c_0x^2-c_2}{\sqrt{c_0(c_0c_4-c_2^2)}} \text{ y } G_3(x)=\frac{c_2x^3-c_4x}{\sqrt{c_2(c_2c_6-c_4^2)}}$$

Para condiciones muy generales, la señal original $L(x)$ se tiene como:

$$V(x-kT)[L(x)-\sum_{n=0}^{\infty} L_n(kT)\cdot G_n(x-kT)]=0 \tag{4.7}$$

con:

$$L_n(kT)=\int_{-\infty}^{+\infty} L(x)\cdot G_n(x-kT)V^2(x-kT)\,dx \quad . \tag{4.8}$$

Para garantizar la convergencia de la expansión en series de la ecuación (4.7) en la mayoría de funciones ventana, se requiere que $L(x)$ sea analítica y finita para toda x. Así el error de aproximación entre una señal y un polinomio puede hacerse arbitrariamente pequeño eligiendo un orden suficientemente grande para la expansión polinomial.

La ecuación (4.8) implica que los coeficientes $L_n(kT)$ pueden ser derivados de la señal $L(x)$, al convolucionar esta última con las siguientes funciones filtro:

$$D_n(x)=G_n(-x)V^2(-x) \quad . \tag{4.9}$$

Después de la convolución, se realiza un muestreo en múltiplos de T. Este mapeo, a partir de la señal original $L(x)$ hasta los coeficientes polinomiales $L_n(kT)$, es llamado Transformada Polinomial Directa.

4.1.1. Transformada Polinomial en dos dimensiones [13]

La Transformada Polinomial puede generalizarse para dos dimensiones dando una función ventana $V(x,y)$ y los polinomios ortonormales $G_{m,n-m}(x,y)$ donde m y $n-m$ representan el grado respecto a x e y respectivamente, esto es:

$$\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} V^2(x,y)G_{m,n-m}(x,y)G_{j,i-j}(x,y)\,dx\,dy=\delta_{ni}\delta_{mj} \quad , \tag{4.10}$$

para $n,i=0,1,\ldots,\infty$, $m=0,1,\ldots,n$ y $j=0,\ldots,i$.

La descomposición de señales 2D en polinomios se convierte en:

$$L(x,y)=\sum_{n=0}^{\infty}\sum_{m=0}^{n}\sum_{(p,q)\in S} L_{m,n-m}(p,q)\cdot P_{m,n-m}(x-p,y-q) \tag{4.11}$$

donde (p,q) está en el rango de todas las coordenadas en una látice de muestreo 2D si la función de peso es:

$$W(x,y)=\sum_{(p,q)\in S} V(x-p,y-q) \tag{4.12}$$

es distinta de cero para todas las coordenadas (x,y) .

Los coeficientes polinomiales $L_{m,n-m}(p,q)$ se derivan de convolucionar la imagen con las siguientes funciones filtro:

$$D_{m,n-m}(x,y)=G_{m,n-m}(-x.-y)\,V^2(-x,-y) \tag{4.13}$$

seguido por el muestreo de la salida en $(p,q)\in S$. Las funciones patrón usadas para interpolar los coeficientes polinomiales son definidas por:

$$P_{m,n-m}(x,y)=G_{m,n-m}(x,y)\,V(x,y)/W(x,y) \tag{4.14}$$

para $n,i=0,1,\ldots,\infty$ y $m=0,1,\ldots,n$.

4.1.1.1. Proyección Unidimensional [13]

Un análisis de una imagen debería intentar descomponerla en patrones perceptualmente significativos, en visión computacional y percepción visual, los patrones unidimensionales como bordes y líneas tienen un rol central. El mejor ajuste local en una dimensión puede encontrarse con ayuda de la Transformada Polinomial, esto equivale a encontrar los coeficientes de las derivadas de las funciones ventana y separarlos dependiendo de su orden

Usando un criterio de error cuadrático ponderado, se minimiza:

$$E^2=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}[K(x\cos\theta+y\sin\theta)-L(x,y)]^2V^2(x,y)\,dx\,dy \tag{4.15}$$

sobre todos los patrones unidimensionales K y ángulos θ .

Se define la función ventana unidimensional:

$$V^2{}_\theta(u)=\int_{-\infty}^{\infty}V^2(u\cos\theta-v\sin\theta,u\sin\theta+v\cos\theta)\,dv \tag{4.16}$$

proyectando la función $V^2(x,y)$ sobre un eje que forma un ángulo θ con el eje x . Esta función ventana es independiente de la orientación si $V^2(x,y)$ es rotacionalmente simétrico. El patrón unidimensional $K(u)$ puede expandirse en la base de polinomios ortonormales { $F_{n,\theta}(u);n=0,1,\ldots$ } sobre $V^2{}_\theta(u)$, esto es:

$$V_\theta(u)\left[K(u)-\sum_{n=0}^{\infty}K_{n,\theta}\cdot F_{n,\theta}(u)\right]=0 \quad . \tag{4.17}$$

Sustituyendo la expansión polinomial 2D y 1D para $L(x, y)$ y $K(u)$, respectivamente, en la ecuación (4.15), y tomando la derivada parcial con respecto a $K_{n,\theta}$ queda la siguiente solución óptima:

$$K_{n,\theta}=\sum_{k=0}^{n}\sum_{l=0}^{k} L_{l,k-l}\cdot h_{n,\theta}(l,k-l) \tag{4.18}$$

para los coeficientes de patrones unidimensionales, donde:

$$h_{n,\theta}(l,k-l)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F_{n,\theta}(x\cos\theta+y\sin\theta)\cdot G_{l,k-l}(x,y)V^2(x,y)\,dx\,dy \tag{4.19}$$

es una función angular completamente determinada por $V^2(x, y)$.

Los polinomios ortonormales $F_{n,\theta}(u)$ y la función angular pueden determinarse sin el conocimiento explícito de $V_\theta(u)$. La ecuación (4.4) implica que únicamente los momentos:

$$c_{n,\theta}=\int_{-\infty}^{\infty} u^n V^2{}_\theta(u)du \tag{4.20}$$

son necesarios para especificar a los polinomios ortogonales. El cálculo de estos momentos puede basarse directamente en $V^2(x, y)$, ya que:

$$c_{n,\theta}=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x\cos\theta+y\sin\theta)^n V^2(x,y)\,dx\,dy \quad . \tag{4.21}$$

De la ortogonalidad de los polinomios $F_{n,\theta}(u)$, pueden derivarse las siguientes propiedades para la función angular:

$$h_{n,\theta}(l,k-l)=0 \quad \text{si} \quad k>n \tag{4.22}$$

$$\sum_{k=0}^{\infty}\sum_{l=0}^{k} h_{m,\theta}(l,k-l)\,h_{n,\theta}(l,k-l)=\delta_{mn} \quad . \tag{4.23}$$

El error de aproximación angular:

$$E^2=\sum_{k=0}^{\infty}\sum_{l=0}^{k} L_{l,l-k}^2-\sum_{n=0}^{\infty} K_{n,\theta}^2 \tag{4.24}$$

puede minimizarse sobre el ángulo θ maximizando la energía direccional

$$\sum_{n=0}^{\infty} K_{n,\theta}^2 \quad , \tag{4.25}$$

donde $K_{n,\theta}$ se determina por los coeficientes polinomiales bidimensionales de la ecuación (4.18). Usualmente, en la práctica, los primeros términos de esta medida de energía direccional son suficientes para una buena estimación de la dirección óptima.

Si la imagen original $L(x, y)$ es localmente unidimensional, entonces el error de estimación debe ser cero para el ángulo óptimo θ, por lo que los coeficientes polinomiales bidimensionales deben satisfacer:

$$L_{l,k-l} = \sum_{n=k}^{\infty} K_{n,\theta} \cdot h_{n,\theta}(l, k-l) \quad . \tag{4.26}$$

Si la imagen $L(x, y)$ no es localmente unidimensional, entonces la expresión anterior puede usarse como una aproximación 1D para los coeficientes polinomiales 2D.

4.2. Transformada de Hermite [13]

En el caso donde la función ventana local de la Transformada Polinomial es Gaussiana:

$$V(x) = \frac{1}{\sqrt{\sqrt{\pi}\sigma}} \exp\left(\frac{-x^2}{2\sigma^2}\right) \quad , \tag{4.27}$$

donde el factor de normalización es tal que $V^2(x)$ tiene energía unitaria. Los polinomios ortogonales que están asociados con $V^2(x)$ son conocidos como Polinomios de Hermite, por lo que el resultado de la técnica de descomposición local resultante es la Transformada de Hermite.

Ya que la función de pesos $W(x)$, es periódica con periodo T y puede ser expandida en series de Fourier, esto es:

$$W(x) = \frac{\sqrt{2\sqrt{\pi}\sigma}}{T} w(x) \tag{4.28}$$

con:

$$w(x)=1+2\sum_{k=1}^{\infty}\exp\left[-\frac{1}{2}\left(k\frac{2\pi\sigma}{T}\right)^2\right]\cdot\cos\left(k\frac{2\pi x}{T}\right) \quad . \tag{4.29}$$

Las principales propiedades de la Transformada de Hermite son determinadas por la funciones filtro, ya que estas determinan la información que se hará explícita en los coeficientes. De la ecuación (4.9), se deriva que:

$$D_n(x)=\frac{(-1)^n}{\sqrt{2^n n!}}\cdot\frac{1}{\sigma\sqrt{\pi}}H_n\left(\frac{x}{\sigma}\right)e^{-x^2/\sigma^2} \tag{4.30}$$

ya que los Polinomios de Hermite $\{H_n(x/\sigma); n=0,1,\ldots\}$ son ortogonales sobre la ventana Gaussiana $V^2(x)$, la función filtro $D_n(x)$ es igual a la derivada de la Gaussiana de orden n, es decir:

$$D_n(x)=\frac{(-1)^n}{\sqrt{2^n n!}}\cdot\frac{d^n}{d\left(\frac{x}{\sigma}\right)^n}\left[\frac{1}{\sigma\sqrt{\pi}}e^{-x^2/\sigma^2}\right] \tag{4.31}$$

siendo su Transformada de Fourier:

$$d_n(\omega)=\frac{(-1)^n}{\sqrt{2^n n!}}\cdot(\omega j\sigma)^n e^{-(\omega\sigma)^2/4} \tag{4.32}$$

donde $w(x)$ está definida por la ecuación (4.29).

Para órdenes muy grandes, los picos de frecuencia se juntan demasiado, por lo que ofrecen poca información adicional. Por ello, en la práctica, la Transformada de Hermite se limita a unos cuantos términos. Los filtros para $n=0,\ldots,4$, juntos con sus respectivas Transformadas de Fourier se muestran en la Figura 4.1.

La Transformada de Hermite es una técnica alternativa de expansión de señales que produce perfiles de campo receptivo que son derivadas de Gaussiana. Siendo el primer y segundo orden de derivación los que más generan interés.

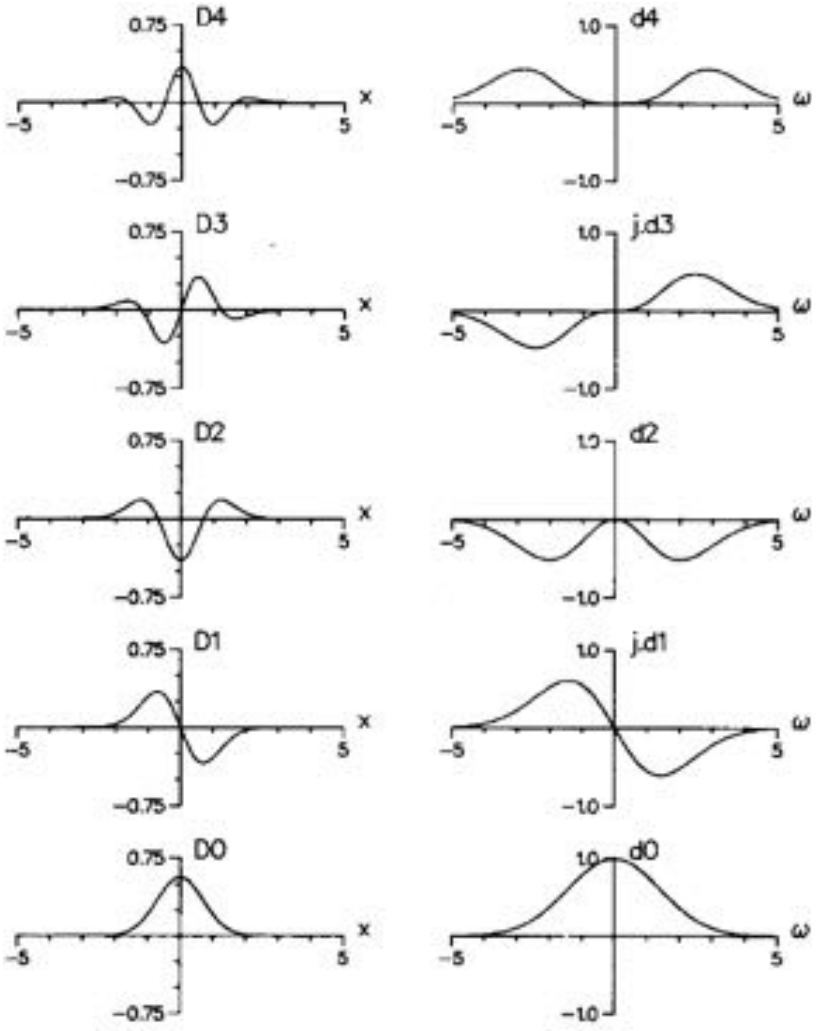


Figura 4.1. Funciones filtro en el dominio espacial y de frecuencia para $\sigma=1$ [13].

4.3. Transformada de Hermite Discreta [13]

La contraparte discreta de una ventana Gaussiana es una ventana binomial, esto es:

$$V^2(x)=\frac{1}{2^M}C_M^x \quad , \tag{4.33}$$

donde $C_M^x=\frac{x!}{(x-M)!\,M!}$ y $x=0,\ldots,M$. Los polinomios ortonormales discretos que están asociados con esta ventana se conocen como Polinomios de Krawtchouk:

$$G_n(x)=\frac{1}{\sqrt{C_M^2}}\sum_{k=0}^{n}(-1)^{n-k}C_{M-x}^{n-k}\cdot C_x^k \tag{4.34}$$

para $x,n=0,\ldots,M$.

Con valores grandes de M , la ventana binomial se reduce a ventana Gaussiana, más específicamente:

$$\lim_{M\to\infty}\frac{1}{2^M}C_M^{x+(M/2)}=\frac{1}{\sqrt{\pi}\sqrt{\frac{M}{2}}}\exp\left[-\left(\frac{x}{\sqrt{\frac{M}{2}}}\right)^2\right] \quad . \tag{4.35}$$

con $x=-(M/2),\ldots,M/2$.

Este limite convierte un Polinomio de Krawtchouk en un Polinomio de Hermite, es decir:

$$\lim_{M\to\infty}G_n\left(x+\frac{M}{2}\right)=\frac{1}{\sqrt{2^n n!}}H_n\left(\frac{x}{\sqrt{\frac{M}{2}}}\right) \tag{4.36}$$

con lo cual, la Transformada de Hermite de tamaño M se aproxima a la Transformada de Hermite analógica con $\sigma=\sqrt{M/2}$, por lo que las propiedades de la Transformada de Hermite discreta pueden ser precedidas con mucha precisión a partir de las propiedades de la transformada analógica. Con M par, las funciones filtro y patrón pueden centrarse en el origen desplazando la ventana binomial sobre $M/2$. Esto conduce a la siguiente definición de las funciones filtro de la Transformada Discreta de Hermite:

$$D_n(x)=G_n\left(\frac{M}{2}-x\right)\cdot V^2\left(\frac{M}{2}-x\right) \tag{4.37}$$

con $x=-(M/2),\ldots,M/2$. Esta función puede expresarse como:

$$D_n\left(\frac{M}{2}-x\right)=\frac{(-1)}{2^M\sqrt{C_M^n}}\Delta^n\left[C_M^x\cdot C_x^n\right] \quad , \tag{4.38}$$

donde:

$$(-1)^n \Delta^n L(x) = \sum_{k=0}^{n} (-1)^k C_n^k L(x+k) \tag{4.39}$$

es el operador diferencia de orden n . Tomando la Transformada z de esta función filtro, queda:

$$d_n(z) = \sum_{x=-M/2}^{M/2} D_n(x) z^{-x} = z^{-M/2} \sqrt{C_M^n} \left(\frac{1-z}{2} \right)^n \left(\frac{1+z}{2} \right)^{M-n} \tag{4.40}$$

que expresado en frecuencia angular es:

$$d_n(e^{-j\omega}) = \sqrt{C_M^2} \left(j \sin \frac{\omega}{2} \right)^n \left(\cos \frac{\omega}{2} \right)^{M-n} \tag{4.41}$$

para $n = 0, \ldots, M$.

Los filtros anteriores tienen la importante ventaja práctica de poder realizarse en cascada con los filtros simples $z^{-1}(1+z)^2$, $z^{-1}(1-z)(1+z)$ y $z^{-1}(1-z)^2$, con los respectivos kernels [1 2 1], [-1 0 1] y [1 -2 1].

Las ventanas Gaussianas en dos dimensiones tienen la propiedad única de ser separables y rotacionalmente simétricas, haciendo que la funciones filtro correspondientes sean separables de forma cartesiana y polar.

4.4. Algoritmo Rápido de la Transformada Discreta de Hermite 2D[3]

Se redefinen las funciones filtro $d_n(z)$ descritas por la ecuación (4.40) como los vectores $d_{N,k}$ de $N+1$ dimensiones, tales que:

$$d_{N,k} = 2^{N-k} \begin{bmatrix} d_{N,k}(N) \\ d_{N,k}(N-1) \\ \vdots \\ d_{N,k}(0) \end{bmatrix} , \quad k = 0,1,\ldots,N \quad . \tag{4.42}$$

En una dimensión, las estimaciones de las derivadas pueden expresarse como:

$$\hat{f}^{(k)} = \frac{1}{2^{N-k}} d_{N,k}^T f_o \quad , \quad k = 0,1,\ldots,N \tag{4.43}$$

donde f_o es un vector de $N+1$ dimensiones y las derivadas son estimadas en el centro de la ventana. Se define la matriz [P_N] de tamaño $N+1$ por $N+1$ cuyo $k+1$ ° renglón es $d_{N,k}^T$. El conjunto de ecuaciones (4.43) se escribe matricialmente de la forma:

$$F=(F_{0,}F_{1,}\ldots,F_N)^T=[P_N]f_o \quad , \tag{4.44}$$

donde $F_k=2^{N-k}\hat{f}^{(k)}$. Las matrices [P_N] para $N=1,2,3$ son:

$$\underset{N=1}{\begin{bmatrix}1 & 1\\ -1 & 1\end{bmatrix}} \quad , \quad \underset{N=2}{\begin{bmatrix}1 & 2 & 1\\ -1 & 0 & 1\\ 1 & -2 & 1\end{bmatrix}} \quad \text{y} \quad \underset{N=3}{\begin{bmatrix}1 & 3 & 3 & 1\\ -1 & -1 & 1 & 1\\ 1 & -1 & -1 & 1\\ -1 & 3 & -3 & 1\end{bmatrix}}$$

En dos dimensiones las derivadas parciales son estimadas usando el operador de una dimensión en cada dirección. La matriz [F] de derivadas parciales puede escribirse como:

$$F=\begin{bmatrix}F_{00} & F_{01} & \cdots & F_{0N}\\ F_{10} & F_{11} & \cdots & F_{1N}\\ \vdots & & & \\ F_{N0} & F_{N1} & \cdots & F_{NN}\end{bmatrix}=[P_N][f_o][P_N]^T \tag{4.45}$$

donde [f_o] es la matriz de datos dentro de una ventana de $N+1$ por $N+1$. los elementos F_{kl} son derivadas parciales estimadas con el factor de escala 2^{2N-k-l} . Las derivadas parciales del mismo orden tienen el mismo factor de escala y las derivadas direccionales pueden calcularse sin tomar en cuenta la escala. Cada elemento de F para cada ventana también se calcula por la correlación de la imagen con el kernel $d_{N,k}d_{N,l}^T$.

Para estimar derivadas de múltiple orden se pueden implementar un conjunto de filtros discretos por varios algoritmos:

a) Hacer convoluciones con un conjunto de máscaras en dos dimensiones
b) Hacer multiplicaciones matriciales con cada dato de la ventana.
c) Hacer convoluciones con un conjunto de máscaras unidimensionales en ambas direcciones.
d) Repetir convoluciones con dos secuencias [1,1] y [1,-1] en ambas direcciones.

Este último puede implementarse con un menor número de operaciones aunque la aplicación directa requiere mucha memoria adicional para los resultados intermedios. La intención de usar un algoritmo rápido es realizar convoluciones repetidas por las dos secuencias [1,1] y [1,-1] con poca memoria adicional para los resultados intermedios.

En una dimensión, la función de transferencia del filtro que estima la k-ésima derivada escrita como:

$$D_{N,k}(z)=\frac{z^{N/2}}{2^{N-k}}(1+z^{-1})^{N-k}(1-z^{-1})^{k} \qquad (4.46)$$

puede ser implementada en cascada como dos filtros de primer orden cuyas funciones de transferencia son ($1+z^{-1}$) y ($1-z^{-1}$), lo que equivale a repetir convoluciones con las secuencias [1,1] y [1,-1].

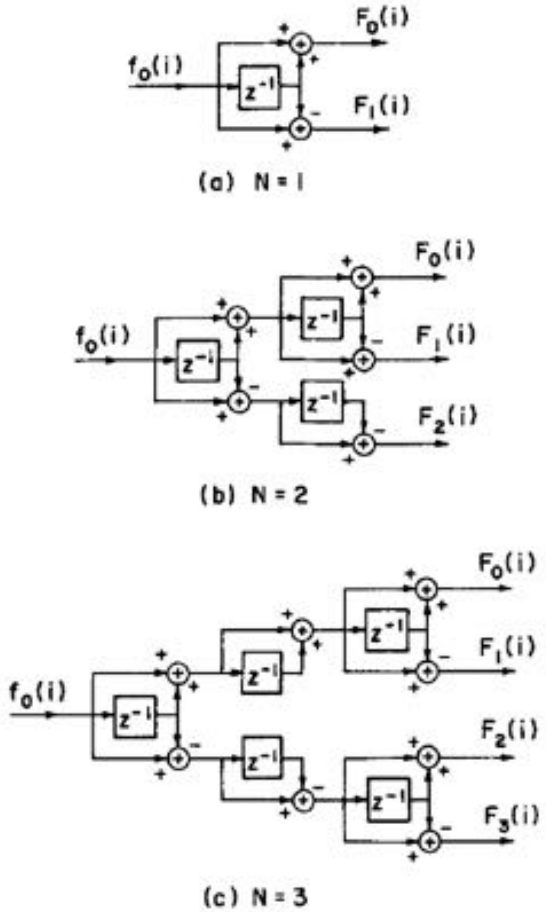


Figura 4.2. Algoritmos de la Transformada Rápida de orden múltiple en una dimensión. (a) $N=1$. (b) $N=2$. (c) $N=3$ [3].

El algoritmo rápido para la estimación de derivadas de orden múltiple se obtiene implementando en cascada cada filtro en paralelo. Debido a que las funciones de transferencia de los filtros de estimación de derivadas tienen muchos términos comunes, las operaciones redundantes pueden eliminarse compartiendo los resultados intermedios. El número de operaciones redundantes aumenta a medida que N se hace más grande. En la Figura 4.2. se muestra el algoritmo rápido para $N=1,2\ y\ 3$ en una dimensión, donde cada filtro de primer orden se implementa con una operación de sustracción y una de adición.

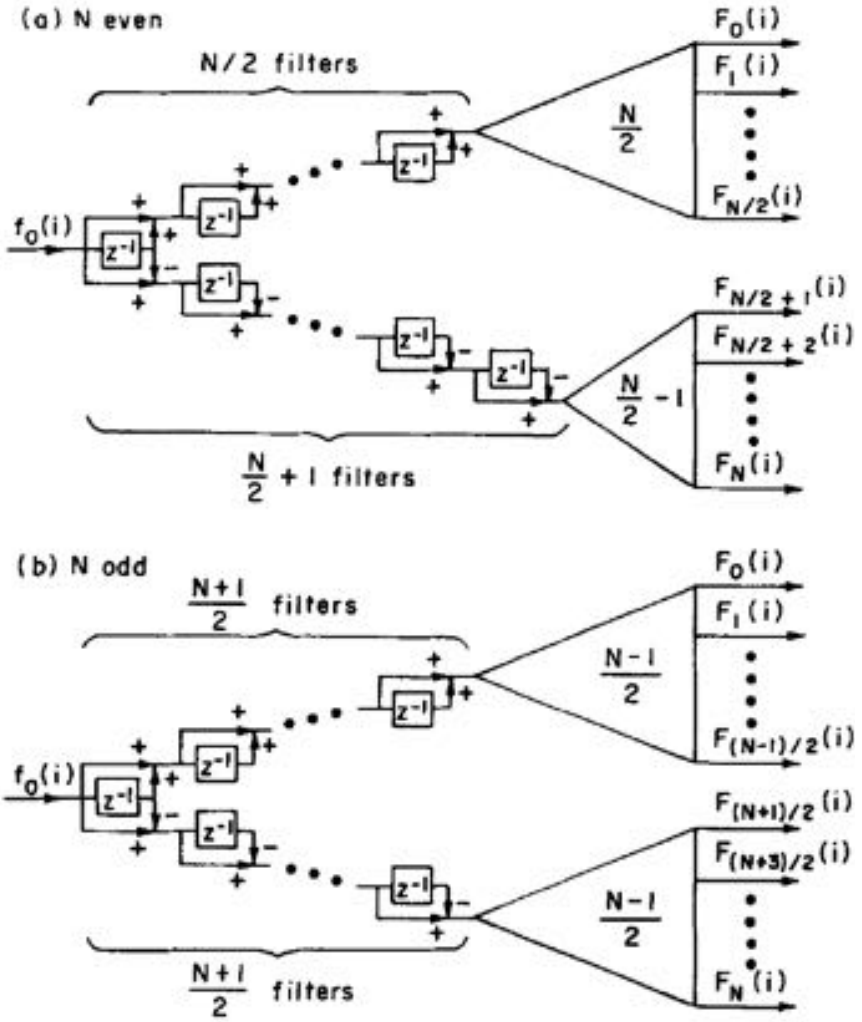


Figura 4.3. Algoritmos de la Transformada Rápida de orden múltiple de orden N . (a) N par. (b) N impar [3].

Ya que la función de transferencia de salida $F_k(i)$ es $(1-z^{-1})^k(1+z^{-1})^{N-k}$, se modifica la salida como:

$$\hat{f}^{(k)} = \frac{1}{2^{N-k}} F_k(i + N/2) \quad . \tag{4.47}$$

El algoritmo rápido para valores grandes de N se compone de algoritmos rápidos para tamaños más pequeños y series de filtros de primer orden como en la Figura 4.3. Las ecuaciones para el número requerido de operaciones suma y resta se pueden expresar en formas recursivas. Cuando $N+1$ es una potencia de 2, el número de sumas por elemento es dado por $(N+1)\log_2(N+1)$ porque la estructura se divide en dos subestructuras y cada una se divide en dos más pequeñas y así sucesivamente. Para $N+1$ impar el entero más pequeño, mayor o igual que $(N+1)\log_2(N+1)$ dará el número exacto de sumas a realizar.

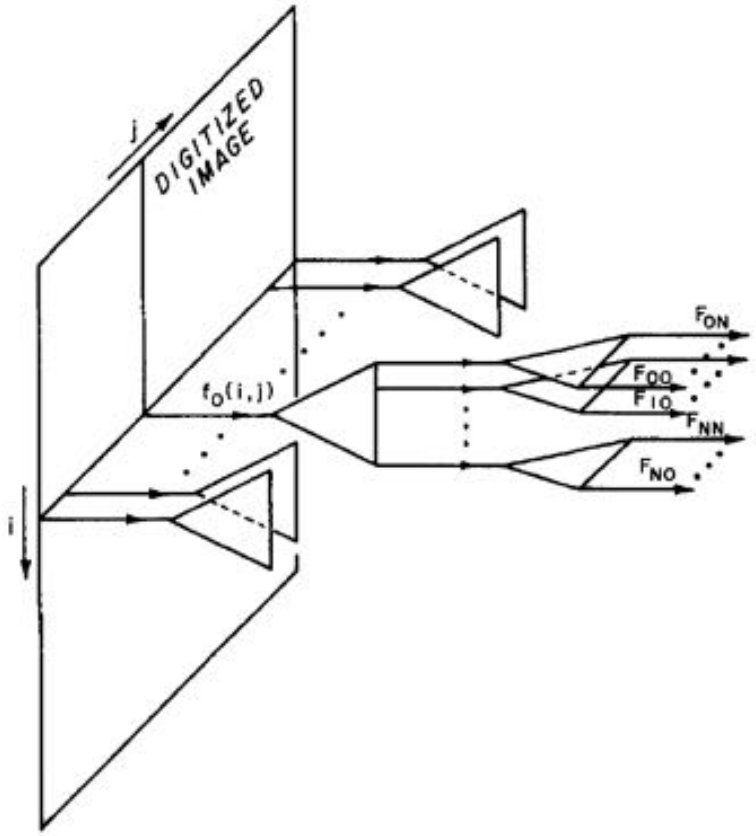


Figura 4.4. Algoritmo de la Transformada Rápida de orden múltiple en dos dimensiones [3].

En dos dimensiones los filtros son separables, las derivadas parciales pueden estimarse aplicando el algoritmo rápido en una dimensión a cada columna (o renglón) de la imagen de entrada y después a cada renglón (o columna) del resultado. Por la propiedad de recursividad del algoritmo, la entrada debe

alimentarse continuamente por el conjunto de filtros. En dos dimensiones no es posible alimentar continuamente la imagen de entrada en ambas direcciones a menos que utilicemos un algoritmo paralelo para cada columna o cada fila. En la Figura 4.4. se muestra el algoritmo rápido en dos dimensiones, en donde cada triángulo representa al algoritmo rápido de una dimensión de tamaño N . Los filtros columna son paralelos a todas las columnas de la imagen de entrada y los filtros renglón son paralelos a las salidas $N=1$ de los filtros columna. La estimación de derivadas parciales es obtenida por la modificación:

$$\hat{f}^{(k,l)}(i,j)=\frac{1}{2^{2N-kl}}F_{kl}(i+N/2,\, j+N/2) \quad . \tag{4.48}$$

Como en el caso de una dimensión, cuando no usamos todas las derivadas parciales $(N+1)^2$ como características locales, podemos omitir la parte innecesaria del algoritmo. Por ejemplo, para el algoritmo de estimaciones de las derivadas parciales de primer y segundo orden con $N=2$ en dos dimensiones, mostrado en la Figura 4.5., se estiman 5 derivadas y se requieren únicamente 12 operaciones por píxel con 3 filas guardadas en memoria.

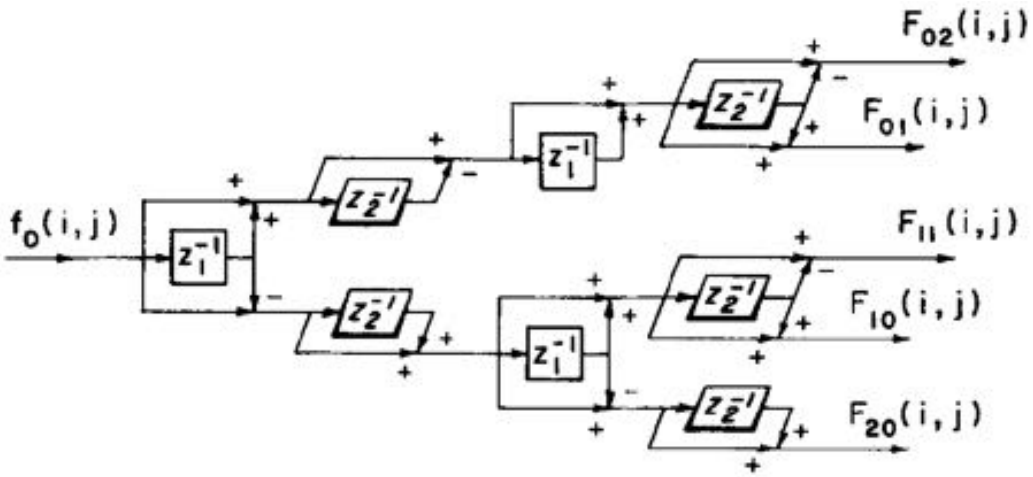


Figura 4.5. Algoritmo para la estimación de derivadas parciales en dos dimensiones de primer y segundo orden con $N=2$ [3].

4.5. Rotación de la Transformada de Hermite [13], [14]

Si las ventanas Gaussianas en dos dimensiones son separables y cuadradas, sus funciones filtro también son separables y por lo tanto pueden implementarse eficientemente. La Transformada de Fourier de $D_m(x)D_{n-m}(y)$, expresada en las coordenadas polares $w_x=w\cos\theta$ y $w_y=w\sin\theta$, es:

$$d_m(w_x)\,d_{n-m}(w_y) = g_{m,n-m}(\theta)\cdot d_n(w) \quad , \tag{4.49}$$

donde $d_n(w)$ es la Transformada de Fourier de la función filtro de Hermite unidimensional $D_n(r)$, con la coordenada radial r , y

$$g_{m,n-m}(\theta) = \sqrt{\frac{n!}{m!(n-m)!}}\cos^m\theta\cdot\sin^{n-m}\theta \quad , \tag{4.50}$$

expresa la selectividad direccional del filtro. Por lo tanto, los filtros de orden creciente n analizan sucesivamente las frecuencias radiales más altas, es decir, las resoluciones espaciales más altas. Los filtros del mismo orden n y diferente índice m distinguen entre diferentes orientaciones en la imagen. La relación con la función angular de la proyección de los coeficientes de la Transformada Polinomial 2D a 1D es:

$$h_{n,\theta}(l, k-l) = \delta_{n,k}\cdot g_{l,k-l}(\theta) \quad . \tag{4.51}$$

Aplicando lo anterior a la ecuación (4.26), los coeficientes de la Transformada de Hermite 2D de la imagen $L(x, y)$ se proyectan sobre la dirección θ como:

$$L_{m,n-m,\theta} = \begin{cases} \sum_{k=0}^{n} L_{k,n-k}\cdot g_{k,n-k}(\theta), & m=0 \\ 0, & m>0 \end{cases} \quad , \tag{4.52}$$

donde $L_{k,n-k}$ son los coeficientes de la Transformada de Hermite 2D y $L_{m,n-m,\theta}$ son los coeficientes rotados de la Transformada de Hermite.

La medida de fuerza de la orientación a lo largo de la dirección θ por el cuadrado de la salida de una cuadratura par de filtros paso banda direccionados en el ángulo θ , es llamada energía orientada $E(\theta)$.

Usando la n-ésima derivada de una Gaussiana y su Transformada de Hilbert como filtros paso banda, se tiene la energía orientada como:

$$E_n(\theta) = [G_n^\theta]^2 + [H_n^\theta]^2 \quad . \tag{4.53}$$

Escribiendo G_n^θ y H_n^θ como una suma de las salidas de los filtros base multiplicadas por funciones de interpolación, simplifica a una serie de Fourier en

ángulo, donde sólo se presentan frecuencias base debido a la operación cuadrática:

$$E_n = C_1 + C_2 \cos(2\theta) + C_3 \sin(2\theta) + [\textit{terminos de orden superior} \ldots] \quad . \qquad (4.54)$$

Se utiliza el término de frecuencia más bajo para aproximar la dirección θ_d de la orientación dominante (la orientación que maximiza $E_n(\theta)$)

$$\theta_d = \frac{arg[C_2, C_3]}{2} \quad . \qquad (4.55)$$

Esta aproximación es exacta sólo si se presenta una orientación localmente.

Capítulo 5
Flujo Óptico [15]

Un problema fundamental en el procesamiento de secuencias de imágenes es la medición del flujo óptico (o velocidad de la imagen) cuyo objetivo es calcular una aproximación del campo de movimiento 2D a las velocidades 3D de los puntos de superficie sobre la la imagen a partir de patrones espaciotemporales de la intensidad de la imagen. Las técnicas actuales requieren que los errores relativos en el flujo óptico sean menores de 10% y las estimaciones exactas del campo de movimiento bidimensional son generalmente inaccesibles. Lo que sugiere que sólo se puede extraer información cualitativa.

Conceptualmente, muchas de las técnicas que se emplean, pueden ser descritas en 3 etapas de procesamiento:

1. Suavizar con filtros de paso bajas / paso bandas para extraer la estructura de interés y mejorar la relación señal-ruido.
2. Obtener mediciones básicas.
3. Integrar estas medidas para producir un campo de flujo.

Las técnicas pueden clasificarse en 4 tipos de métodos:

- Diferenciales.
- Basados en coincidencia entre regiones.
- Basados en energía.
- Basados en Fase.

5.1. Métodos Diferenciales

Los métodos diferenciales calculan la velocidad a partir de derivadas espaciotemporales de la intensidad de la imagen o sus versiones filtradas. Inicialmente se utilizaron derivadas de primer orden y se basaron en el movimiento de la imagen [16], [17], [18].

Los métodos con derivadas de segundo orden restringen la velocidad en dos dimensiones [18], [19], [20], si la intensidad es unidimensional, entonces debido a la sensibilidad de la diferenciación numérica, las derivadas de segundo orden no pueden medirse con suficiente precisión para determinar la componente tangencial de la velocidad. Por lo que se suele suponer que las estimaciones de los métodos de segundo orden son escasas y menos precisas que las estimaciones de los métodos de primer orden [15].

Una manera típica de superar los problemas mal planteados de los métodos de flujo óptico diferencial consiste en el uso de técnicas y presunciones de suavizado: Comúnmente se suaviza la secuencia de imágenes antes de la diferenciación con el fin de eliminar el ruido y estabilizar el proceso de diferenciación.

Muchos métodos diferenciales para el flujo óptico se basan en el supuesto de que los valores de gris de los objetos de la imagen actual en las siguientes imágenes de la secuencia no cambian, esto es:

$$f(x+u, y+v, t+1)=f(x, y, t) \quad , \tag{5.1}$$

donde (x, y) denota posición dentro del dominio rectangular de la imagen Ω, $t\in[0,T]$ denota tiempo y el campo de desplazamiento $(u, v)^{\top}(x, y, t)$ es llamado flujo óptico [4].

5.1.1. El Suavizado en los Métodos Diferenciales [4]

Todas estrategia de suavizado espacial puede extenderse al dominio temporal. Es común suavizar la secuencia de imagen antes de la diferenciación. Para pequeños desplazamientos se le puede realizar a la ecuación (5.1) una expansión de Taylor de primer orden, produciendo la restricción de flujo óptico:

$$f_x u+f_y v+f_t=0 \quad , \tag{5.2}$$

donde los subíndices denotan derivaciones parciales. Esta ecuación no es suficiente para calcular de forma única las dos incógnitas u y v (problema de la apertura, Figura 5.1.). Para gradientes de la imagen que no varían, sólo es posible determinar el componente de flujo paralelo a $\nabla f=(f_x u, f_y)^T$, esto es llamado flujo normal y esta dado por:

$$w_n=\frac{-f_t}{|\nabla f|}\frac{\nabla f}{|\nabla f|} \quad . \tag{5.3}$$

Para hacer frente al problema de la apertura, Lucas-Kanade [21] propusieron asumir que el vector de flujo óptico desconocido es constante dentro de algún vecindario de tamaño ρ . En este caso, es posible determinar las dos constantes u y v en algún lugar (x, y, t) de una ponderación de mínimos cuadrados ajustada por la minimización de la función:

$$E_{LK}(u, v)=K_{\rho}*((f_x u+f_y v+f_t)^2) \quad , \tag{5.4}$$

donde $*$ denota convolución y K_ρ es una Gaussiana en 2D con desviación estándar ρ .

Aquí ρ sirve como una integración a escala sobre la cual la principal contribución del ajuste de mínimos cuadrados es calculada.

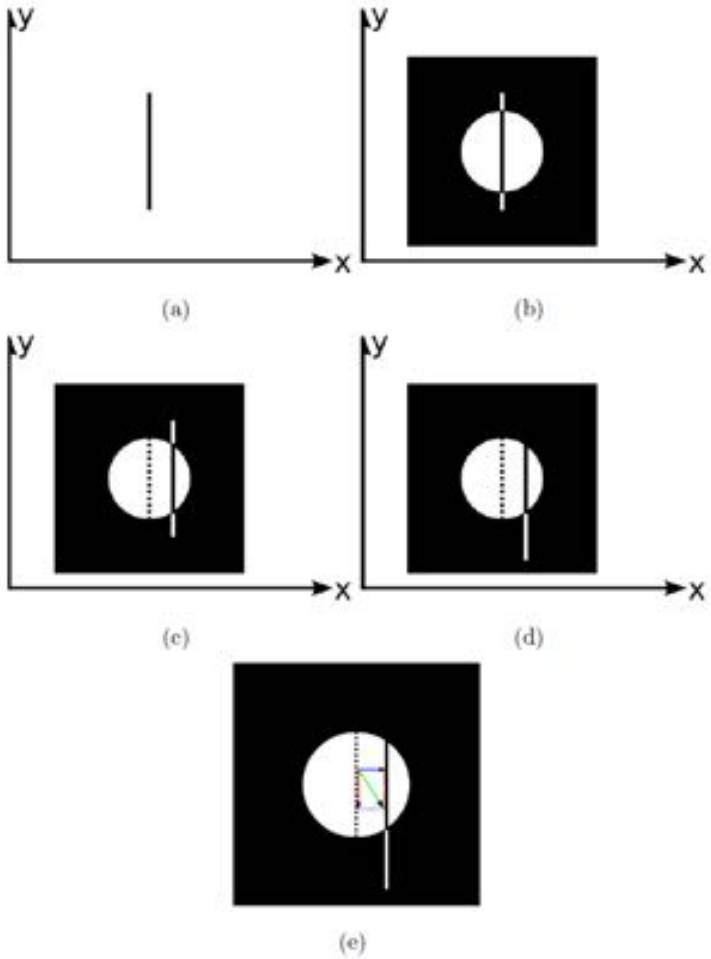


Figura 5.1. Problema de la apertura. (a) Una línea recta en el plano XY. (b) Campo visual restringido localmente por una ventana circular. (c) Desplazamiento local detectado a través de la ventana. (d) Únicamente se detecta el desplazamiento horizontal a través de la ventana. (e) Desplazamiento actual en (d) [1].

Un mínimo (u, v) de E_{LK} satisface $\partial_u E_{LK}$ y $\partial_v E_{LK}$, esto da el sistema lineal:

$$\begin{pmatrix} K_\rho * (f_x^2) & K_\rho * (f_x f_y) \\ K_\rho * (f_x f_y) & K_\rho * (f_y^2) \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} -K_\rho * (f_x f_t) \\ -K_\rho * (f_y f_t) \end{pmatrix} \tag{5.5}$$

que puede resolverse siempre que su matriz sea invertible. Este no es el caso en las regiones planas donde el gradiente de la imagen se desvanece. En algunas otras regiones, el eigenvalor más pequeño de la matriz del sistema puede estar cerca de 0, de manera que el problema de la apertura permanece presente y los datos no permiten determinar fiablemente el flujo óptico completo, resultando en campos de flujo no denso. Esto constituye el inconveniente más severo de los métodos de gradiente local: en las aplicaciones de visión computacional que requieren estimaciones de flujo denso se necesita una subsecuente interpolación pero por otro lado se desean eigenvalores pequeños como una medida de confianza que caracteriza la fiabilidad de la estimación

Un valor suficientemente grande de ρ es muy exitoso en hacer que el método Lucas y Kanade sea robusto bajo ruido. Para obtener estimaciones de flujo denso se puede integrar la restricción de flujo óptico en un sistema de regularización. Horn-Schunck [17] iniciaron esta clase de métodos diferenciales globales. Determinan las funciones desconocidas $u(x,y,t)$ y $v(x,y,t)$ como los minimizadores del funcional energía global

$$E_{HS}(u,v)=\int_{\Omega}\left((f_x u+f_y v+f_t)^2+\alpha(|\nabla u|^2+|\nabla v|^2)\right)dx\,dy \tag{5.6}$$

donde el peso del suavizado $\alpha>0$ sirve como un parámetro de regularización, grandes valores para α generan una fuerte penalización de grandes gradientes de flujo y conducen a campos de flujo más suaves.

Minimizar este funcional se reduce a resolver sus correspondientes ecuaciones de Euler-Lagrange, dadas por:

$$0=\Delta u-\frac{1}{\alpha}(f_x^2 u+f_x f_y v+f_x f_t) \tag{5.7}$$

$$0=\Delta v-\frac{1}{\alpha}(f_x f_y u+f_y^2 v+f_y f_t) \tag{5.8}$$

con condiciones de contorno reflectantes. Δ denota el operador espacial de Laplace:

$$\Delta=\partial_{xx}+\partial_{yy} \quad . \tag{5.9}$$

La solución de estas ecuaciones de difusión-reacción no sólo es única [22] , también se beneficia del efecto de rellenado: En ubicaciones con $|\nabla f|\approx 0$, no es posible una estimación de flujo local fiable, pero la regularización $|\nabla u|^2+|\nabla v|^2$

ocupa información del vecindario. Esto da lugar a campos de flujo denso y hace que los pasos de interpolación subsiguientes sean obsoletos, una clara ventaja sobre los métodos locales.

Sin embargo, para tales métodos diferenciales globales no se dispone de buenas medidas de confianza que ayuden a determinar los lugares en los que los cálculos son más fiables. También se ha observado que pueden ser más sensibles al ruido que los métodos diferenciales locales [15], [23].

5.1.2. Combinación de Métodos Locales y Globales [4]

Los métodos diferenciales pueden clasificarse en métodos locales como la técnica de Lucas-Kanade [21] y en métodos globales como la aproximación de Horn-Schunck [17]. A menudo, los métodos locales son más robustos bajo ruido, mientras que las técnicas globales producen campos de flujo denso. En consecuencia, es deseable combinar los diferentes efectos de los métodos locales y globales para diseñar nuevos enfoques que combinen la alta robustez de los métodos locales con la densidad total de las técnicas globales y por tanto, no hay necesidad de una etapa posterior de procesamiento para interpolar datos.

Con la notación:

$$w=(u,v,1)^{\top} \quad , \tag{5.10}$$

$$|\nabla w|^2=|\nabla u|^2+|\nabla v|^2 \quad , \tag{5.11}$$

$$\nabla_3 f=(f_x+f_y+f_t)^{\top} \quad , \tag{5.12}$$

y

$$J_{\rho}(\nabla_3 f)=K_{\rho}*(\nabla_3 f\,\nabla_3 f^{\top}) \tag{5.13}$$

el método Lucas-Kanade se minimiza de forma cuadrática como:

$$E_{LK}(w)=w^{\top}J_{\rho}(\nabla_3 f)w \quad , \tag{5.14}$$

mientras que el método Horn-Schunck como:

$$E_{HS}(w)=\int_{\Omega}(w^{\top}J_0(\nabla_3 f)w+\alpha|\nabla w|^2)\,dx\,dy \quad . \tag{5.15}$$

Esta terminología sugiere una forma natural de extender el funcional de Horn-Schunck al funcional de la combinación local y global. Se sustituye la matriz

$J_0(\nabla_3 f)$ por el tensor estructural $J_\rho(\nabla_3 f)$ con una escala de integración $\rho>0$:

$$E_{CLG}(w)=\int_\Omega (w^\top J_\rho(\nabla_3 f)w+\alpha|\nabla w|^2)\,dx\,dy \quad . \tag{5.16}$$

Su minimización de campo de flujo (u,v) satisface las ecuaciones de Euler-Lagrange:

$$0=\Delta u-\frac{1}{\alpha}(K_\rho*(f_x^2)u+K_\rho*(f_x f_y)v+K_\rho*(f_x f_t)) \tag{5.17}$$

y

$$0=\Delta u-\frac{1}{\alpha}(K_\rho*(f_x f_y)u+K_\rho*(f_y^2)v+K_\rho*(f_y f_t)) \quad . \tag{5.18}$$

La aproximación anterior sólo usa operadores de suavizado espacial. En general, una aproximación espaciotemporal da mejores resultados que debido a las propiedades adicionales de eliminación de ruido a lo largo del tiempo. Una versión espaciotemporal del funcional en (5.16) está dada como:

$$E_{CLG3}(w)=\int_\Omega (w^\top J_\rho(\nabla_3 f)w+\alpha|\nabla_3 w|^2)\,dx\,dy\,dt \tag{5.19}$$

donde las convoluciones con Gaussianas son espaciotemporales y

$$|\nabla_3 w|^2=|\nabla_3 u|^2+|\nabla_3 v|^2 \quad . \tag{5.19}$$

Devido a los diferentes roles del espacio y el tiempo, las Gaussianas espaciotemporales pueden tener diferentes desviaciones estándar en ambas direcciones. Denotando J_{nm} como el componente (n,m) del tensor estructural $J_\rho(\nabla_3 f)$, las ecuaciones de Euler-Lagrange para el funcional descrito en la ecuación (5.19) están dadas por:

$$0=\Delta_3 u-\frac{1}{\alpha}(J_{11}u+J_{12}v+J_{13}) \tag{5.20}$$

y

$$0=\Delta_3 v-\frac{1}{\alpha}(J_{21}u+J_{22}v+J_{23}) \quad . \tag{5.21}$$

y el Laplaciano Δ es reemplazado por el Laplaciano espaciotemporal:

$$\Delta_3=\partial_{xx}+\partial_{yy}+\partial_{tt} \quad . \tag{5.22}$$

5.1.3. Optimización con Funciones Estadísticas Robustas [4]

Los enfoques anteriores de los métodos Lucas-Kanade y Horn-Schunck son lineales y basados en optimización cuadrática. Estos pueden reemplazarse por problemas de optimización no cuadráticos que conducen a métodos no lineales. Desde un punto de vista estadístico, esto puede considerarse como la aplicación de métodos que usan estadísticas robustas donde los valores extremos son penalizados de forma menos severa que los cuadráticos. En general, los métodos no lineales dan mejores resultados en lugares con discontinuidades de flujo.

Para hacer el enfoque más robusto contra los valores atípicos en los datos y el término de suavidad, se minimiza el siguiente funcional:

$$E_{CLG3-N}(w)=\int_{\Omega x[0,T]}(\Psi_1(w^\top J_\rho(\nabla_3 f)w)+\alpha\Psi_2(|\nabla_3 w|^2))dx\,dy\,dt \tag{5.23}$$

Donde las funciones $\Psi_1(s^2)$ y $\Psi_2(s^2)$ son penalizaciones no cuadráticas. Para garantizar que esté bien planteado lo que resta del problema, sólo se ocupan penalizaciones convexas en s ,en particular la siguiente función propuesta por Charbonnier et al. [24] :

$$\Psi_i(s^2)=2\beta_i^2\sqrt{1+\frac{s^2}{\beta_i^2}}\quad,\quad i\in 1,2\quad, \tag{5.24}$$

donde β_i es un parámetro de escala. En la Figura 5.2. se muestra esta función cuando $\beta=0.1$.

Bajo algunos requisitos técnicos, la elección de penalizaciones convexas garantiza una solución única del problema de minimización y permite construir algoritmos globales convergentes sencillos.

Las ecuaciones de Euler-Lagrange del funcional de energía (ecuación 5.23) están dadas por:

$$0=div(\Psi'_2(|\nabla_3 w|^2)\nabla_3 u)-\frac{1}{\alpha}\Psi'_1(w^\top J_\rho(\nabla_3 f)w)(J_{11}u+J_{12}v+J_{13}) \tag{5.25}$$

y

$$0=div(\Psi'_2(|\nabla_3 w|^2)\nabla_3 u)-\frac{1}{\alpha}\Psi'_1(w^\top J_\rho(\nabla_3 f)w)(J_{21}u+J_{22}v+J_{23}) \tag{5.26}$$

con:

$$\Psi_i(s^2)=\frac{1}{\sqrt{1+\frac{s^2}{\beta_i^2}}} \quad , \quad i\in 1,2 \tag{5.27}$$

Para valores muy grandes de β_i , el caso no lineal se convierte en lineal ya que $\Psi'_i(s^2)\approx 1$.

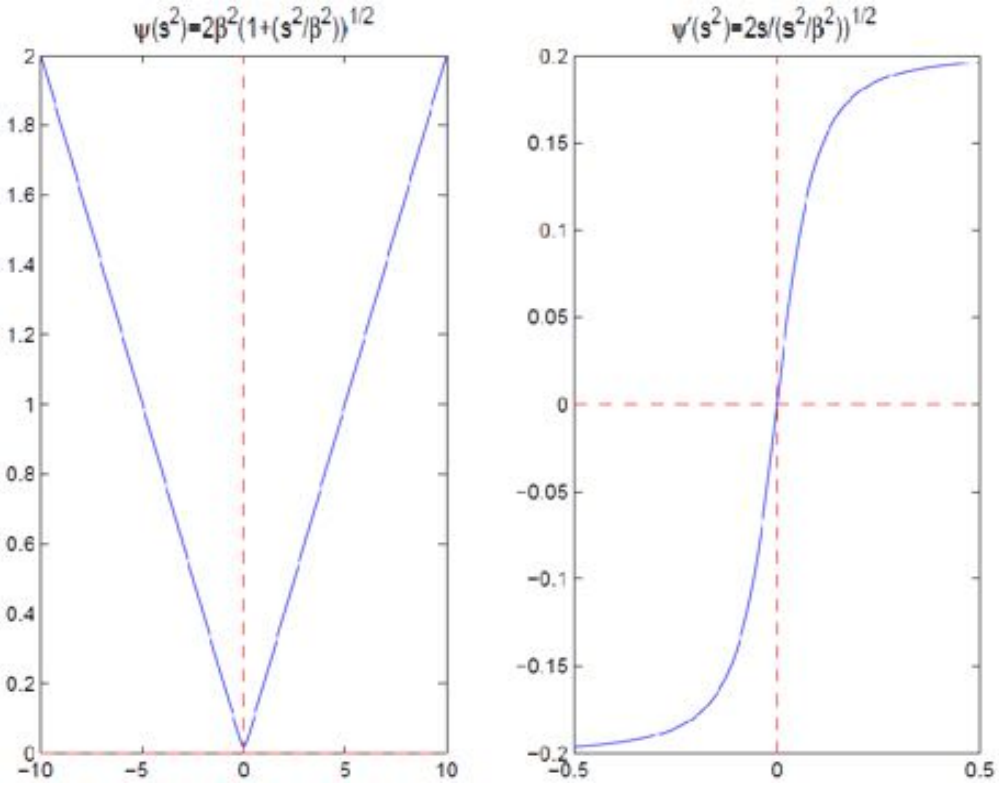


Figura 5.2. Función de optimización propuesta por Charbonnier et al.: $\Psi(s^2)=2\beta^2\sqrt{\left(1+\frac{s^2}{\beta^2}\right)}$, con parámetro de escala $\beta=0.1$ [1].

5.1.4. Multiescala [4], [1]

Todas las variantes del método de combinación local y global consideradas hasta ahora se basan en una linealización que asume el valor de gris como constante. En consecuencia, se requiere que u y v sean relativamente pequeños para que la linealización se mantenga, algo que no puede ser garantizado para secuencias arbitrarias. Sin embargo, existen estrategias que permiten superar esta

limitación, las técnicas denominadas de enfoque multiescala (o multiresolución) [25] - [28] que calculan en incrementos el campo de flujo óptico con base en la estrategia de refinar paso a paso la resolución a partir de una escala burda. En particular, para los funcionales de energía con un mínimo global, tal procedimiento sólo conduce a acelerar la convergencia, ya que el resultado no cambia. En su lugar, el movimiento en una escala grande se utiliza para deformar (corregir) la secuencia original antes de pasar al siguiente nivel más pequeño. Dando como resultado una jerarquía de problemas modificados que sólo requieren calcular campos de desplazamiento pequeños, los llamados incrementos de movimiento. Por lo tanto, el campo de desplazamiento final que es obtenido por la suma de todos los incrementos de movimiento es mucho más preciso con respecto a la linealización de la suposición del valor de gris constante.

Sea δw^m el incremento de movimiento en la escala m , donde $m=0$ es la escala más grande con la inicialización $w^0=(0,0,0)^\top$, δw^m se obtiene optimizando el siguiente funcional de energía espaciotemporal:

$$E^m_{CLG3-N}(\delta w^m)=\int_{\Omega x[0,T]}\Big(\Psi_1\big(\delta w^{m\top} J_\rho(\nabla_3 f(X+w^m))\,\delta w^m\big) +\alpha\,\Psi_2\big(\left|\nabla_3(w^m+\delta w^m)\right|^2\big)\Big)dX \quad , \tag{5.28}$$

donde Ω es el dominio rectangular de la imagen, $[0,T]$ denota tiempo, $w^{m+1}=w^m+\delta w^m$ y $X=(x,y,t)$.

Al deformar la secuencia original sólo se afecta a los términos de los datos. Dado que la suposición del suavizado se aplica al campo de flujo completo, $w^m+\delta w^m$ se utiliza como argumento de la penalización.

Papenberg et al. [29] proponen el siguiente funcional que combina restricción para la intensidad constante [30], restricción para el gradiente constante [20], restricción para el suavizado espaciotemporal y enfoque multiescala:

$$E_{warp}(u,v)=\int_{\Omega x[0,T]}\Psi\big(\left|f(X+w)-f(X)\right|^2+\gamma\left|\nabla f(X+w)-\nabla f(X)\right|^2\big)dX +\alpha\int_{\Omega x[0,T]}\Psi\big(\left|\nabla_3 u\right|^2+\left|\nabla_3 v\right|^2\big)dX \tag{5.29}$$

donde $\Psi(s^2)=\sqrt{(s^2+\varepsilon^2)}$ es la regularización isotrópica de flujo, que regulariza la variación total asegurando su diferenciabilidad con ε , y γ es un peso entre la restricción de la intensidad constante y la restricción del gradiente constante.

5.2. Cálculo de Flujo Óptico usando la Transformada de Hermite

El cálculo de la estimación de movimiento usando la Transformada de Hermite ya se ha implementado [31], [32]. El desarrollado por Moya-Albor [1] ha demostrado ser un método robusto al ruido, un factor muy importante cuando se trabaja con imágenes de ultrasonido [2].

El método consiste en dadas dos imágenes consecutivas $L(x, y, t)$ y $L(x+u, y+v, t+1)$, se obtiene la descomposición polinomial de las imágenes utilizando los coeficientes rotados de la Transformada de Hermite, presentando las características locales de las imágenes desde una aproximación perceptiva en un sistema de multiresolución. El método incluye elementos encontrados en los últimos avances en métodos diferenciales [33], [4], [29] que permiten obtener un flujo óptico preciso [1].

5.2.1. Suposición de Constantes [1]

Con la imagen descompuesta polinomicamente, se toma como restricción local el coeficiente de orden cero de la Transformada de Hermite que representa la intensidad de la imagen y las derivadas direccionales de orden superior que están contenidas implícitamente en los coeficientes de Hermite orientados, de forma similar al sistema de visión humano. El uso de las derivadas de una función Gaussiana permite incorporar información de los píxeles vecinos para determinar las restricciones locales de la imagen, similar a los métodos de diferencia locales, como en la ecuación (5.14), que tienen la característica de ser robustos al ruido. Esto se incluye en un funcional diferencial global con el fin de obtener flujo de campo denso como en [17], [18].

5.2.1.1. Restricción para la Intensidad Constante [1]

El coeficiente de orden 0 de la Transformada de Hermite L_{00} (por simplicidad L_0) contiene una versión suavizada de la imagen original, por lo que puede usarse para definir la restricción para la intensidad constante:

$$L_0(x, y, t) = L_0(x+u, y+v, t+1) \quad , \tag{5.30}$$

donde el suavizado del coeficiente permite eliminar cualquier componente de ruido de alta frecuencia.

5.2.1.2. Restricción para los Coeficientes de Hermite Orientados [1]

Un cambio de intensidad global de las imágenes en dos tiempos consecutivos infringe la restricción para la intensidad constante, pero no el gradiente de las imágenes, que sólo está afectado en su magnitud y no en su dirección. Además la restricción para el gradiente constante considera derivadas de orden superior para formular suposiciones de ser constante como la del Hessiano, del Laplaciano y de la norma y el determinante del Hessiano. También se pueden considerar derivadas de orden superior a dos por lo que para este método se incluyen los coeficientes de Hermite orientados $l_{n0}(\theta)$ (por simplicidad $l_{n\theta}$) hasta el orden N para ocupar varios tipos de movimientos, como los de traslación y rotación. Esto permite incluir derivadas superiores simplemente cambiando el orden máximo de N en la Transformada de Hermite.

De esta manera la restricción para los coeficientes de Hermite orientados se define como:

$$\sum_{n=1}^{N} l_{n\theta}(x, y, t) = \sum_{n=1}^{N} l_{n\theta}(x+u, y+v, t+1) \tag{5.31}$$

donde $n = 1,2,\ldots,N$ y θ es el ángulo de máxima energía para la posición (x, y) , el cual está dado por la ecuación (4.55).

Usando los coeficientes de Hermite orientados se obtiene invarianza a la rotación como se demostró en [34]. Esto permite definir las restricciones locales de la imagen considerando la constancia en la descomposición polinómica de las imágenes y la identificación de patrones visuales que son perceptualmente relevantes y representan las características más importantes de la imagen.

5.2.1.3. Restricción para el Suavizado [1]

Se asume que el flujo está suavizado parte por parte como en [4], [33] permitiéndole tener robustez en discontinuidades de flujo:

$$min \int_{\Omega} \Psi\left(|\nabla u|^2 + |\nabla v|^2\right) dX \tag{5.32}$$

donde $\Psi(s^2)$ es una función de penalización no cuadrática que permite capturar localmente el movimiento no suavizado, permitiendo valores atípicos en la presunción de suavidad

5.2.2. Energía [1]

Se define un funcional que penaliza las desviaciones de las asunciones del modelo. Todas las desviaciones de las suposiciones de restricción para la intensidad constante (ecuación (5.30)), de restricción para los coeficientes de Hermite orientados (ecuación (5.31)) y de restricción para el suavizado (ecuación (5.32)) se miden por la energía:

$$E(u,v)=E_{Data}(u,v)+\alpha E_{Smooth}(u,v) \tag{5.33}$$

donde el parámetro de regularización $\alpha<0$ es un peso de suavizado.

El término de los datos $E_{Data}(u,v)$ de la ecuación (5.33) está dado por:

$$E_{Data}(u,v)=\int_{\Omega}\Psi\left(\left|L_0(X+w)-L_0(X)\right|^2+\gamma\left|\sum_{n=1}^{N}l_{n\theta}(X+w)-\sum_{n=0}^{N}l_{n\theta}(X)\right|^2\right)dX \tag{5.34}$$

donde γ es un peso entre la restricción para la intensidad constante y la restricción para los coeficientes de Hermite orientados, y la función de optimización es $\Psi(s^2)=\sqrt{(s^2+\varepsilon^2)}$ que es robusta a los valores atípicos con $\varepsilon=0.001$ para garantizar la diferenciación de $\Psi(s^2)$ en $s=0$. En la Figura 5.3. se muestra esta función.

El término del suavizado $E_{Smooth}(u,v)$ de la ecuación 5.33 está dado por:

$$E_{Smooth}(u,v)=\int_{\Omega}\Psi\left(|\nabla u|^2+|\nabla v|^2\right)dX \quad . \tag{5.35}$$

Entonces, el funcional de energía global para la minimización que incluye los términos de datos y suavizado tiene la forma:

$$\begin{aligned}E(u,v)=&\int_{\Omega}\Psi\left(\left|L_0(X+w)-L_0(X)\right|^2+\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w)-l_{n\theta}(X)\right|^2\right)\right)dX\\&+\alpha\int_{\Omega}\Psi\left(|\nabla u|^2+|\nabla v|^2\right)dX\ .\end{aligned} \tag{5.36}$$

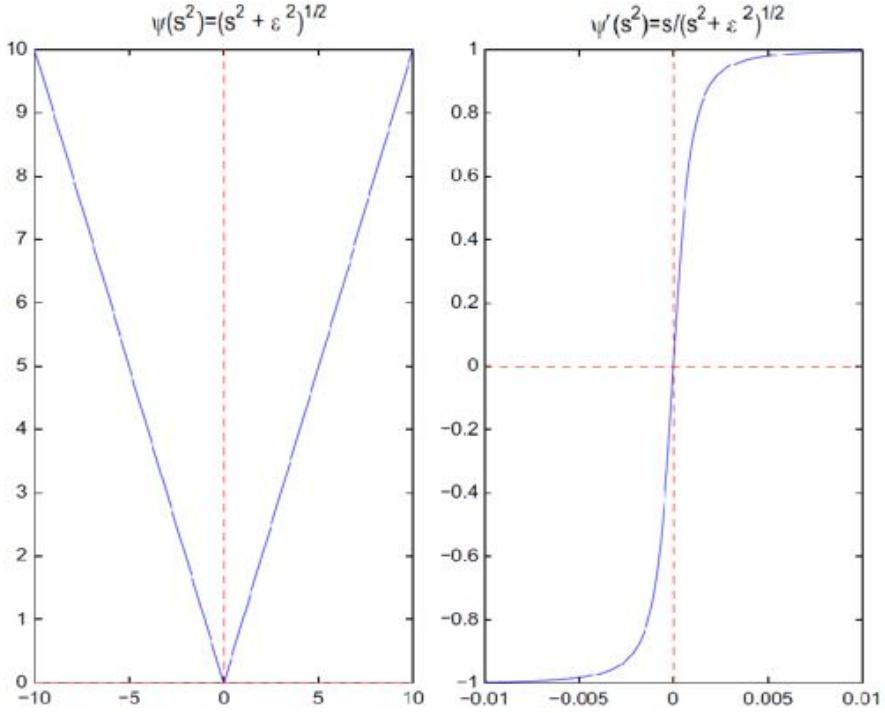


Figura 5.3. Función de optimización $\Psi(s^2)=\sqrt{(s^2+\varepsilon^2)}$ con $\varepsilon=0.001$ [1].

Las ecuaciones de Euler-Lagrange que corresponden al funcional (5.36) son:

$$\Psi'\left(\left|L_0(X+w)-L_0(X)\right|^2+\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w)-l_{n\theta}(X)\right|^2\right)\right)\cdot$$
$$\left[\left|L_0(X+w)-L_0(X)\right|\frac{\partial L_0(X+w)}{\partial u(x)}+\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w)-l_{n\theta}(X)\right|\frac{\partial l_{n\theta}(X+w)}{\partial u(x)}\right)\right]- \tag{5.37}$$
$$\alpha\, div\left(\Psi'\left(|\nabla u|^2+|\nabla v|^2\right)\nabla u\right)=0$$

y

$$\Psi'\left(\left|L_0(X+w)-L_0(X)\right|^2+\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w)-l_{n\theta}(X)\right|^2\right)\right)\cdot$$
$$\left[\left|L_0(X+w)-L_0(X)\right|\frac{\partial L_0(X+w)}{\partial v(y)}+\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w)-l_{n\theta}(X)\right|\frac{\partial l_{n\theta}(X+w)}{\partial v(y)}\right)\right]- \tag{5.38}$$
$$\alpha\, div\left(\Psi'\left(|\nabla u|^2+|\nabla v|^2\right)\nabla v\right)=0$$

La diferencia entre ambas ecuaciones recae en que, en (5.37) la derivada parcial es con respecto a u y la función de optimización a la que se le aplica el divergente

esta multiplicada por el gradiente de u , mientras que en la función (5.38), la derivada parcial y el gradiente son respecto a v .

Aplicando la regla de la cadena a las derivadas parciales de las ecuaciones (5.37) y (5.38) se tiene:

$$\begin{aligned} \frac{\partial L_0}{\partial u(x)} &= \frac{\partial L_0}{\partial x}, \\ \frac{\partial L_0}{\partial v(y)} &= \frac{\partial L_0}{\partial y} \end{aligned} \tag{5.39}$$

y

$$\begin{aligned} \frac{\partial l_{n\theta}}{\partial u(x)} &= \frac{\partial l_{n\theta}}{\partial x}, \\ \frac{\partial l_{n\theta}}{\partial v(y)} &= \frac{\partial l_{n\theta}}{\partial y} \end{aligned} \tag{5.40}$$

y ya que la derivada Gaussiana de orden k de la imagen es el producto interno de la imagen con el Polinomio de Hermite de orden k [32], las derivadas parciales quedan como:

$$\begin{aligned} \frac{\partial L_0(X+w)}{\partial x} &= L_{10}(X+w), \\ \frac{\partial L_0(X+w)}{\partial y} &= L_{01}(X+w) \end{aligned} \tag{5.41}$$

y

$$\begin{aligned} \frac{\partial}{\partial x}\sum_{n=1}^{N} l_{n\theta}(X) &= \sum_{n=1}^{N} l_{n\theta_{(m)+1}}(X), \\ \frac{\partial}{\partial y}\sum_{n=1}^{N} l_{n\theta}(X) &= \sum_{n=1}^{N} l_{n\theta_{(n)+1}}(X), \end{aligned} \tag{5.42}$$

donde $l_{n\theta_{(m)+1}}$ y $l_{n\theta_{(n)+1}}$ se calculan usando el proceso de rotación para el coeficiente de la Transformada de Hermite sustituyendo $L_{n,n-m}$ por $L_{n,n-(m+1)}$ o $L_{(n+1),(n+1)-m}$, según sea el caso.

Con el desarrollo de las derivadas parciales en (5.41) y (5.42), las ecuaciones de Euler-Lagrange para el funcional descrito en la ecuación (5.36) quedan como:

$$\Psi'\left(\left|L_0(X+w)-L_0(X)\right|^2+\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w)-l_{n\theta}(X)\right|^2\right)\right)\cdot$$
$$\left[\left|L_0(X+w)-L_0(x)\right|L_{10}(X+w)+\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w)-l_{n\theta}(X)\right|l_{n\theta_{(m)+1}}(X+w)\right)\right]- \quad (5.43)$$
$$\alpha\, div\left(\Psi'\left(|\nabla u|^2+|\nabla v|^2\right)\nabla u\right)=0$$

y

$$\Psi'\left(\left|L_0(X+w)-L_0(X)\right|^2+\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w)-l_{n\theta}(X)\right|^2\right)\right)\cdot$$
$$\left[\left|L_0(X+w)-L_0(X)\right|L_{01}(X+w)+\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w)-l_{n\theta}(X)\right|l_{n\theta_{(n)+1}}(X+w)\right)\right]- \quad (5.44)$$
$$\alpha\, div\left(\Psi'\left(|\nabla u|^2+|\nabla v|^2\right)\nabla v\right)=0\ .$$

.

5.2.3. Aproximación Iterativa [1]

Definir un sistema iterativo de punto fijo, como en [29], para obtener una solución para w en las ecuaciones (5.43) y (5.44), conduce a las siguientes ecuaciones:

$$\Psi'\left(\left|L_0(X+w^{k+1})-L_0(X)\right|^2+\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w^{k+1})-l_{n\theta}(X)\right|^2\right)\right)\cdot$$
$$\left[\left|L_0(X+w^{k+1})-L_0(X)\right|L_{10}(X+w^{k})+\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w^{k+1})-l_{n\theta}(X)\right|l_{n\theta_{(m)+1}}(X+w^{k})\right)\right] \quad (5.45)$$
$$\alpha\, div\left(\Psi'\left(|\nabla u|^2+|\nabla v|^2\right)\nabla u\right)=0$$

y

$$\Psi'\left(\left|L_0(X+w^{k+1})-L_0(X)\right|^2+\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w^{k+1})-l_{n\theta}(X)\right|^2\right)\right)\cdot$$
$$\left[\left|L_0(X+w^{k+1})-L_0(X)\right|L_{01}(X+w^{k})+\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w^{k+1})-l_{n\theta}(X)\right|l_{n\theta_{(n)+1}}(X+w^{k}\right. \quad (5.46)$$
$$\alpha\, div\left(\Psi'\left(|\nabla u|^2+|\nabla v|^2\right)\nabla v\right)=0\ ,$$

donde $w^k=(u^k,v^k,1)^T$ son las variables desconocidas u^k y v^k en la iteración externa k y w^{k+1} son las soluciones a las ecuaciones (5.45) y (5.46).

Las ecuaciones (5.45) y (5.46) no son lineales en términos de la forma $f(X+w^{k+1})-f(X)$, por lo que se utiliza el primer orden de la expansión de Taylor de dichos términos.

Para $L_0(X+w^{k+1})-L_0(X)$ la expansión de Taylor queda como:

$$\begin{aligned}
&L_0(X+w^{k+1})-L_0(X)\approx \\
&\left(L_0(X+w^{k+1})+du^k\frac{\partial L_0(X)}{\partial x}+dv^k\frac{\partial L_0(X)}{\partial y}\right)-L_0(X)\approx \\
&\left(L_0(X+w^{k+1})+du^k L_{01}(X)+dv^k L_{10}(X)\right)-L_0(X)\approx \\
&(L_0(X+w^{k+1})-L_0(X))+du^k L_{01}(X)+dv^k L_{10}(X)
\end{aligned} \tag{5.47}$$

y para $l_{n\theta}(X+w^{k+1})-l_{n\theta}(X)$ queda como:

$$\begin{aligned}
&\sum_{n=1}^{N}(l_{n\theta}(X+w^{k+1})-l_{n\theta}(X))\approx \\
&\sum_{n=1}^{N}\left(\left[l_{n\theta}(X+w^{k+1})+du^k\frac{\partial l_{n\theta}(X)}{\partial x}+dv^k\frac{\partial l_{n\theta}(X)}{\partial y}\right]-l_{n\theta}(X)\right)\approx \\
&\sum_{n=1}^{N}\left(\left[l_{n\theta}(X+w^{k+1})+du^k l_{n\theta_{(m)+1}}(X)+dv^k l_{n\theta_{(n)+1}}(X)\right]-l_{n\theta}(X)\right)\approx \\
&\sum_{n=1}^{N}\left(\left[l_{n\theta}(X+w^{k+1})-l_{n\theta}(X)\right]+du^k l_{n\theta_{(m)+1}}(X)+dv^k l_{n\theta_{(n)+1}}(X)\right).
\end{aligned} \tag{5.48}$$

Aplicando las ecuaciones (5.48) y (5.47) a (5.45) y (5.46), los sistemas de ecuaciones resultantes son:

$$\begin{aligned}
&\Psi'\left(\left|L_0(X+w^k)-L_0(X)+du^k L_{10}(X+w^k)+dv^k L_{01}(X+w^k)\right|^2+\right. \\
&\left.\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w^k)-l_{n\theta}(X)+du^k l_{n\theta_{(m)+1}}(X+w^k)+dv^k l_{n\theta_{(n)+1}}(X+w^k)\right|^2\right)\right)\cdot \\
&\left[\left|L_0(X+w^k)-L_0(X)+du^k L_{10}(X+w^k)+dv^k L_{01}(X+w^k)\right|L_{10}(X+w^k)+\right. \\
&\left.\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w^k)-l_{n\theta}(X)+du^k l_{n\theta_{(m)+1}}(X+w^k)+dv^k l_{n\theta_{(n)+1}}(X+w^k)\right|\cdot l_{n\theta_{(m)+1}}(X+w^k)\right)\right]\cdot \\
&\alpha\, div\left(\Psi'\left(\left|\nabla(u^k+du^k)\right|^2+\left|\nabla(v^k+dv^k)\right|^2\right)\nabla(u^k+du^k)\right)=0
\end{aligned} \tag{5.49}$$

y

$$\Psi'\left(\left|L_0(X+w^k)-L_0(X)+du^k L_{10}(X+w^k)+dv^k L_{01}(X+w^k)\right|^2+\right.$$
$$\left.\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w^k)-l_{n\theta}(X)+du^k l_{n\theta_{(m)+1}}(X+w^k)+dv^k l_{n\theta_{(n)+1}}(X+w^k)\right|^2\right)\right)\cdot$$
$$\left[\left|L_0(X+w^k)-L_0(x)+du^k L_{10}(X+w^k)+dv^k L_{01}(X+w^k)\right|L_{01}(X+w^k)+\right. \qquad (5.50)$$
$$\left.\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w^k)-l_{n\theta}(X)+du^k l_{n\theta_{(m)+1}}(X+w^k)+dv^k l_{n\theta_{(n)+1}}(X+w^k)\right|\cdot l_{n\theta_{(n)+1}}(X+w^k)\right)\right.$$
$$\left.-\alpha\, div\left(\Psi'\left(\left|\nabla(u^k+du^k)\right|^2+\left|\nabla(v^k+dv^k)\right|^2\right)\nabla(v^k+dv^k)\right)=0\right].$$

Para remover la no linealidad en Ψ' se aplica una segunda iteración de punto fijo a las ecuaciones (5.49) y (5.50), para esta iteración interna se usa el índice q y las ecuaciones quedan como:

$$\Psi'\left(\left|L_0(X+w^k)-L_0(X)+du^{k,q}L_{10}(X+w^k)+dv^{k,q}L_{01}(X+w^k)\right|^2+\right.$$
$$\left.\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w^k)-l_{n\theta}(X)+du^{k,q}l_{n\theta_{(m)+1}}(X+w^k)+dv^{k,q}l_{n\theta_{(n)+1}}(X+w^k)\right|^2\right)\right)\cdot$$
$$\left[\left|L_0(X+w^k)-L_0(X)+du^{k,q}L_{10}(X+w^k)+dv^{k,q}L_{01}(X+w^k)\right|L_{10}(X+w^k)+\right. \qquad (5.51)$$
$$\left.\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w^k)-l_{n\theta}(X)+du^{k,q+1}l_{n\theta_{(m)+1}}(X+w^k)+dv^{k,q+1}l_{n\theta_{(n)+1}}(X+w^k)\right|\cdot\right.\right.$$
$$\left.\left.l_{n\theta_{(m)+1}}(X+w^k)\right)\right]-\alpha\, div\left(\Psi'\left(\left|\nabla(u^k+du^k)\right|^2+\left|\nabla(v^k+dv^k)\right|^2\right)\nabla(u^k+du^{k,q+1})\right)=0$$

y

$$\Psi'\left(\left|L_0(X+w^k)-L_0(X)+du^{k,q}L_{10}(X+w^k)+dv^{k,q}L_{01}(X+w^k)\right|^2+\right.$$
$$\left.\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w^k)-l_{n\theta}(X)+du^{k,q}l_{n\theta_{(m)+1}}(X+w^k)+dv^{k,q}l_{n\theta_{(n)+1}}(X+w^k)\right|^2\right)\right)\cdot$$
$$\left[\left|L_0(X+w^k)-L_0(X)+du^{k,q+1}L_{10}(X+w^k)+dv^{k,q+1}L_{01}(X+w^k)\right|L_{01}(X+w^k)+\right. \qquad (5.52)$$
$$\left.\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w^k)-l_{n\theta}(X)+du^{k,q+1}l_{n\theta_{(m)+1}}(X+w^k)+dv^{k,q+1}l_{n\theta_{(n)+1}}(X+w^k)\right|\cdot\right.\right.$$
$$\left.\left.l_{n\theta_{(n)+1}}(X+w^k)\right)\right]-\alpha\, div\left(\Psi'\left(\left|\nabla(u^k+du^k)\right|^2+\left|\nabla(v^k+dv^k)\right|^2\right)\nabla(v^k+dv^{k,q+1})\right)=0.$$

5.2.4. Estrategia Multiescala [1]

Para considerar grandes desplazamientos se aplica una estrategia multiescala [25], esto no contradice la linealización considerada en las ecuaciones (5.49) y (5.50).

Se genera una pirámide Gaussiana de la imagen usando un factor de disminución de escala $\eta \in (0,1)$, donde η permanece constante en cada etapa. La imagen burda es obtenida escalando la imágenes $L(x, y, t)$ y $L(x+u, y+v, t+1)$ por un factor η^i para $i = M-1, M-2, \ldots, 0$,donde M representa el número de niveles de descomposición.

Comenzando en el nivel $i = M-1$ con $W^0 = (0,0,1)^T$, $du^{k,0} = 0$ y $dv^{k,0} = 0$ la iteración interna (q) permite obtener el incremento du^k y dv^k para la iteración externa (k), donde $u^{k+1} = u^k + du^{k,q+1}$ y $v^{k+1} = v^k + dv^{k,q+1}$ son las soluciones para los sistemas de ecuaciones (5.51) y (5.52), respectivamente, en el nivel actual. Las soluciones se interpolan y propagan hasta el siguiente nivel $i = i-1$, donde se emplean para la inicialización de la iteración exterior.

5.3. Restricción de Regiones Homogéneas para los Coeficientes de Hermite [2]

En el método de Moya-Albor [1] se calculan los vectores de velocidad en toda la imagen incluyendo las zonas donde no existe movimiento debido al esquema de suavizado incluido en el funcional, esto puede causar errores en el proceso de estimación. En busca de solucionar en cierta medida este tipo de problemas, algunos autores proponen incorporar el término de regiones homogéneas [35], [36] para reducir las propagaciones indeseables del campo de velocidades en todos los puntos de la imagen donde no existe movimiento. La función de restricción en regiones homogéneas está dada por:

$$E_h = \int_{\Omega} H(X)(w)^2 dx \quad , \tag{5.53}$$

donde H es la función que penaliza las zonas homogéneas en la imagen. Este parámetro incluye una función de penalización en regiones donde el gradiente es bajo, es decir, donde se generan movimientos poco perceptibles. En términos de los coeficientes de Hermite, la función H se calcula como:

$$H(X) = \frac{1}{\sqrt{\left(L_0(X+w) - L_0(X)\right)^2}} \quad . \tag{5.54}$$

Esta función penaliza las zonas donde no se genera movimiento entre los coeficientes de Hermite $L_0(X+w)$ y $L_0(X)$, los cuales corresponden a distintos tiempos de la secuencia. Al integrar la penalización al funcional que calcula la estimación de movimiento, queda:

$$E(u,v)=\int_{\Omega}\Psi\left(\left|L_0(X+w)-L_0(X)\right|^2+\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w)-l_{n\theta}(X)\right|^2\right)\right)dX \\ +\alpha\int_{\Omega}\Psi\left(|\nabla u|^2+|\nabla v|^2\right)dX+\beta\int_{\Omega}H(X)(w)^2dX \qquad (5.55)$$

donde β es el peso que se le asigna a la penalización.

Capítulo 6
Desarrollo

Las características del Equipo de Cmputo que se emplea son:

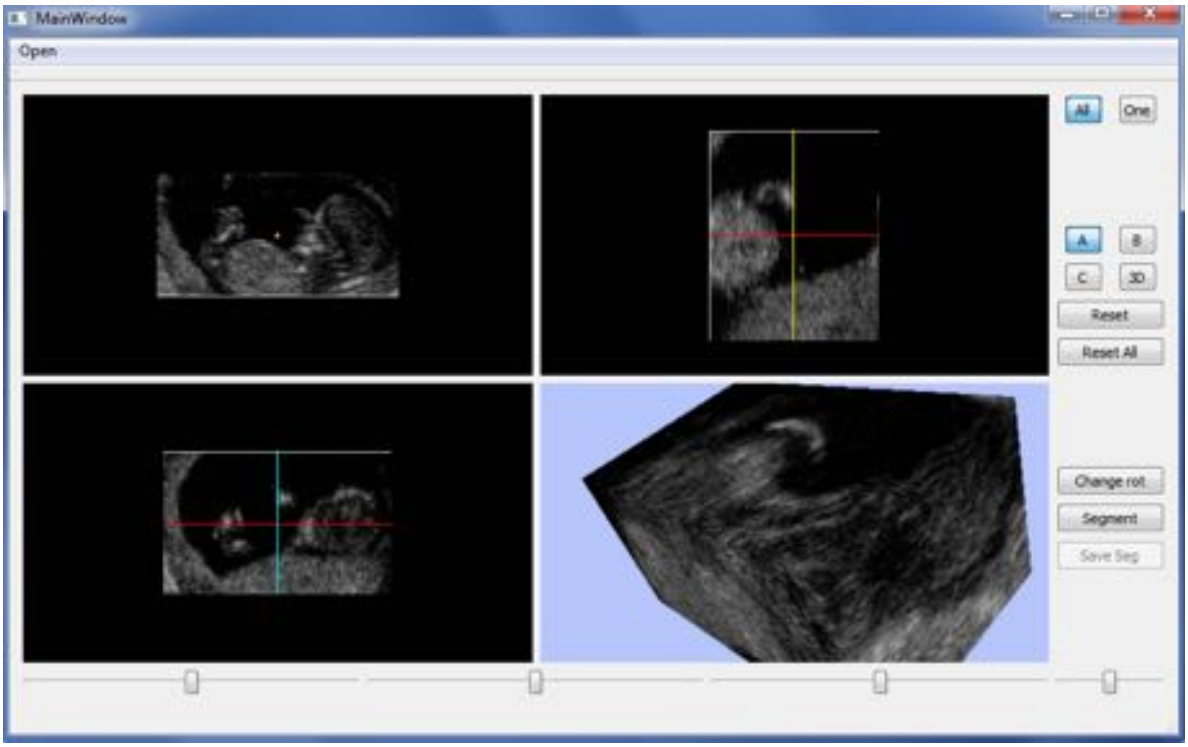
- Sistema Operativo Windows 7 Professional
- Sistema Operativo de 64 bits
- Procesador Intel(R) Xenon(R) CPU E5-2620 v4 @ 2.10 GHz
- Memoria RAM de 32 GB

Debido a que se necesita que el flujo óptico y el modo M sean integrados al proyecto mencionado en el planteamiento del problema, como punto de partida se tiene una versión muy básica de la interfaz principal del proyecto entero, su despliegue se muestra en la Figura 6.1., y cuenta con las funciones para:

- Leer volúmenes 3D
- Visualizar y navegar en el volumen mostrado en tres vistas 2D y una 3D. Las opciones disponibles en ellas son:
  - Cambiar el contraste al mantener presionado el botón izquierdo del mouse sobre la imagen, al mismo tiempo que se mueve el cursor.
  - Alejar y acercar manteniendo presionado el botón derecho sobre la imagen muestra se mueve, o deslizando el scroll del mouse.
  - Cambiar de lugar a la imagen al mantener presionado el scroll y mover el mouse.
  - Rotar la vista seleccionada sobre un punto, siendo el centro el punto definido inicialmente. Las rotaciones se hacen mediante las barras deslizantes localizadas debajo de las vistas.
  - Cambiar el corte visualizado con la barra deslizante localizada en la esquina inferior derecha.
  - Ver sólo una ventana al seleccionar el botón *One*. Se muestra la que estaá seleccionada con *A, B, C* o *3D.*
  - Reiniciar visualización de la ventana con *Res*et o de todas ellas con *Reset All*.
- Segmentar y guardar un pedazo de imagen de alguna de las vistas 2D mostradas.

Además de desplegar una ventana de diálogo que muestra información relacionada con el volumen y las acciones que se realizan.

a)



b)

Figura 6.1. Visualización de un volumen 3D de un archivo a través de la Interfaz inicial básica. (a) Ventana de Diálogo. (b) Interfaz de visualización.

Los pasos para la generación de la interfaz a partir de la que ya se tiene son:

1. Descargar e instalar Software y Librerías para una aplicación de 64 bits.
2. Compilar ITK y VTK para funcionar de forma individual y en conjunto.
3. Probar funcionamiento e iniciar el aprendizaje de las funciones de las librerías ITK y VTK.
4. Analizar y compilar la interfaz recibida.
5. Generar el código para la lectura de volúmenes 4D con base en el de 3D que ya existe.
6. Rehusar la interfaz de despliegue de volúmenes 3D para los volúmenes 4D.
7. Agregar y modificar funciones para la visualización de volúmenes 4D.
8. Rehusar y modificar parte de las funciones asociadas a la segmentación para generar la línea de captura y la imagen del Modo M.
9. Desplegar el modo M en su propia ventana.
10. Agregar elementos de interacción con el modo M.
11. Creación del filtro ITK que calcula los coeficientes de Hermite de orden 3 y sus rotaciones.
12. Creación del filtro ITK que calcula la estimación de movimiento.
13. Visualizar el Flujo óptico en su propia ventana.
14. Añadir funciones para generar, guardar y mostrar una secuencia de contornos con su flujo óptico asociado.
15. Graficar las magnitudes de los vectores de flujo óptico localizados en los contornos seleccionados.
16. Visualizar el modo M traslapado con la gráfica.
17. Agregar interacción con elementos observados.

6.1. Software

El Software y las librerías ocupadas son:

- Microsoft Visual Studio 2008
- Qt 4.8.7
  - Visual Studio Add-in 1.1.11 para Qt4
  - Qt Creator 4.4.0
- CMake 2.8.9
- ITK 4.4.2
- VTK 5.10.1

6.1.1. Microsoft Visual Studio

Visual Studio es un conjunto de herramientas de desarrollo para la creación de aplicaciones y servicios Web, de escritorio y móviles. Utilizan un mismo entorno de

desarrollo integrado (IDE) para Visual Basic, Visual C# y Visual C++, permitiéndoles compartir herramientas y facilitando la creación de soluciones de idioma mixto. Además, de utilizar la funcionalidad del Framework .NET, que simplifica el desarrollo de las aplicaciones [37].

Visual C ++ es un poderoso lenguaje diseñado para brindar un control profundo y detallado al crear aplicaciones nativas de Windows o aplicaciones de Windows gestionadas por el Framework .NET [38].

Para el sistema realizado se genera una aplicación de 64 bits debido a la cantidad de memoria manejada en los cálculos pero también funciona correctamente con una configuración a 32 bits. Aunque hay versiones más recientes de Visual Studio, se designó 2008 por su compatibilidad con las versiones de las librerías usadas. Este es el primer software en instalar ya que se emplea su compilador en la generación del las librerías y la interfaz gráfica.

6.1.2. Qt [39]

Qt es un framework de desarrollo de aplicaciones multiplataforma (Linux, OS X, Windows, VxWorks, QNX, Android, iOS, BlackBerry, Sailfish OS, etc) para escritorio, embebido y móvil con elementos de interfaz de usuario, bibliotecas C ++ y un entorno de desarrollo integrado con herramientas. El propio framework y las aplicaciones o librerías que lo utilizan pueden compilarse con cualquier compilador compatible con C ++ estándar como Clang, GCC, ICC, MinGW y MSVC.

Qt inició en 1990 con la empresa Trolltech, vendiendo licencias Qt y le brindó soporte. Pasando por varias adquisiciones a lo largo de los años, actualmente se llama The Qt Company y aunque es el principal impulsor detrás de Qt. Este es desarrollado por una alianza más grande compuesta por muchas empresas y personas de todo el mundo siguiendo un modelo de gobernanza meritocrática.

La ventaja de usar Qt, es su mecanismo de comunicación entre las funciones programadas y los elementos gráficos de la interfaz de manera que cuando se produce un evento sobre algún, éste emite una señal a sus funciones asociadas para que realicen alguna tarea.

6.1.3. ITK [5]

Insight Segmentation and Registration Toolkit (ITK) es un sistema multiplataforma de código abierto que proporciona un enorme conjunto de herramientas de software para el análisis de imágenes de 2 o más dimensiones. ITK se implementa en C ++, utilizando el entorno de compilación de CMake para administrar el

proceso de configuración. Además de un proceso automatizado de envoltura que genera interfaces entre C++ y lenguajes de programación interpretados como Java y Python.

ITK surge en 1999 cuando la Biblioteca Nacional de Medicina del Instituto Nacional de Salud de los Estados Unidos otorga un contrato de tres años para desarrollar un kit de herramientas de registro y segmentación de código abierto, que finalmente llegó a conocerse como el Insight Toolkit (ITK).

6.1.4. VTK [6]

Visualization Toolkit (VTK) es un sistema de software de código abierto y libremente disponible para gráficos de computadora 3D, modelado, procesamiento de imágenes, renderizado de volumen, visualización científica y visualización de información. VTK también incluye soporte auxiliar para widgets de interacción 3D, anotación bidimensional y tridimensional, y computación paralela. En su núcleo, VTK se implementa como un kit de herramientas de C++, que requiere que los usuarios construyan aplicaciones combinando varios objetos en una aplicación. El sistema también admite el envoltorio automatizado del núcleo C++ en Python, Java y Tcl. VTK se utiliza en todo el mundo en aplicaciones comerciales, así como en investigación y desarrollo. Es la base de muchas aplicaciones avanzadas de visualización como ParaView, VisIt, VisTrails, Slicer, MayaVi y OsiriX.

VTK originalmente se crea en diciembre de 1993 con el permiso legal de su entonces patrón, GE Corporate R&D como parte del libro de texto The Visualization Toolkit. Un enfoque orientado a objetos para Gráficos 3D, escrito por Will Schroeder, Ken Martin y Bill Lorensen, tres investigadores de graficación y visualización, con la motivación de colaborar con otros investigadores y desarrollar un marco abierto para la desarrollar de visualización de vanguardia y aplicaciones gráficas. Después de escribir el núcleo de VTK, los usuarios y desarrolladores de todo el mundo comienzan a mejorar y aplicar el sistema a los problemas del mundo real.

6.1.5. CMake [40]

CMake es un conjunto de herramientas multiplataforma de código abierto para el desarrollo, prueba y empaquetado de software. Controla el proceso de compilación del software utilizando plataformas simples y archivos de configuración independientes del compilador, y genera makefiles y espacios de trabajo nativos que pueden usarse en un entorno de compilación definido. El conjunto de herramientas CMake se crea como una respuesta a la necesidad de un entorno de construcción para proyectos de código abierto como ITK y VTK.

6.2. Descripción del Sistema

Figura 6.2. Diagrama de clases utilizadas para la visualización en la interfaz principal.

La clase principal de todo el sistema es *MainWindow,* que se encarga de la visualización principal y desde ahí llamar a todas las demás clases que se ejecutan al interactuar con la interfaz a través de botones, barras de desplazamiento (slider) y casillas de verificación (checkbox). En la Figura 6.2. se muestra el diagrama de las clases utilizadas para la visualización de la interfaz principal.

Para cargar una ecocardiografía 4D de tipo .vol que se genera con los ultrasonidos General Electric ocupados en el Instituto Nacional de Perinatología, se tiene a la clase *Open4D*, que guarda la información como una imagen 4D de ITK. Ésta es mandada a llamar por la función *open4D* definida en *MainWindow*

Para obtener una imagen 3D de ITK a partir de la de 4D de ITK, se usa el método *itkExtractImageFilter*. Al obtener la imagen 3D en ITK, se le cambia su tipo a VTK con el método *itkImageToVTKImafeFilter*, mientras que con *vtkImageFlip* se invierte el sentido del eje Y. Los volúmenes se visualizan al asignarlos a la variable *volumeData* que se emplea mandarle a la interfaz principal el volumen que se desea mostrar.

Al tener una secuencia de varios volúmenes es necesario agregar elementos de interacción, eligiendo un checkbox nombrado como *FixPlaneBox* y asociándolo a la función *fixPlaneBoxClicked* para seleccionar el corte actual cuando es activado, y un slider llamado *sequenceSld* que se comunica con la función *changeDisplaySeq* para navegar entre los distintos volúmenes o cortes, si el checkbox anterior está seleccionado, ambas funciones pertenecen a la clase *MainWindow*.

Al seleccionar un corte, se toma ese corte en cada volumen de la secuencia y se guardan como imágenes 2D de VTK dentro del vector *imgSecVTK* en el orden de la secuencia. El corte correspondiente al volumen actual se guarda en la variable *sliceImageAxial* para actualizar la vista.

La selección del modo M se inicia al presionar el botón *mModeBtn* que se conecta con la función *mModeBtnClicked* y generar un objeto de la clase existente *vtkTracerInteractorStyle*, la cual hace posible realizar trazos sobre la imagen desplegada. Se modifica *vtkTracerInteractorStyle* para que el resultado del trazo pueda entregarse a otras clases asociadas a las ventanas sistema y no únicamente a la principal.

MainWindow recibe el trazo de *vtkTracerInteractorStyle* por medio de la función *setSegmentedPath*, la cual se complementa para distinguir entre el trazo activado con el botón de segmentación y el activado con el botón de modo M. Para el segundo caso se genera la imagen 2D de VTK correspondiente y se guarda para

después desplegarse en su propia ventana creada con Qt, asociada a la clase *MmodeWidget* y consta de sus propias funciones para su visualización. En la Figura 6.3. se tiene diagrama de las clases que intervienen en la creación y manipulación del modo M.

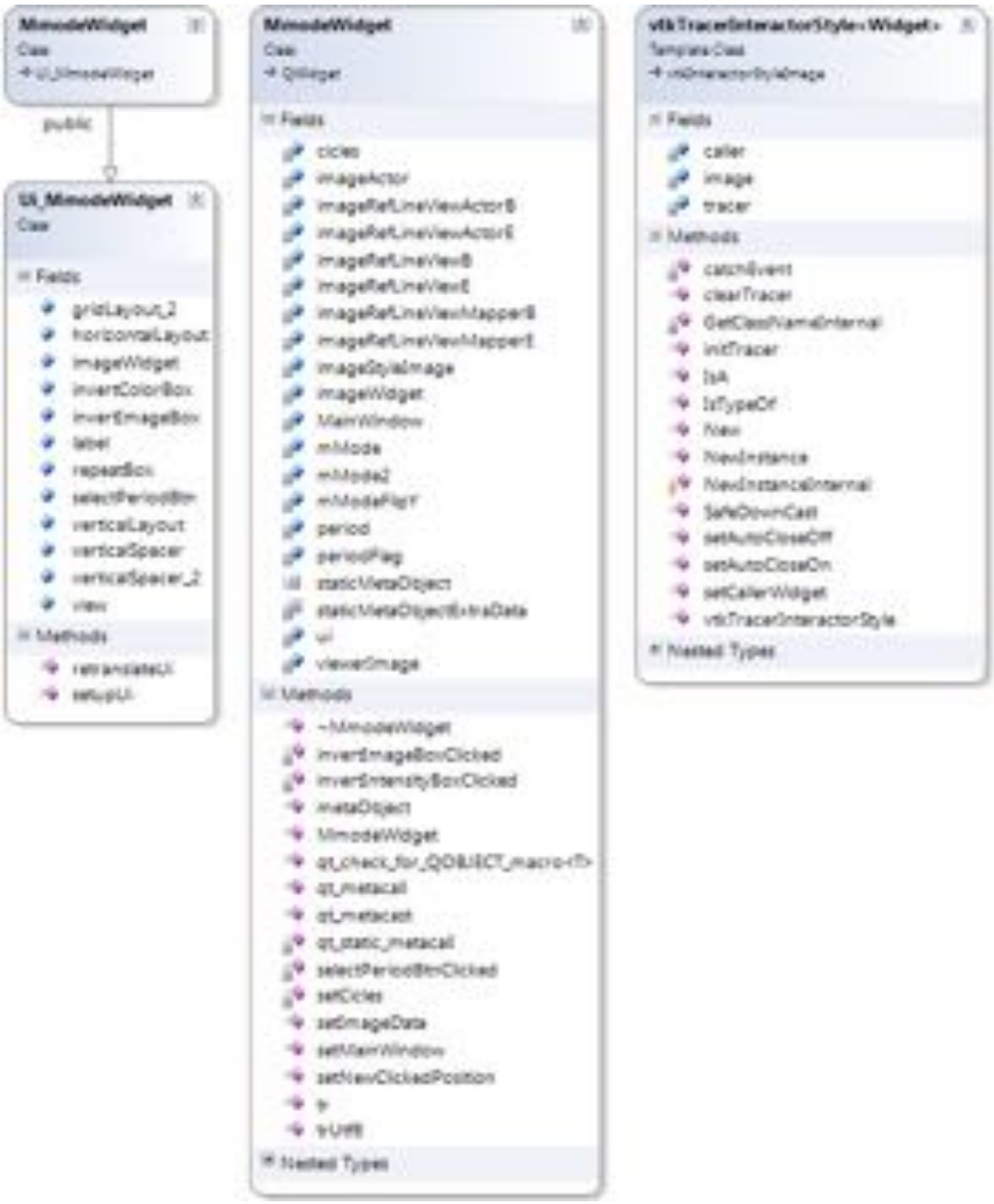


Figura 6.3. Diagrama de clases relacionadas con el modo M.

La posición dos puntos contenidos en la recta son obtenidos con la utilización de los métodos *vtkParametricSpline* y *vtkParametricFunctionSource* para generar un spline a partir de los puntos de inicio y fin de una línea recta, el spline regresa todos los puntos que contiene y a partir de estos se calculan las posiciones de los píxeles del corte que coinciden con la línea. Tomando las coordenadas de los píxeles se acomodan sus valores en una imagen 2D de VTK llamada *mMode*. Por el tipo de datos en la secuencia de cortes, es importante indicar que la imagen use escalares de tipo unsigned char de 1 componente, de lo contrario, aunque se guarden los datos, se ve la imagen como un rectángulo negro. Con el modo M creado se remueve el objeto *mModeStyle* y se inicializa una ventana de la clase *MmodeWidget* pasándole la imagen *mMode*.

Ya sea que el modo M sea generado o no, con la secuencia de cortes se calcula la estimación de movimiento al presionar el botón *opticalFlowBtn* el cual esta ligado con la función *opticalFlowBtnClicked* dentro de la clase principal, encargándose de ejecutar la función *opticalFlowHermite* y generar la ventana de flujo óptico entregando los datos obtenidos de la estimación y el modo M en caso de haber sido generado.

La función *opticalFlowHermite*, perteneciente a la clase *MainWindow*, interactuá con el filtro *itkHetmiteOpticalFlow* realizado con métodos de ITK y encargado de calcular los vectores de flujo óptico mediante la resolución de la ecuación (5.55). Dicho filtro, a su vez ocupa a otro, llamado *itkHetmiteImageFilter* y también realizado con métodos de ITK, para obtener los coeficientes y los coeficientes rotados de Hermite de tercer orden a través de un algoritmo rápido que calcula derivadas parciales hasta orden 3 y las regresan individualmente al llamarlas como imágenes 2D de ITK. i*tkHermiteOpticalFlow* e *itkHetmiteImageFilter* funcionan como filtro gracias al método *itkImageToImageFilter.*

ItkHermiteOpticalFlow recibe dos imágenes 2D de ITK mediante las funciones *SetInputImage0* para la primera imagen y *SetInputImage1* para la segunda, y opcionalmente el número de escalas e iteraciones en cada escala con las funciones *SetScale* y *SetIterations* respectivamente, si no son llamadas se usa 1 para ambas. La estimación de movimiento se devuelve mediante las funciones *GetDu* y *GetDv*.

Las derivadas promediadas son llevadas a cabo por la función *Gradient* de la clase *itkHermiteOpticalFlow* que recibe u o v , y una bandera que indica a cuál de las dos variables pertenece.

La modificación del tamaño de las imágenes se realiza con el método *itkResampleImageFilter* complementandolo para que realice una interpretación lineal con *itkLinearInterpolateImageFunction* e *itkIdentityTransform*. También se

ocupa a *itkLinearInterpolateImageFunction* para calcular la interpolación de la segunda imagen.
Las matrices se crean con el método *vnl/vnl_sparse_matrix* para poder ocuparse con *vnl/algo/vnl_sparse_lu*, con la que se hace la factorizacion LU . Mientras que los resultados se guardan como imágenes en *sorDu* y *sorDv*.

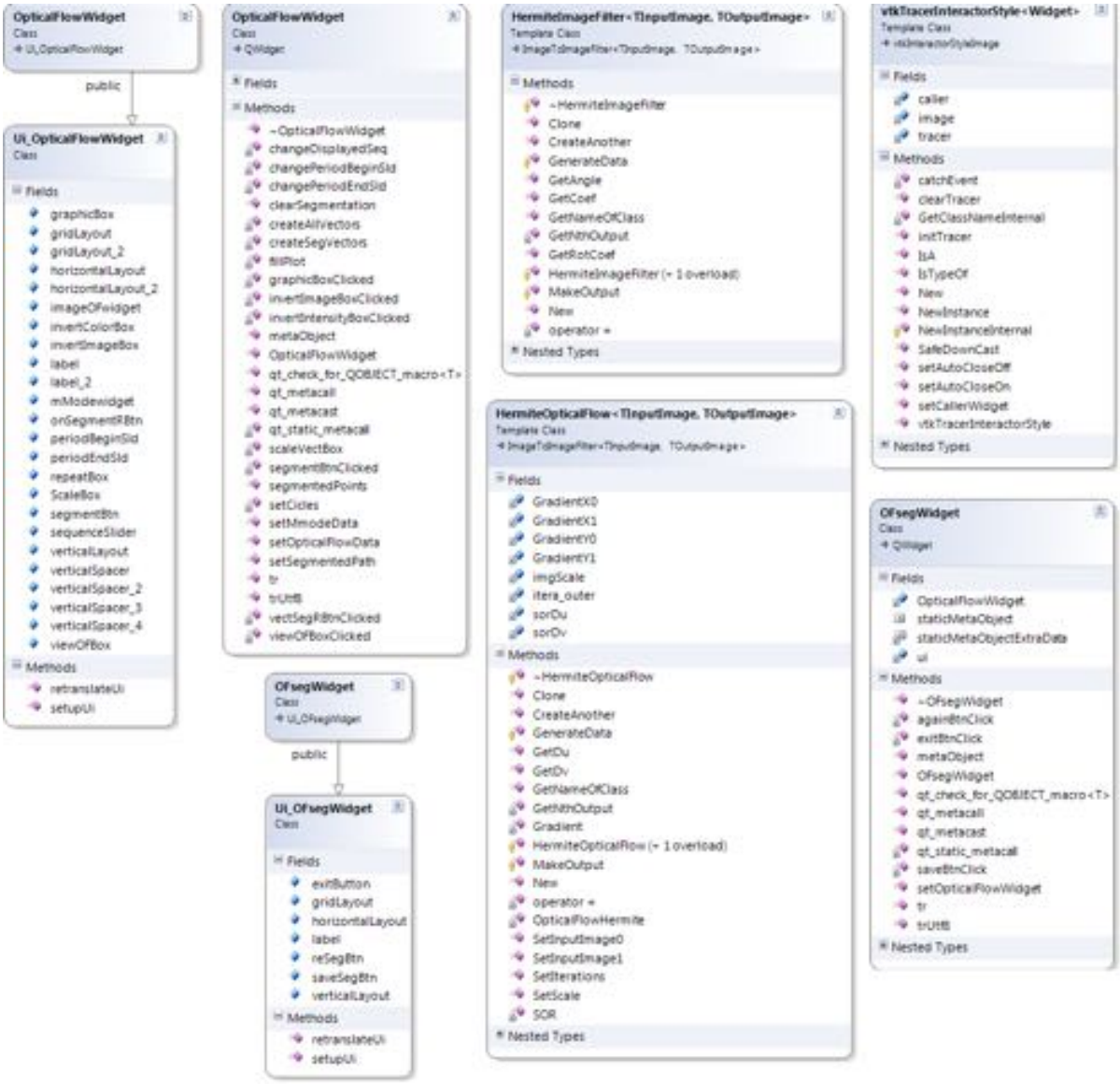


Figura 6.4. Diagrama las clases asociadas con el flujo óptico.

La ventana de flujo óptico, al igual que para el modo M, es creada con Qt y está asociada a la clase *OpticalFlowWidget* que consta de funciones para la visualización del modo M similares a los implementados en su correspondiente

ventana, funciones de visualización parecidas a las de la interfaz inicial para interactuar con la secuencia de cortes y funciones para interactuar con los vectores de flujo óptico, segmentar un contorno por el usuario y generar la gráfica de las magnitudes de los vectores sobre ese contorno.El diagrama de la clases que intervienen en el flujo óptico se muestra en la Figura 6.4.

Juntando todo lo anterior, el diagrama completo de las clases hechas para la interfaz, se muestra en la Figura 6.5.

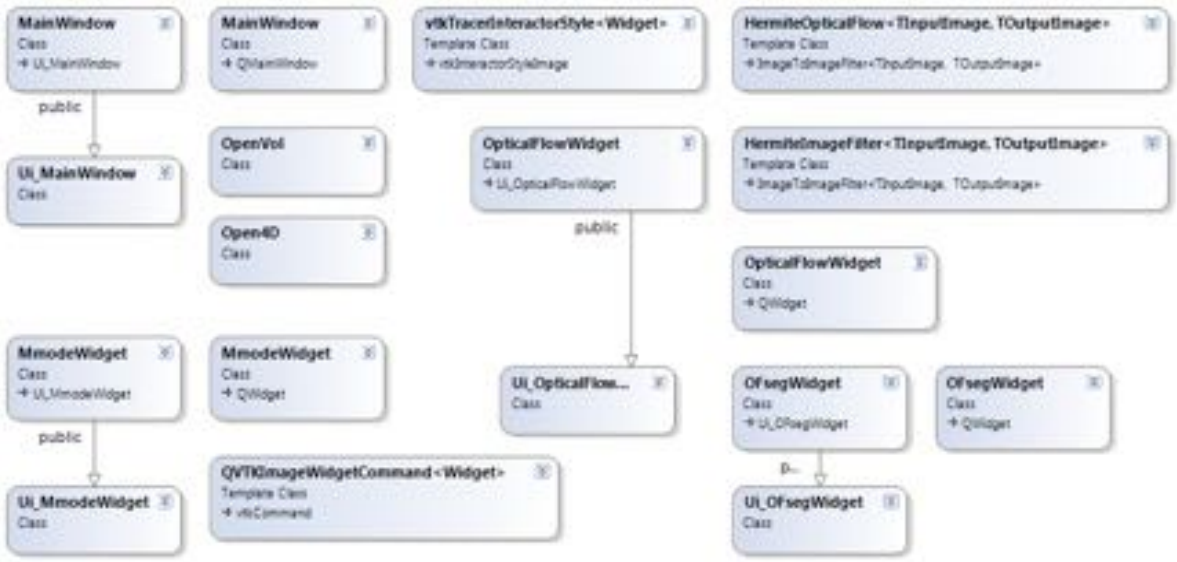


Figura 6.5. Diagrama de todas las clases del Sistema.

En las guiás de ITK y VTK [41], [42] y la documentación en sus páginas web asociadas [5], [6] se describen y ejemplifican el uso y definición de sus respectivos métodos, por lo que en la descripción del proceso de desarrollo únicamente se mencionan.

6.3. Manipulación de Volúmenes 4D

Se organiza la información de un archivo .vol que contiene una ecocardiografía 4D de tal manera que lo referido a los píxeles se guarda en una imagen 4D de ITK mientras que los datos técnicos del equipo, institución, paciente e imagen sólo se muestran en la ventana de línea de comandos. La imagen creada tiene su origen en las coordenadas (0,0,0,0) y los datos que guarda son del tipo unsigned char.

La imagen 4D de ITK es separada en una secuencia de imágenes 3D y convertidas a imágenes de VTK con el sentido del eje Y invertido, para pasar su contenido a un vector de imágenes 3D de VTK en el orden que les corresponde

según su posición en la imagen 4D para ser visualizadas individualmente de forma secuencial.

La visualización de los volúmenes respeta el orden en que conforman a la imagen 4D, siendo el primer volumen de la secuencia él que se visualiza primero. Ya que la interfaz inicial ahora cuenta con elementos para interactuar un único volumen y con secuencia de volúmenes, al utilizar la visualización para secuencias, se activan o desactivan los elementos de interacción.

6.3.1. Selección de un corte

En la interfaz principal, en la ventana superior izquierda se visualiza la cara del volumen correspondiente a los cortes que pueden elegirse para generar una la secuencia de imágenes 2D correspondientes al cambio de ese plano a lo largo del tiempo.

Al seleccionar un corte se recorre la imagen 4D de ITK y en cada volumen se toma una imagen con la misma posición del corte seleccionado y se guarda como una imagen 2D de VTK dentro del un vector en el orden de la secuencia. El proceso empleado es el mismo que para generar el vector con imágenes 3D pero colapsando una dimensión más, obteniendo así una secuencia de cortes 2D.

Del vector de imágenes 2D se selecciona el corte correspondiente al volumen actual para actualizar la vista correspondiente mientras que las otras tres no se modifican. Además se activan los botones para obtener el modo M y el flujo óptico mientras que se desactivan los sliders para las rotaciones y la navegación en el volumen.

Así la visualización de las secuencias depende de si se ha o no seleccionado un corte, en caso negativo se le asigna a la variable que guarda el volumen de despliegue, el volumen correspondiente al vector de imágenes 3D de VTK y se actualizan todas las vistas. En caso afirmativo sólo se modifica la vista superior izquierda con la imagen 2D actualizada. Ya que asignar una imagen 2D es más rápido que asignar un volumen 3D cuando ambos tienen el mismo tamaño en X y Y, y que con la imagen sólo se modifica una vista mientras que con el volumen se modifican las cuatro, la navegación entre los cortes es más fluida que entre los volúmenes perteneciente a la misma ecocardiografía 4D.

6.4. Modo M

El modo M se obtiene a partir de los datos de cada corte a la largo de la secuencia seleccionada que coinciden con una línea recta trazada por el usuario. y sirve para

ver pictóricamente como los valores asociados se van modificando. El proceso para obtener el modo M puede verse en la Figura 6.6., el cual consiste en:

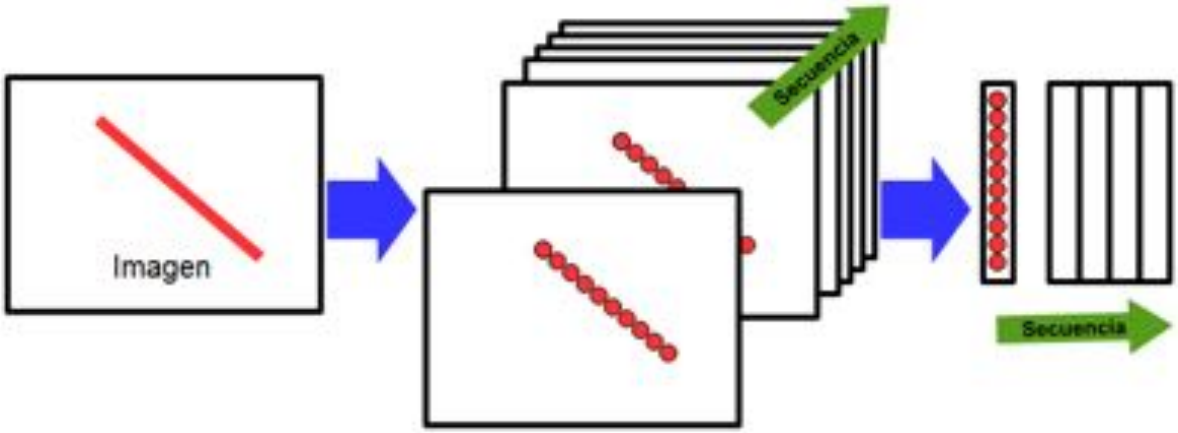


Figura 6.6. Diagrama de la obtención del modo M.

1. Trazar una línea recta sobre un corte.

2. En el sentido en que se dibuja la línea, obtener los valores de los píxeles asociados en cada imagen de la secuencia.

3. En el sentido de la secuencia, generar una tira por cada imagen con los valores obtenidos y acomodarlas en un único arreglo.

El modo M se calcula a partir de la información a una misma línea recta sobre cada corte, para trazar la línea se selecciona únicamente dos puntos de control, el origen y el fin.

6.4.1. Visualización del Modo M

A la nueva ventana se le agregan, con Qt, dos checkbox con las tareas de invertir el color de la imagen e invertir la imagen sobre el eje Y, un spinbox para repetir la imagen hasta 10 veces sobre el eje X y un botón para poder seleccionar el inicio y fin de un periodo de interés con el mouse. El diseño de esta ventana y su contenido se visualiza en la Figura 6.7.

El checkbox para Invertir colores se llama *invertColorBox* y se asocia con la función *invertIntensityBoxClicked* de la clase *MmodeWidget*. Esta acción únicamente requiere del método *vtkImageShiftScale.*

La inversión de una imagen ya se realizó en *MainWindow* para visualizar los cortes invertidos en Y, sin embargo al compilar el proyecto a 64 bits el resultado en

todas las ventanas secundarias (las correspondientes a modo M y flujo óptico) muestran como resultado una imagen completamente negra por lo que posteriormente se modificó generando una nueva imagen llamada *mModeFlipY* y asignándole los píxeles correspondientes a la coordenada Y en orden inverso al que están en *mMode*. De manera similar a la inversión de la imagen sobre Y, se define la imagen *mMode2* que repite sobre sí misma a *mMode* a lo largo del eje X. Tanto *mModeFlipY* como *mMode2* tienen el mismo tipo imagen que *mMode*.

Figura 6.7. Interfaz completa del modo M en Qt Creator.

El checkbox para invertir la imagen se llama *invertImageBox* y se conecta con la función *invertImageBoxClicked* de la clase *MmodeWidget*, esta solo se encarga de intercambiar entre la imagen del modo M con su imagen inversa. Para repetir el modo M se usa el spinbox *repeatBox* que se comunica con *valueChange* de *MmodeWidget*, aquí se generan las imágenes repetida e invertida del modo M.

Para identificar y seleccionar los índices de las secuencias de inicio y fin de una sección del modo M se ocupa el botón *selectPeriodBtn* asociado a la función *selectPeriodBtnClicked* de *MmodeWidget.* Al activarse permite seleccionar un par de posiciones con el mouse, para lo cual se ayuda de el método *vtkPropPicker*. Por cada clic se crea una de línea vertical usando el método *vtkLineSource* y se dibuja gracias a *vtkActor*. Las locaciones se ordenan con el tamaño de los índices las secuencias a los que corresponden, de menor a mayor, para que junto con el

número de repeticiones del modo M, se envíen a la clase *MainWindow* mediante la función *setPeriod* de esta última.

6.5. Transformada Rápida de Hermite

Siguiendo el método de M. Hashimoto y J. Sklansky [3], descrito en el Capítulo 4.4., se desarrolló el árbol completo de derivación para $N=3$ mostrado en el anexo A. Ya que las derivadas resultantes que se repiten son equivalentes, se selecciona el camino más corto para implementar las derivadas que se desean . Como resultado se obtiene el algoritmo mostrado en la Figura 6.8.

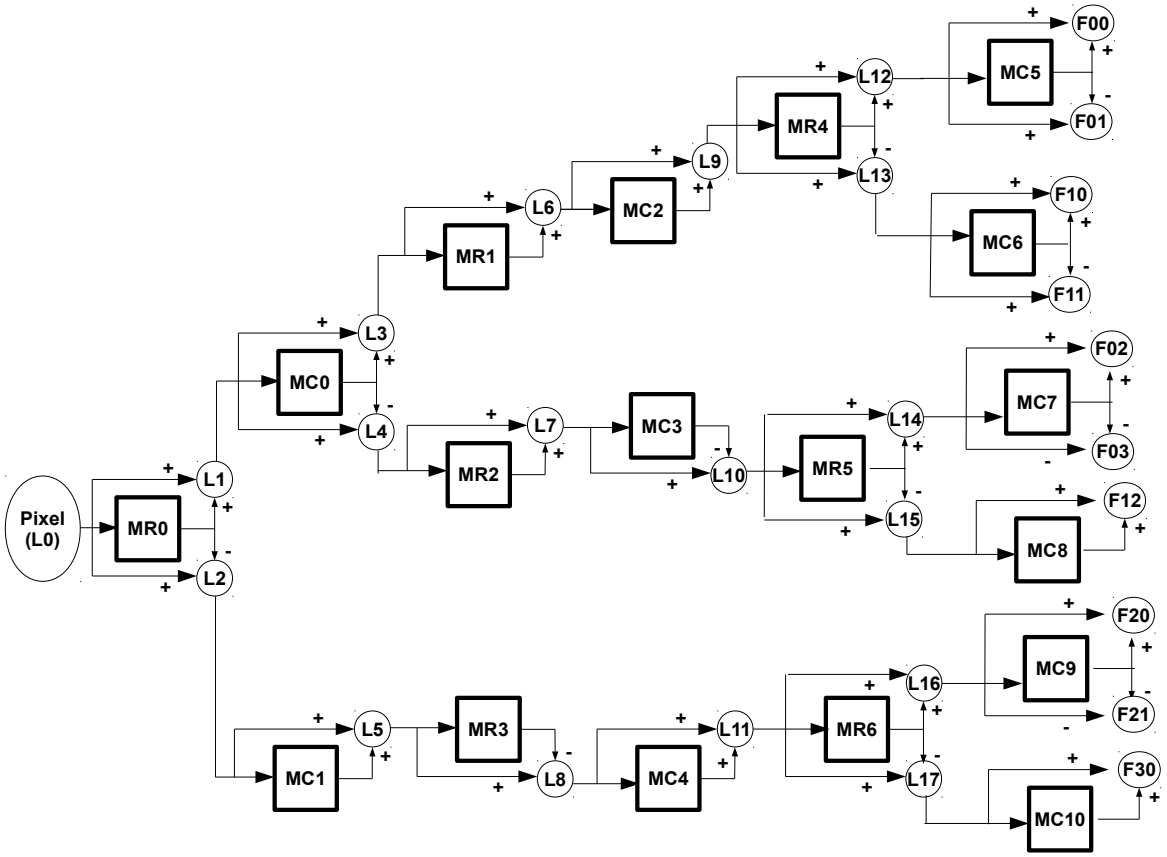


Figura 6.8. Algoritmo para la estimación de derivadas parciales en dos dimensiones de primer, segundo y tercer orden con $N=3$.

donde:

- MR = Registros de almacenamientos de retraso de renglones.
- MC = Registros de almacenamientos de retraso de columnas.

- L = Registros de resultados intermedios, siendo L0 el valor del píxel a operar.
- F = derivadas parciales del pixel operado (coeficientes de la Transformada de Hermite).

En el algoritmo entran los pixeles de uno en uno de forma ordenada y requieren de su antecesor para operarse por lo que al implementar este algoritmo se necesita agregar previamente un margen de mínimo 2 píxeles de ancho como para que los píxeles en el borde de la imagen tengan con que operarse. Las derivadas resultantes normalizan sus valores al dividirse entre el número resultante de elevar dos al número de adiciones efectuadas para esa derivada, utilizando la nomenclatura del diagrama de la Figura 6.8., esto es:

$$coef_{00}=\frac{F_{00}}{64}, coef_{10}=\frac{F_{10}}{32}, coef_{01}=\frac{F_{01}}{32}, coef_{20}=\frac{F_{20}}{16}, coef_{11}=\frac{F_{11}}{16},$$
$$coef_{02}=\frac{F_{02}}{16}, coef_{03}=\frac{F_{03}}{8}, coef_{30}=\frac{F_{30}}{8}, coef_{12}=\frac{F_{12}}{8}, coef_{21}=\frac{F_{21}}{8}$$

El margen que se aplica es de 2 píxeles de ancho en la parte superior y en el lado izquierdo, mientras que 3 píxeles en los otros dos bordes. Este margen está compuesto con la información del borde de la imagen de tal manera que este se repite hacia los extremos y las esquinas que se generan tienen la misma información que la del borde correspondiente, como se muestra en la Figura 6.9.

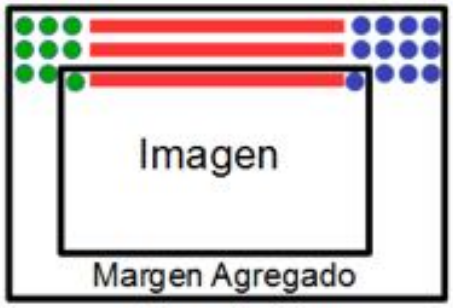


Figura 6.9. Diagrama de la implementación del margen.

Las derivadas obtenidas, respecto a la imagen original, quedan recorridas 3 píxeles a la derecha y 3 hacia abajo. Al eliminar los píxeles recorridos, el tamaño de la derivada queda con un píxel extra en el largo y ancho en comparación con el tamaño de la imagen original. Esto se resuelve al aplicar un filtro promediador en dirección X y en dirección Y, lo que sería equivalente a aplicar a cada coeficiente un registro de almacenamientos de retraso de renglones y otro de columnas haciendo exclusivamente la operación de adición y de la misma forma, se normaliza dependiendo del número de adiciones, que para este caso son 2 por lo

que la división del promedio es entre 4. En la Figura 6.10. se muestra el filtro promediador con la división incluida y en la 6.11., un ejemplo del resultado del coeficiente 00 al que se le aplica el recorte y promediación descritos.

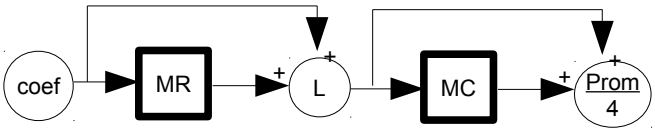


Figura 6.10. Diagrama del filtro promediador normalizado.

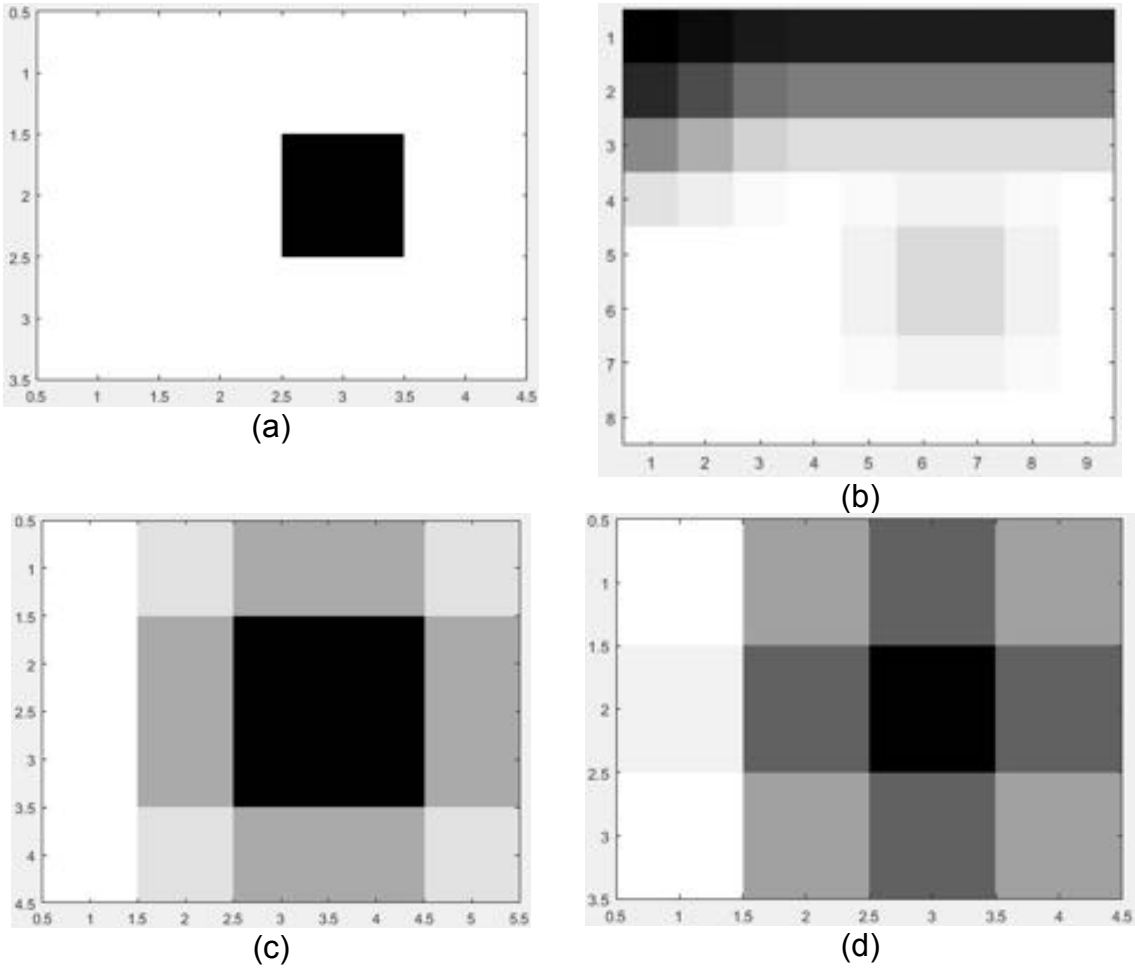


Figura 6.11. Derivada 00 de una imagen de 3x4. (a) Imagen original. (b) Derivada 00 sin recortar ni promediar de (a) con el padding. (c) Imagen de (b) recortada sin promediar. (d) Imagen de (b) promediada y recortada.

De esta manera, el procedimiento seguido para la obtención de los coeficientes de Hermite es:

1. Agregar el margen a la imagen.
2. Aplicar el algoritmo de la Transformada rápida de Hermite con $N=3$.
3. Normalizar cada coeficiente.
    1. Calcular el resultado de dos elevado al número de adiciones hechas.
    2. Dividir el coeficiente entre el resultado del punto anterior.
4. Implementar el filtro promediador normalizado en los coeficientes normalizados.
5. Recortar los coeficientes.

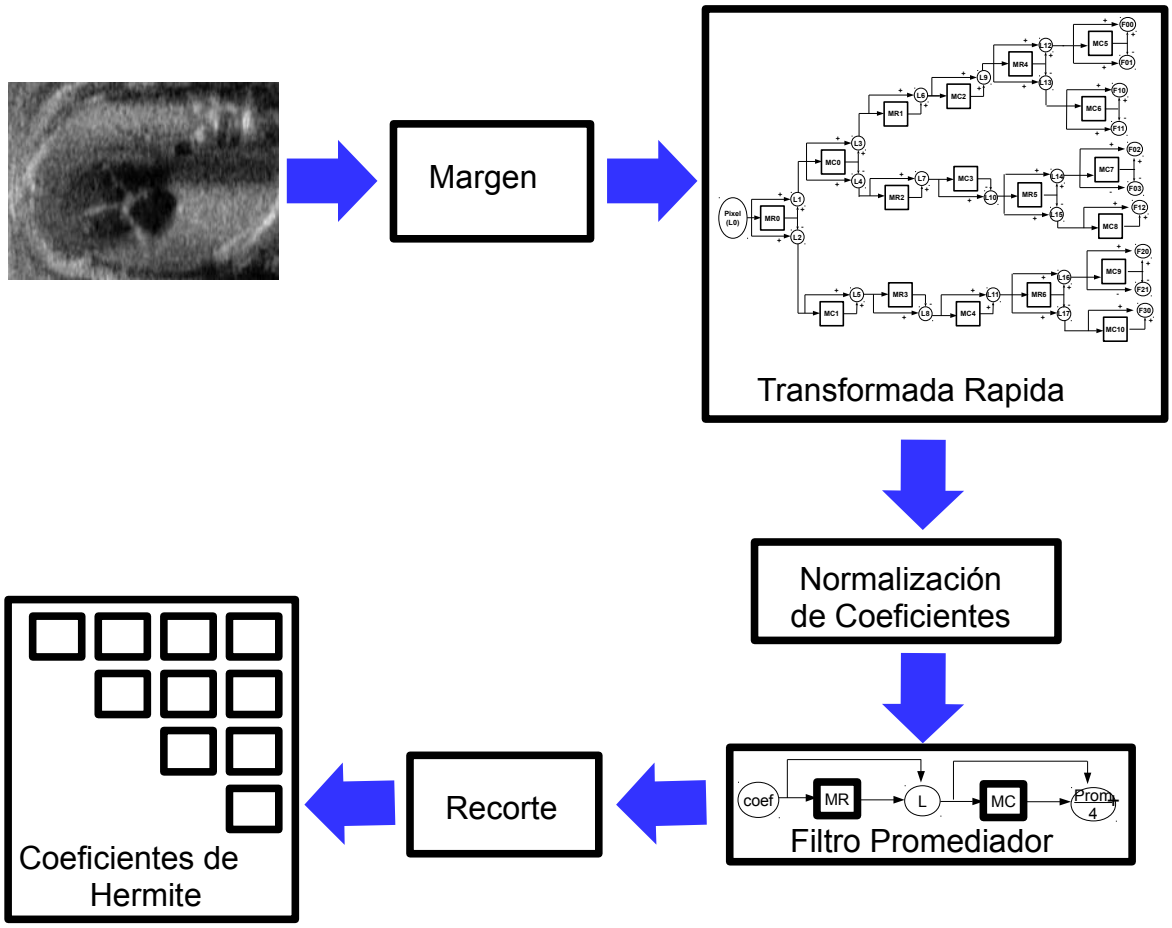


Figura 6.12. Diagrama del proceso aplicado para la obtención los coeficientes de Hermite de tercer orden.

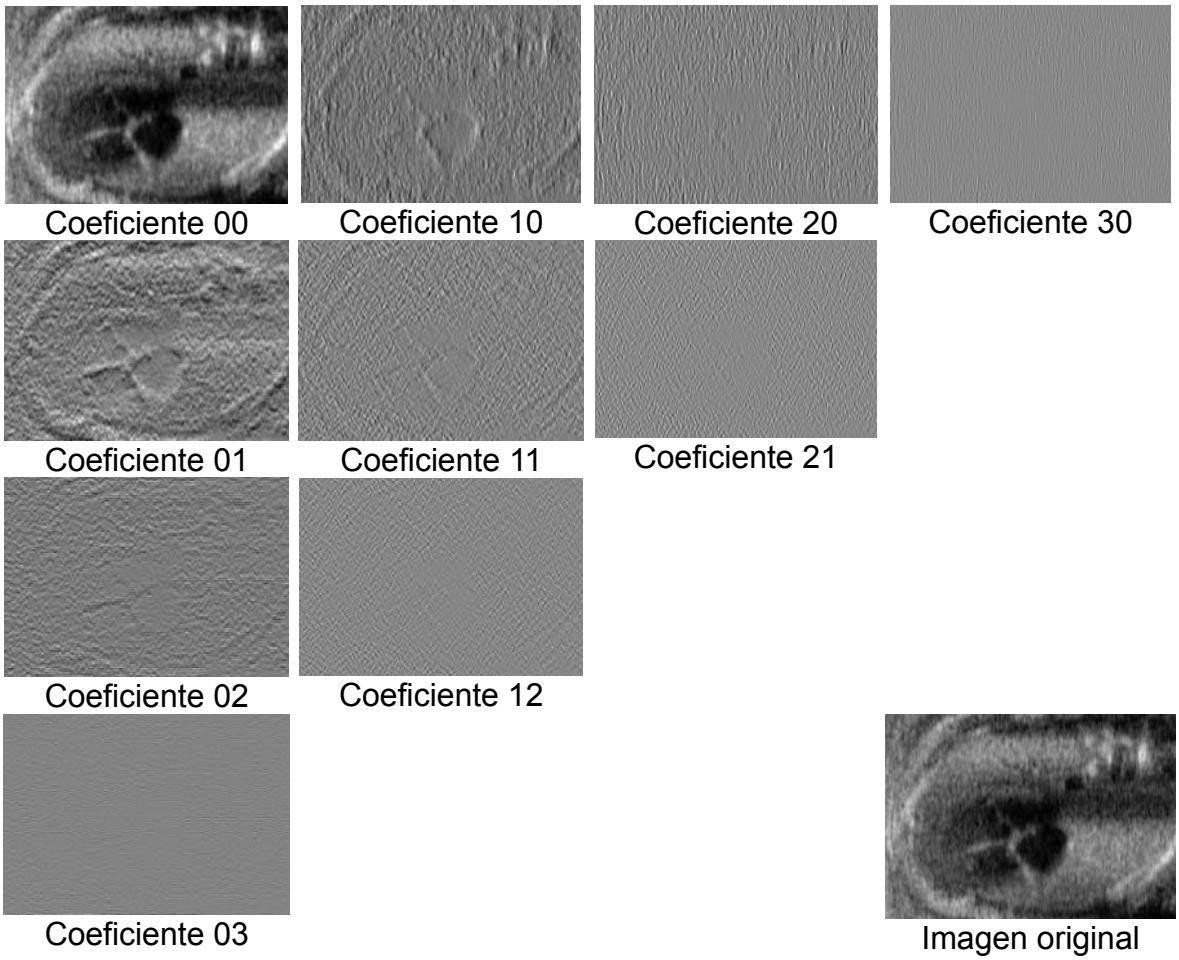


Figura 6.13. Coeficientes de la Transformada de Hermite de una ecocardiografía con una normalización en el rango de 0 a 255 niveles de gris.

6.5.1. Rotación de los Coeficientes de Hermite

Se aplica la ecuación (4.52) a los coeficientes de Hermite con el ángulo de máxima energía descrito en la ecuación (4.55).

Tomando los coeficientes de Hermite resultantes de la aplicación del algoritmo de la Transformada Rápida de Hermite de orden 3 con las observaciones mencionadas para obtenerlos, los coeficientes rotados 10, 20 y 30 se calculan como:

- $F_{10,\theta} = F_{10} \cdot g_{10}(\theta) + F_{01} \cdot g_{01}(\theta)$,
- $F_{20,\theta} = F_{20} \cdot g_{20}(\theta) + F_{11} \cdot g_{11}(\theta) + F_{02} \cdot g_{02}(\theta)$,
- $F_{30,\theta} = F_{30} \cdot g_{30}(\theta) + F_{21} \cdot g_{21}(\theta) + F_{12} \cdot g_{12}(\theta) + F_{03} \cdot g_{03}(\theta)$,

donde los coeficientes de Hermite obtenidos con el algoritmo rápido (F_{10} , F_{01} , F_{20} , F_{11} , F_{02} , F_{30} , F_{21} , F_{12} , F_{03}) están normalizados, promediados y recortados. El ángulo de máxima energía queda como:

$$\theta = \frac{arg[F_{01}, F_{10}]}{2}$$

y la componente direccional $g_{m,n-m}(\theta)$:

- $g_{10}(\theta) = \cos(\theta)$.
- $g_{01}(\theta) = \sin(\theta)$.
- $g_{20}(\theta) = \cos^2(\theta)$.
- $g_{11}(\theta) = 1.4142\cos(\theta)\cdot\sin(\theta)$.
- $g_{02}(\theta) = \sin^2(\theta)$.
- $g_{30}(\theta) = \cos^3(\theta)$.
- $g_{12}(\theta) = 1.7321\cos(\theta)\cdot\sin^2(\theta)$.
- $g_{21}(\theta) = 1.7321\cos^2\theta\cdot\sin(\theta)$.
- $g_{03}(\theta) = \sin^3(\theta)$.

En la Figura 6.14. se muestra un diagrama de este proceso y en la Figura 6.15. los coeficientes rotados de Hermite de la imagen original de la Figura 6.13. Igual que con los coeficientes de Hermite de ésta última Figura, los de la 6.15. se normalizaron en el rango de 0 a 255 niveles de gris para visualizarlos, pero cuando son utilizados en algún procesamiento o cálculo, la normalización de niveles de gris no se aplica.

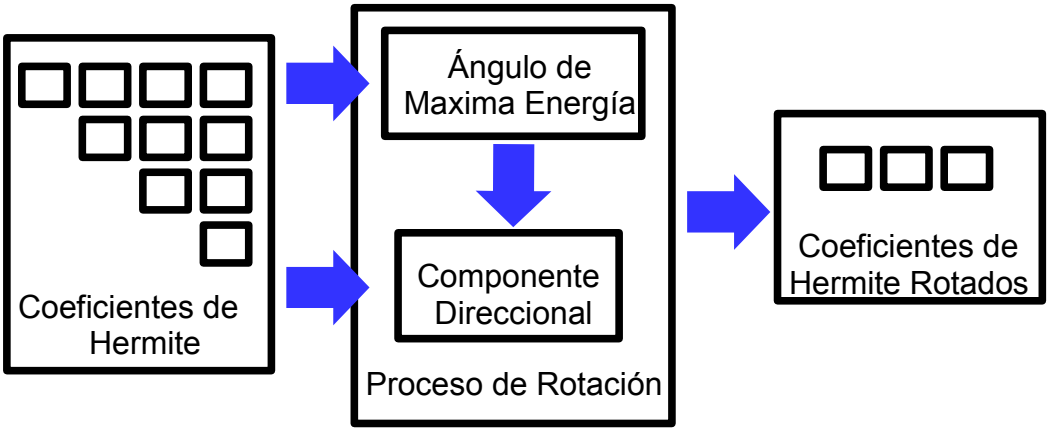
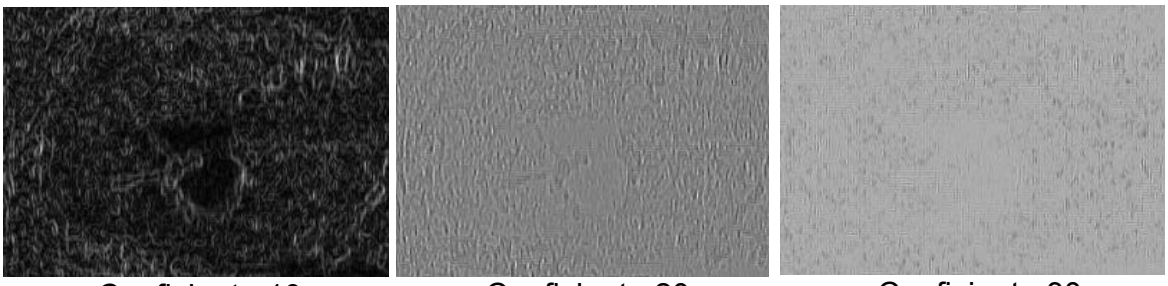


Figura 6.14. Diagrama del proceso de obtención los coeficientes de Hermite rotados de tercer orden.



Coeficiente 10 Coeficiente 20 Coeficiente 30

Figura 6.15. Coeficientes Rotados de la Transformada de Hermite de la imagen original que aparece en la Figura 6.13. normalizados en el rango de 0 a 255 niveles de gris.

6.6. Flujo Óptico

El flujo óptico entre dos imágenes consecutivas se obtiene de la resolución del funcional descrito por la ecuación (5.55), obteniendo los valores de u y v en escalas sucesivas de mayor a menor. Se inicializan los valores de u y v en 0 y se realiza la primera estimación con las imágenes 1 y 2 , y las matrices con los valores de u y v encogidas. Con las reducciones para las estimaciones siguientes se toman las soluciones obtenidas de la escala previa para actualizar u y v , e interpolar la escala de la imagen 2 con u y v . Las variables u y v corresponden a la estimación de movimiento de cada píxel en la imagen, u para la coordenada X, y v para la coordenada Y.

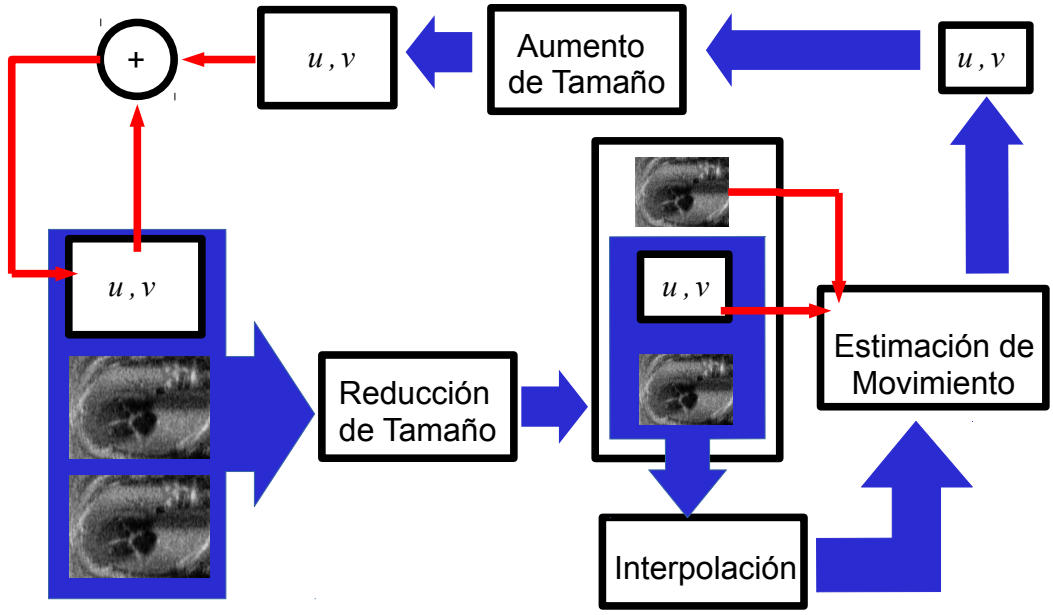


Figura 6.16. Diagrama global del proceso de estimación de movimiento.

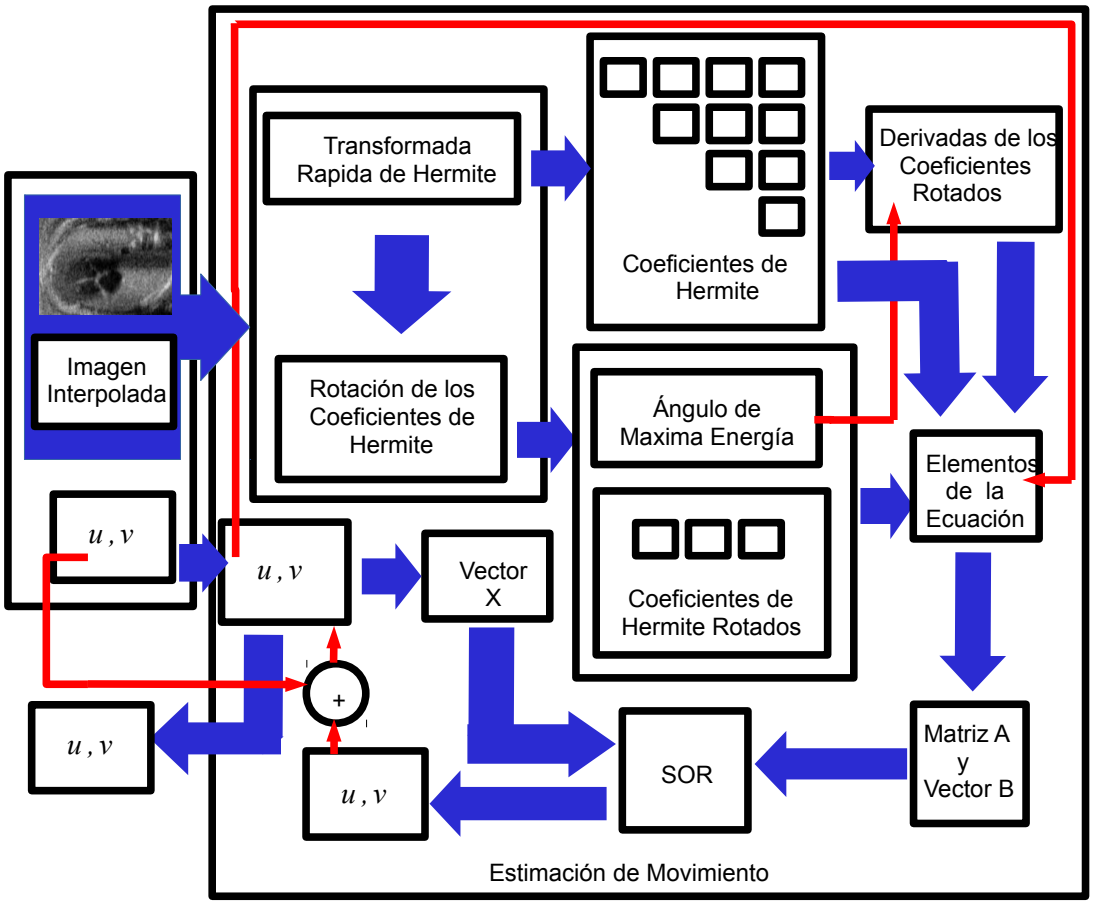


Figura 6.17. Diagrama del funcionamiento del módulo de estimación de movimiento de la Figura 6.17.

Los pasos generales que sigue todo el procedimiento del estimación de movimiento son expuestos de forma gráfica en el diagrama de la Figura 6.16 y la Figura 6.17., estos son:

1. Inicializar los valores de u y v con ceros.

2. Reducir el tamaño del par de imágenes, u y v con el factor de escala que se obtiene de: $constante\ de\ escalamiento^{numero\ de\ escalas}$.

3. Calcular la estimación de movimiento entre las imágenes escaladas.

   a) Obtener los coeficientes de Hermite hasta nivel 3 y los coeficientes de Hermite rotados hasta nivel 2.

b) Generar las derivadas de los coeficientes de Hermite rotados.
c) Calcular la restricción de regiones homogéneas para los coeficientes de Hermite.
d) Producir la restricción para el suavizado.

e) Organizar la ecuación (5.55) de la forma $Ax-b=0$, donde A es una matriz mientras que x y b son vectores.

1. Generar el vector b con las partes constantes de la ecuación, intercalando las que pertenecen a u con las que pertenecen a v .

2. Definir el vector x acomodando los valores que se tienen de u y v escalados, de la misma forma que el vector b .

3. Organizar las variables que aparecen en la ecuación (5.55) con respecto a sus relación con u y v para generar la matriz A .

f) Aplicar el método de solución de Ecuaciones SOR.

g) Sumar u y v con las soluciones asociadas, obtenidas por el método SOR.

h) Repetir desde paso e), 0 o más veces, según se crea más conveniente.

4. Actualizar u y v con los resultados del paso anterior.

a) Aumentar el tamaño de los resultados para que tengan el mismo tamaño que las imágenes introducidas.

b) Sumar el resultado agrandado con u y v .

5. Reducir el tamaño del par de imágenes de entrada, u y v con el factor de escala: $constante\ de\ escalamiento^{numero\ de\ escalas-escala\ actual}$.

6. Interpolar la reducción de la segunda imagen con los desplazamientos de u y v sobre la posición del píxel respectivo.

7. Repetir desde paso 3 mientras que $numero\ de\ escalas-1>0$.

La estimación de movimiento se devuelve mediante dos imágenes 2D de ITK, una para u y otra para v

6.6.1. Derivadas de los Coeficientes Rotados de Hermite

La estimación de movimiento dada por la ecuación (5.55) requiere de las derivadas parciales de los coeficientes rotados de Hermite de la segunda imagen, definidas por la ecuación (5.42). Para su obtención se necesitan los coeficientes de Hermite de un nivel superior al nivel que corresponde la derivada. En este caso, para obtener las de nivel 2 se necesitan los coeficientes rotados de nivel 3.

Desarrollando las derivadas parciales $l_{n\theta_{(m)+1}}$ y $l_{n\theta_{(n)+1}}$ para nivel 1 y 2 con los elementos para el cálculo de las rotaciones de los coeficientes de Hermite, quedan como:

- $F_{10,\theta_{(n)+1}} = F_{11}\cdot g_{10}(\theta) + F_{20}\cdot g_{01}(\theta)$,
- $F_{20,\theta_{(n)+1}} = F_{12}\cdot g_{20}(\theta) + F_{21}\cdot g_{11}(\theta) + F_{30}\cdot g_{02}(\theta)$,
- $F_{10,\theta_{(m)+1}} = F_{02}\cdot g_{10}(\theta) + F_{11}\cdot g_{01}(\theta)$,
- $F_{20,\theta_{(m)+1}} = F_{03}\cdot g_{20}(\theta) + F_{12}\cdot g_{11}(\theta) + F_{21}\cdot g_{02}(\theta)$,

de forma más explícita:

- $R_{1n1} = \cos(\theta) F_{11} + \sin(\theta) F_{20}$,
- $R_{2n1} = \cos^2(\theta) F_{12} + 1.4142\cos(\theta)\sin(\theta) F_{21} + \sin^2(\theta) F_{30}$,
- $R_{1m1} = \cos(\theta) F_{02} + \sin(\theta) F_{11}$,
- $R_{2m1} = \cos^2(\theta) F_{03} + 1.4142\cos(\theta)\sin(\theta) F_{12} + \sin^2(\theta) F_{21}$,

donde θ es el ángulo de máxima energía calculado en la rotación de los coeficientes de Hermite.

6.6.2. Función de Regularización Isotrópica

Para el cálculo de flujo óptico realizado en [2] la función Ψ' se aplica como:

$$\Psi'(s) = \frac{1}{max\left(min\left(2\sqrt{s}, \frac{1}{\varepsilon}\right), \varepsilon\right)} \quad .$$

Ya que se puede aplicar píxel a píxel, las variables de la clase i*tkHermiteOpticalFlow* que la utilizan, hacen la operación directamente sin llamar a ninguna función pero se da la opción de modificar de forma global el valor de ε .

6.6.3. Restricción para el Suavizado

Con base en el algoritmo para el cálculo de las derivadas parciales descrito en el Capítulo 4.4., para los elementos del suavizado en el flujo óptico, se realizó uno que calcula las primeras derivadas sobre las coordenadas X e Y, y a ambos casos se le aplica un filtro promedio en ambas direcciones.

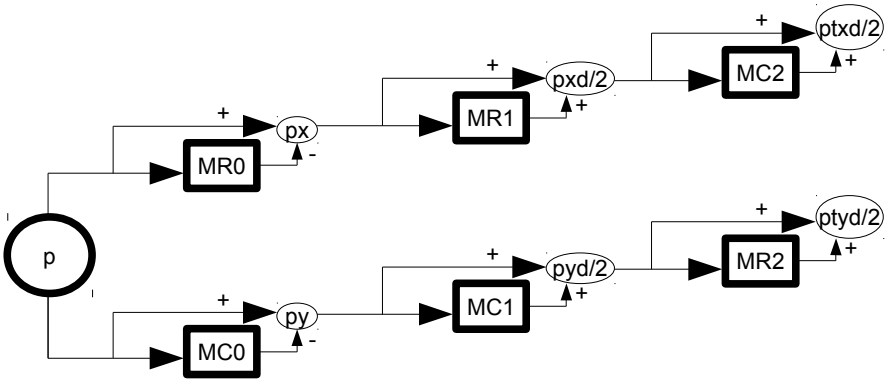


Figura 6.18. Algoritmo para el cálculo de los elemento del suavizado.

donde:

- p es el valor de u o v .
- px es la derivada parcial de p sobre X.
- py es la derivada parcial de p sobre Y.
- pxd es el valor promedio de px sobre X.
- pyd es el valor promedio de py sobre Y.
- ptxd es el valor promedio de pxd sobre Y.
- ptyd es el valor promedio de pyd sobre X.
- La división entre 2 es para normalizar el resultado de la suma.

Los elementos correspondientes a X y Y se calcula como:

- $\frac{\partial f}{\partial x} = (ux^2 + utyd^2) + (vx^2 + vtyd^2)$ y
- $\frac{\partial f}{\partial y} = (uy^2 + utxd^2) + (vy^2 + vtxd^2)$,

donde p ya está sustituida por u o v .

La aplicación del algoritmo ocupa un padding del tamaño de dos píxeles en cada extremo para tener elementos para operar los valores con algún índice 0 y debido a los desfases que esto ocasiona. De forma similar a lo que sucede con el

algoritmo para encontrar los coeficientes de Hermite. Las sumas para calcular $\frac{\partial f}{\partial x}$ y $\frac{\partial f}{\partial y}$ se realiza con las matrices recortadas entre las coordenadas:

- (2,2) y (X-1,Y-2) para px.
- (2,2) y (X-2,Y-1) para py.
- (3,2) y (X-1,Y-1) para ptxd.
- (2,3) y (X-1,Y-1) para ptyd.

donde X y Y son los índices del elemento más extremo con el padding incluido.

El divergente es aproximado con la suma de los valores de $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$ y de los valores desfasados un lugar en X en $\frac{\partial f}{\partial x}$ y uno en Y en $\frac{\partial f}{\partial y}$

6.6.4. SOR [43]

El método de sobrerrelajación sucesiva para la resolución de ecuaciones diferenciales, abreviado como SOR por sus siglas en inglés (Successive Over Relaxation), parte de que para un enorme sistema lineal no singular de la forma:

$$Ax=b \quad , \quad A\in\mathbb{C}^{n,n} \quad , \quad b\in\mathbb{C}^{n}\setminus\{0\} \quad , \tag{6.1}$$

considerando métodos iterativos basados en una separación de la matriz A , como:

$$A=M-N \tag{6.2}$$

donde M es la matriz de precondicionamiento y se le considera invertible y barato de invertir. Haciendo que el sistema de ecuaciones que emplea un término matricial M sea más económico de resolver que (6.1). En base a (6.2) en forma de punto fijo (6.1) se escribe:

$$x=Tx+c \quad , \quad T:=M^{-1}N \quad , \quad c:=M_{-1}b \quad , \tag{6.3}$$

lo que produce el siguiente esquema iterativo para solucionar a (6.1):

$$x^{(m+1)}=Tx^{m}+c \quad , \quad m=0,1,2,\ldots, \quad \text{y} \quad x^{(0)}\in\mathbb{C}^{n} \quad \text{arbitrario} \tag{6.4}$$

Una condición suficiente y necesaria para que (6.4) converja en la solución de (6.1) es $\rho(T)<1$, donde $\rho(.)$ denota radio espectral, mientras que una condición suficiente para converger es $\|T\|<1$, donde $\|.\|$ denota norma matricial inducida por una norma vectorial.

Partiendo de los métodos iterativos clásicos se escribe $A=D-L-U$, con $D=diag(A)$, asumiendo que $det(D)\neq 0$, L y U son las matrices inferior y superior triangular, respectivamente. El método iterativo de sobrerrelajación sucesiva $(M\equiv(1/\omega)(D-\omega L))$ está definido por:

$$\begin{gathered} x^{(m+1)}=T_{\omega}x^{(m)}+c_{\omega} \quad , \\ T_{\omega}:=(D-\omega L)^{-1}[(1-\omega)D-\omega U] \quad , \\ c_{\omega}:=\omega(D-\omega L)^{-1}b \quad , \end{gathered} \tag{6.5}$$

donde $\omega\in\mathbb{C}\setminus\{0\}$ es el factor de relajación (o parámetro de sobrerrelajación), siendo $|\omega-1|<1$ una condición para la convergencia del método [44].

6.6.4.1. Generación del Vector b y la Matriz A

Para organizar los elementos de la ecuación (5.55) de la forma $Ax-b=0$, se agrupan en:

- Constante para u :

$$\begin{gathered} \Psi'\left(\left|L_0(X+w^k)-L_0(X)+du^{k,q}L_{10}(X+w^k)+dv^{k,q}L_{01}(X+w^k)\right|^2+\right. \\ \left.\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w^k)-l_{n\theta}(X)+du^{k,q}l_{n\theta_{(m)+1}}(X+w^k)+dv^{k,q}l_{n\theta_{(n)+1}}(X+w^k)\right|^2\right)\right)\cdot \\ \left[\left|L_0(X+w^k)-L_0(X)\right|L_{10}(X+w^k)+\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w^k)-l_{n\theta}(X)\right|\cdot l_{n\theta_{(m)+1}}(X+w^k)\right)\right]- \\ \alpha\, div\left(\Psi'\left(\left|\nabla(u^k+du^k)\right|^2\right)\right). \end{gathered}$$

- Constante para v :

$$\Psi'\left(\left|L_0(X+w^k)-L_0(X)+du^{k,q}L_{10}(X+w^k)+dv^{k,q}L_{01}(X+w^k)\right|^2+\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w^k)-l_{n\theta}(X)+du^{k,q}l_{n\theta_{(m)+1}}(X+w^k)+dv^{k,q}l_{n\theta_{(n)+1}}(X+w^k)\right|^2\right)\right)\cdot\left[\left|L_0(X+w^k)-L_0(x)\right|L_{01}(X+w^k)+\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w^k)-l_{n\theta}(X)\right|\cdot l_{n\theta_{(n)+1}}(X+w^k)\right)\right]-\alpha\, div\left(\Psi'\left(\left|\nabla(v^k+dv^k)\right|^2\right)\right).$$

- Primera ecuación de Euler-Lagrange para u :

$$\Psi'\left(\left|L_0(X+w^k)-L_0(X)+du^{k,q}L_{10}(X+w^k)+dv^{k,q}L_{01}(X+w^k)\right|^2+\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w^k)-l_{n\theta}(X)+du^{k,q}l_{n\theta_{(m)+1}}(X+w^k)+dv^{k,q}l_{n\theta_{(n)+1}}(X+w^k)\right|^2\right)\right)\cdot\left[L_{10}(X+w^k)\cdot L_{10}(X+w^k)+\gamma\left(\sum_{n=1}^{N}l_{n\theta_{(m)+1}}(X+w^k)\cdot l_{n\theta_{(m)+1}}(X+w^k)\right)\right]-\alpha\, div\left(\Psi'\left(\left|\nabla(u^k+du^k)\right|^2+\left|\nabla(v^k+dv^k)\right|^2\right)\right)+\beta\frac{1}{\sqrt{\left(L_0(X+w^k)-L_0(X)\right)^2+\left(L_{10}(X+w^k)-L_{10}(X)\right)^2+\left(L_{01}(X+w^k)-L_{01}(X)\right)^2}}.$$

- Segunda ecuación de Euler-Lagrange para u :

$$\Psi'\left(\left|L_0(x+w^k)-L_0(x)+du^{k,q}L_{10}(x+w^k)+dv^{k,q}L_{01}(x+w^k)\right|^2+\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(x+w^k)-l_{n\theta}(x)+du^{k,q}l_{n\theta_{(m)+1}}(x+w^k)+dv^{k,q}l_{n\theta_{(n)+1}}(x+w^k)\right|^2\right)\right)\cdot\left[L_{10}(x+w^k)\cdot L_{01}(x+w^k)+\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta_{(m)+1}}(x+w^k)\right|\cdot l_{n\theta_{(n)+1}}(x+w^k)\right)\right].$$

- Primera ecuación de Euler-Lagrange para v :

$$\Psi'\left(\left|L_0(x+w^k)-L_0(x)+du^{k,q}L_{10}(x+w^k)+dv^{k,q}L_{01}(x+w^k)\right|^2+\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(x+w^k)-l_{n\theta}(x)+du^{k,q}l_{n\theta_{(m)+1}}(x+w^k)+dv^{k,q}l_{n\theta_{(n)+1}}(x+w^k)\right|^2\right)\right)\cdot\left[L_{10}(x+w^k)\cdot L_{01}(x+w^k)+\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta_{(m)+1}}(x+w^k)\right|\cdot l_{n\theta_{(n)+1}}(x+w^k)\right)\right].$$

Esta ecuación es igual a la segunda ecuación de Euler-Lagrange para u .

- Segunda ecuación de Euler-Lagrange para v :

$$\Psi'\left(\left|L_0(X+w^k)-L_0(X)+du^{k,q}L_{10}(X+w^k)+dv^{k,q}L_{01}(X+w^k)\right|^2+\gamma\left(\sum_{n=1}^{N}\left|l_{n\theta}(X+w^k)-l_{n\theta}(X)+du^{k,q}l_{n\theta_{(m)+1}}(X+w^k)+dv^{k,q}l_{n\theta_{(n)+1}}(X+w^k)\right|^2\right)\right)\cdot\left[L_{01}(X+w^k)\cdot L_{01}(X+w^k)+\gamma\left(\sum_{n=1}^{N}l_{n\theta_{(n)+1}}(X+w^k)\cdot l_{n\theta_{(n)+1}}(X+w^k)\right)\right]-\alpha\, div\left(\Psi'\left(\left|\nabla(u^k+du^k)\right|^2+\left|\nabla(v^k+dv^k)\right|^2\right)\right)+\beta\frac{1}{\sqrt{\left(L_0(X+w^k)-L_0(X)\right)^2+\left(L_{10}(X+w^k)-L_{10}(X)\right)^2+\left(L_{01}(X+w^k)-L_{01}(X)\right)^2}}.$$

El vector b se llena intercalando uno por uno los elementos constantes de u con los de v y son tomados respecto al píxel al que se asocia, recorriendo columna por columna y comenzando en la coordenada (0,0). El tamaño final del vector b es el doble de la cantidad de los píxeles contenidos en alguno de los cortes reducidos y a los valores se guardan en él se les invierte su signo para coincidir con el acomodo $Ax-b=0$, así el vector b queda de la forma:

$$b=[-const_u 0,-const_v 0,-const_u 1,-const_v 1,\cdots,-const_u n,-const_v n] \quad ,$$

donde *const* se refiere a alguno de los elementos constantes, se indica con un subíndice si pertenece a u o v y n es el número de píxeles en cada uno de los cortes utilizados.

De igual forma se organizan los valores escalados de u y v actuales en el vector x .

La matriz A es cuadrada, del mismo tamaño que el vector b en ancho y en largo. En ella se encuentran esparcidos las primeras y segundas ecuaciones de Euler-Lagrange, y los valores de $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$ y sus desfases, utilizados en el cálculo de la restricción de suavizado.

De forma similar al acomodo de los elementos constantes de u y v dentro del vector b , cada uno de los valores de cada elemento, está asociado a las coordenadas de un píxel en ambos cortes escalados. La asignación con respecto a la posición de los píxeles, se genera renglón por renglón comenzando en la coordenada (0,0).

El llenado de la matriz se hace de la siguiente manera:

$$A(col, ren) = val \quad ,$$

donde val es un valor de la matriz cuadrada A ,localizado en la columna col y en el renglón reg . El ordenamiento depende del recorrido sobre los renglones, de tal manera que para el píxel i con $i = 0,1,2,\ldots,(ht \cdot wt) - 1$, siendo ht y wt la altura y la anchura en píxeles del corte escalado, se tienen:

- $ren = 2i$, para el cual:
  - $col = ren - 2ht$ y $val = -\frac{\partial f}{\partial x}$.
  - $col = ren - 2$ y $val = -\frac{\partial f}{\partial x}$.
  - $col = ren$ y $val = -\frac{\partial f}{\partial y}$.
  - $col = ren + 1$ y $val = -\frac{\partial f}{\partial y}$.
  - $col = ren + 2$ y val es el resultado de la primera ecuación de Euler-Lagrange para u.
  - $col = ren + 2ht$ y val es el resultado de la segunda ecuación de Euler-Lagrange para u.
- $ren = 2i + 1$, para el cual:

- $col = ren - 2ht$ y val es el resultado de la segunda ecuación de Euler-Lagrange para v.
- $col = ren - 2$ y val es el resultado de la primera ecuación de Euler-Lagrange para v.
- $col = ren - 1$ y val es el valor desfasado de $-\frac{\partial f}{\partial y}$.
- $col = ren$ y val es el valor desfasado de $-\frac{\partial f}{\partial y}$.
- $col = ren + 2$ y val es el valor desfasado de $-\frac{\partial f}{\partial x}$.
- $col = ren + 2ht$ y val es el valor desfasado de $-\frac{\partial f}{\partial x}$.

Asegurándose de que para poderle asignar a val algún valor, se debe cumplir que $col \geq 0$ para que esté dentro de la matriz.

Siendo:

- $\frac{\partial f_d}{\partial x}$ y $\frac{\partial f_d}{\partial y}$ corresponden respectivamente a $\frac{\partial f}{\partial x}$ y $\frac{\partial f}{\partial y}$ desfasados.
- EL se refiere a las ecuaciones de Euler-Lagrange con subíndice de la variable a la que pertenece en minúscula (u o v) e indicando si es la primera o la segunda ecuación.

Los distintos elementos de la matriz se posicionan diagonalmente, con las diagonales de $\frac{\partial f_d}{\partial x}$ y $\frac{\partial f}{\partial x}$ iniciando respectivamente a la mitad del primer renglón y a la mitad de la primera columna de la matriz, correspondiente al índice $ht \cdot wt$. La matriz se ve de la siguiente manera:

$$
\begin{array}{cccccccccc}
ELu_1 & ELv_1 & -\frac{\partial f_d}{dy} & 0 & \cdots & 0 & -\frac{\partial f_d}{dx} & 0 & \cdots & 0 \\
ELu_2 & ELv_2 & 0 & -\frac{\partial f_d}{dy} & 0 & \cdots & 0 & -\frac{\partial f_d}{dx} & \ddots & \vdots \\
-\frac{\partial f}{dy} & 0 & ELu_1 & ELv_1 & -\frac{\partial f_d}{dy} & 0 & \cdots & \ddots & -\frac{\partial f_d}{dx} & 0 \\
0 & -\frac{\partial f}{dy} & ELu_2 & ELv_2 & 0 & -\frac{\partial f_d}{dy} & 0 & \vdots & 0 & -\frac{\partial f_d}{dx} \\
\vdots & 0 & -\frac{\partial f}{dy} & 0 & \ddots & \ddots & -\frac{\partial f_d}{dy} & \ddots & \vdots & 0 \\
0 & \vdots & 0 & -\frac{\partial f}{dy} & \ddots & \ddots & 0 & -\frac{\partial f_d}{dy} & 0 & \vdots \\
-\frac{\partial f}{dx} & 0 & \vdots & 0 & -\frac{\partial f}{dy} & 0 & ELu_1 & ELv_1 & -\frac{\partial f_d}{dy} & 0 \\
0 & -\frac{\partial f}{dx} & \ddots & \cdots & \ddots & -\frac{\partial f}{dy} & ELu_2 & ELv_2 & 0 & -\frac{\partial f_d}{dy} \\
\vdots & \ddots & -\frac{\partial f}{dx} & 0 & \cdots & 0 & -\frac{\partial f}{dy} & 0 & ELu_1 & ELv_1 \\
0 & \cdots & 0 & -\frac{\partial f}{dx} & 0 & \cdots & 0 & -\frac{\partial f}{dy} & ELu_2 & ELv_2
\end{array}
$$

6.6.4.2. Aplicación del Método

La matriz A definida como $A=D-L-U$ siendo D su diagonal principal, y L y U sus matrices inferior y superior triangular, se separa en dos matrices triangulares M y N donde $M=D+\omega L$ y $N=(1-\omega)D-\omega U$, siendo ω el parámetro de sobrerrelajación.

Con las matrices M y N y los vectores b y x se construye el nuevo sistema lineal $Mx'=Nx+\omega b$ para encontrar al vector x' mediante la descomposición LU . Esto se repite actualizando x como $x=x'$ hasta que el error entre x y x' , calculado como $error=\frac{||(x'-x)||}{||x'||}$ donde $||.||$ denota norma del vector, esté dentro de una tolerancia o se llegue a un cierto número de repeticiones.

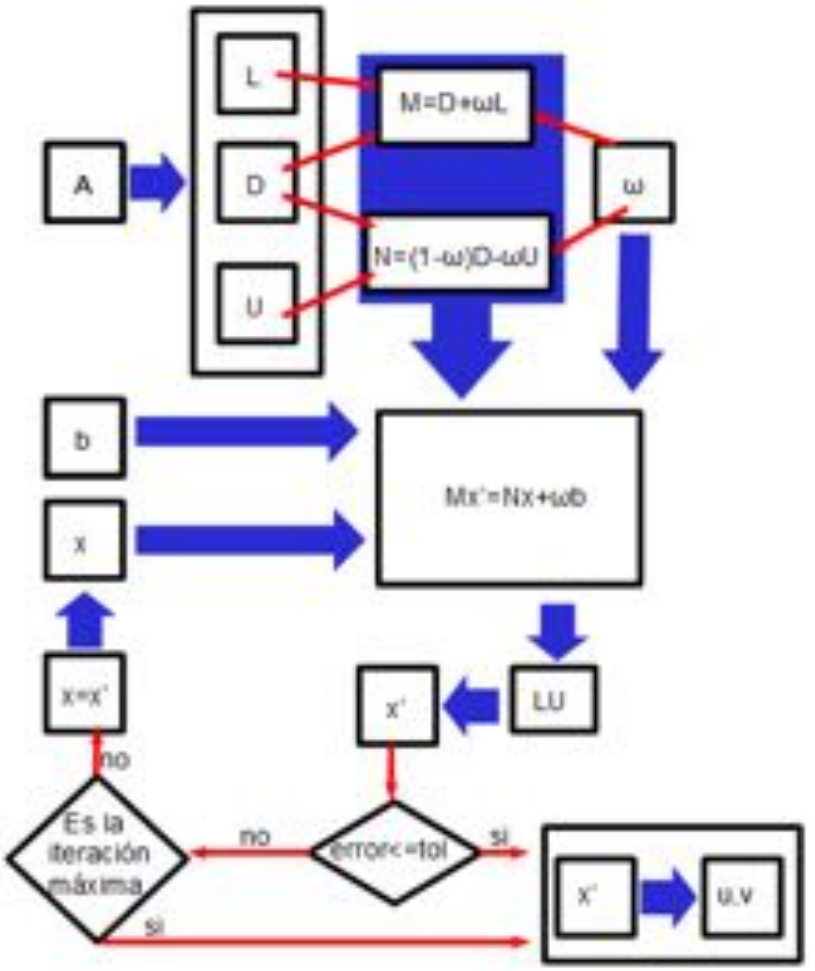


Figura 6.19. Diagrama de la implementación del método SOR.

El vector solución de la última iteración realizada se organiza en dos imágenes de ITK, una para u y otra para v . El orden que se sigue es el inverso al ocupado para generar los vectores b y x , es decir, toma los elementos intercalados de u y v , de uno en uno. En la Figura 6.19. se muestra un diagrama de la forma en que se implementa el método SOR.

6.6.5. Visualización del Flujo Óptico

El filtro *itkHermiteOpticalFlow* calcula la estimación de movimiento entre dos imágenes. Para que abarque la secuencia completa, la función *opticalFlowHermite* localizada dentro de *MainWindow* configura el cálculo entre pares de imágenes consecutivas de la secuencia de tal manera que para el primer corte se calcula con este y el segundo, para el segundo se calcula con el segundo y el tercero, así sucesivamente hasta el penúltimo ya que para que la visualización sea cíclica, la

estimación de movimiento de el último corte se realiza entre este y el primero. Cabe destacar que el orden importa y no es lo mismo calcular la estimación de movimiento entre la primera y la segunda imagen que entre la segunda y la primera, tiene que respetarse el orden de aparición.

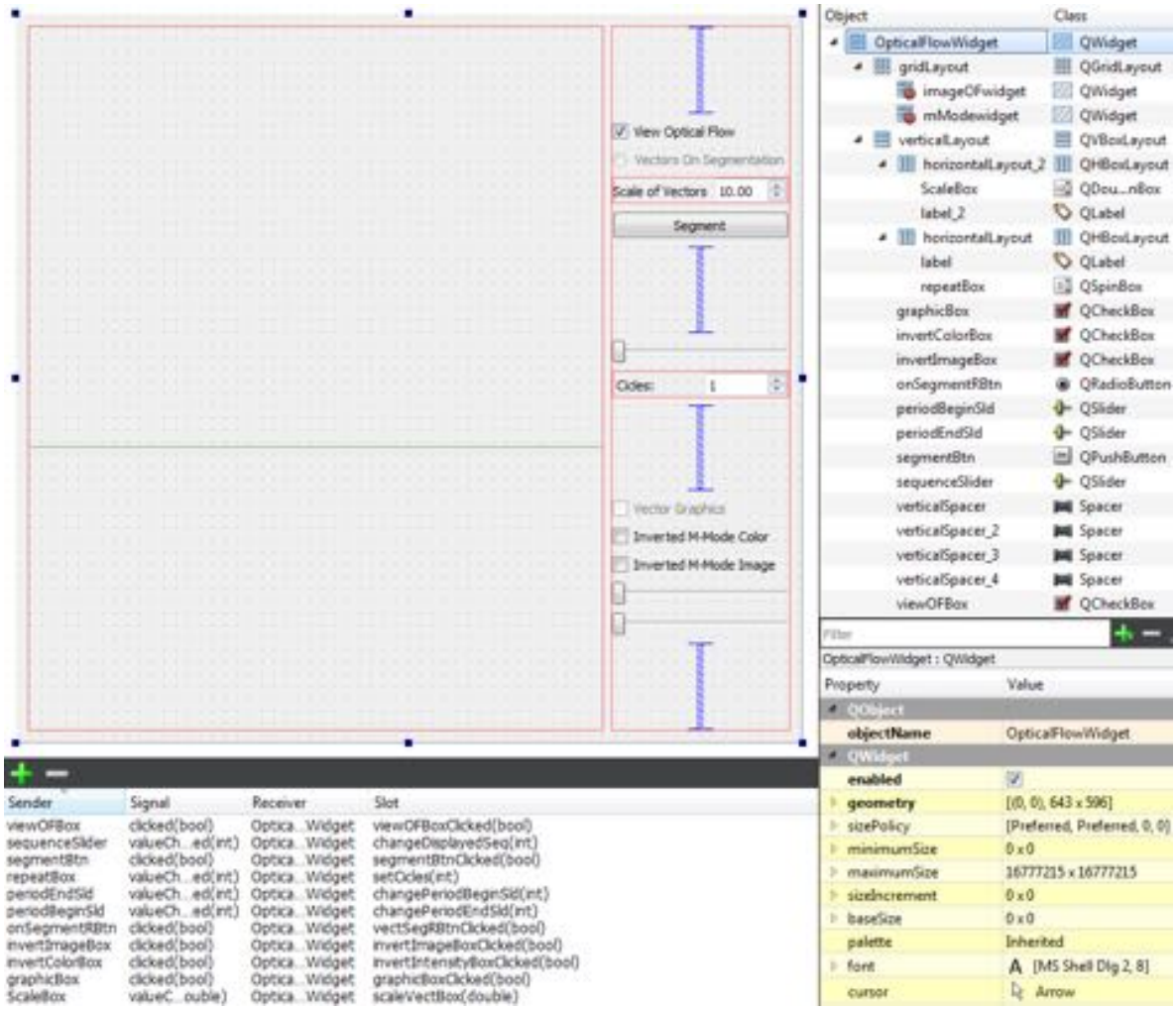


Figura 6.20. Ventana creada en Qt Creator para la visualización del Flujo Óptico.

El botón *opticalFlowBtn* se comunica con la función *opticalFlowBtnClicked de MainWindow* que al activarse se encarga de llamar a la función *opticalFlowHermite* de la misma clase para obtener el flujo óptico en toda la secuencia y luego pasárselo a un objeto de la clase *OpticalFlowWidget* para desplegar lo obtenido. A esta visualización, también se le puede entregar el modo M si se calculó previamente. La creación de la ventana *OpticalFlowWidget* requiere el mismo proceso utilizado para *MmodeWidget,* correspondiente a la

clase del modo M, pero adicionado elementos para la visualización e interacción con el flujo óptico.

La nueva ventana cuenta con las vistas *imageOFwidget* y *mModewidget*, el primero para desplegar en la parte superior de la interfaz al flujo óptico sobre un corte de la secuencia y el segundo muestra en la parte inferior al modo M y/o una gráfica de las magnitudes de los vectores que coinciden con una secuencia de contornos trazados por el usuario. Los elementos de interacción se encuentran divididos en tres secciones, la de arriba se asocia a *imageOFwidget,* la de abajo a *mModewidget* y la de en medio involucra a ambos *QVTKwidgets*. El contenido de esta ventana desarrollada se muestra en la Figura 6.20.

6.6.5.1. Representación del Flujo Óptico

La función *createAllVectors* de la clase *OpticalFlowWidget* crea el objeto *polydata* con el método *vtkPolyData* dándole los datos de la estimación de movimiento en forma de vectores y las posiciones de los píxeles a los que pertenecen en forma de punto. Los vectores están escalados con el tamaño del píxel del corte y el factor *scale* asociado al spinbox *ScaleBox* para aumentar o disminuir su tamaño en la visualización. Debido a que cada vector se relaciona con un píxel del corte, para no saturar la imagen se despliegan cada 4 píxeles, como se muestra en la Figura 6.21.

Los métodos *vtkGlyphSource2D* y *vtkGlyph2D* son empleados para que cada vector del flujo óptico contenido en *polydata* sea representado con una flecha que varia de tamaño y color según su magnitud. Dando las coordenadas del píxel, el cetro de la flecha es dibujada en su esquina inferior izquierda. Por lo que los puntos se recorren la mitad de las componentes del vector en X y Y y así el inicio de la flecha esta dicha esquina. Esto también se ejemplifica en la Figura 6.21.

Para el flujo óptico de cada corte se crea una barra de colores como referencia a sus magnitudes, asociando al azul claro a la menor, pasando por azul oscuro y violeta hasta llegar al rojo para la mayor. Las representaciones de cada flujo óptico y su barra de referencia se guardan en vectores que guardan este tipo de objetos. Así, cuando se manda a dibujar un corte en la vista, sólo hay que darles el índice que le corresponde para desplegarlos.

El checkbox *viewOFBox* se conecta con la función *viewOFBoxClicked* para agregar o quitar las flechas de flujo óptico y la barra de referencia. Mientras que el slider *sequenceSlider* está asociado con la función *changeDisplayedSeq* para navegar en la secuencia de cortes, cuando se tiene el modo M, una línea amarilla vertical indica su correspondencia con el corte visualizado.

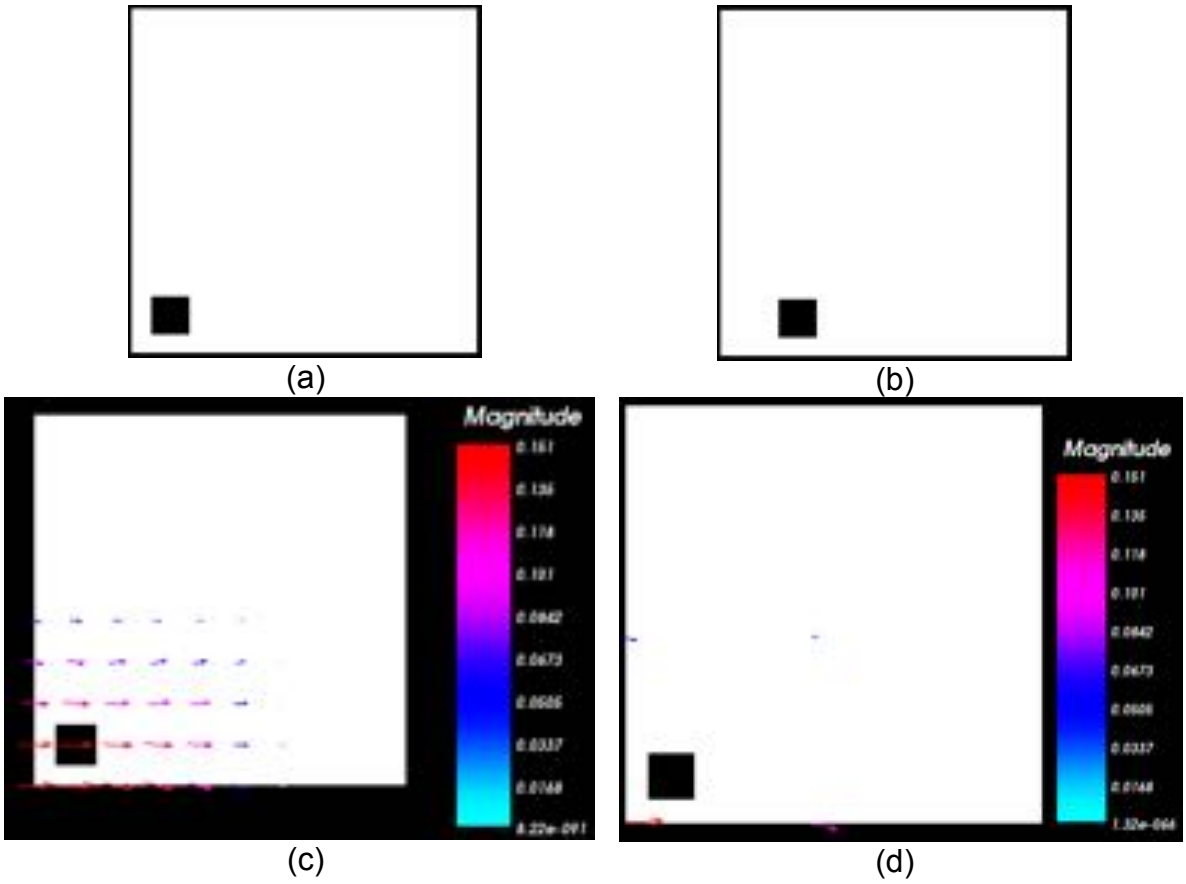


Figura 6.21. Visualización del flujo óptico aumentado 10 veces sobre una imagen de 10x10 con un píxel que se desplaza un lugar a la derecha. (a) Primera imagen. (b) Segunda imagen. (c) Flujo óptico en cada pixel sin recorrer. (d) Flujo óptico cada 4 píxeles con las flechas recorridas.

6.6.6. Secuencia de Contornos

La función *segmentBtnClicked* que se activa con el botón *segmentBtn,* es similar a *mModeBtnClicked* de la clase *MainWindow* en cuanto a que inicializa y configura al objeto *segmentStyle* de la clase *vtkTracerInteractorStyle* para realizar un trazo sin embargo, aquí adicionalmente tiene la tarea de desactivar la visualización del flujo óptico junto con sus interactores correspondientes para colocar al primer corte de la secuencia y poder iniciar ahí el trazo. Con la secuencia de contornos seleccionada, la vista regresa al corte que se tenía antes de presionar el botón y en caso de tener *viewOFBox* encendida, se muestra el flujo óptico sobre el contorno mostrado en verde.

Al realizar cualquier tipo de línea, ya sea a mano alzada o con puntos guía, la clase *vtkTracerInteractorStyle* se comunica con *OpticalFlowWidget* mediante la función *setSegmentedPath* para pasar la información del contorno hecho.

Cada corte requiere que se le haga un contorno y previendo que el realizado en la vista actual puede no satisfacer los deseos del usuario, en la función *setSegmentedPath* de *OpticalFlowWidget,* después de guardar los datos recibidos, se inicializa una ventana de la clase *OFsegWidget* sin contener ningún *QVTKwidget.* Aquí se elige si el trazo se acepta, se vuelve a realizar o se descarta cualquier trazo de cortes anteriores y se sale de la segmentación. La realización del contenido de la ventana de opción se muestra en la Figura 6.22.

Al salir de la segmentación sin guardar, se ejecuta la función *clearSegmentation* que regresa la visualización al estado en el que estaba antes de seleccionar la opción de segmentación. Para aceptar o repetir el dibujo se llama a la función *segmentedPoints* con el parámetro *true* o *false,* respectivamente, se decide rehacerlo y *segmentStyle* se reinicializa sobre el corte actual o en caso opuesto, lo guarda y se reinicia *segmentStyle* sobre el corte siguiente.

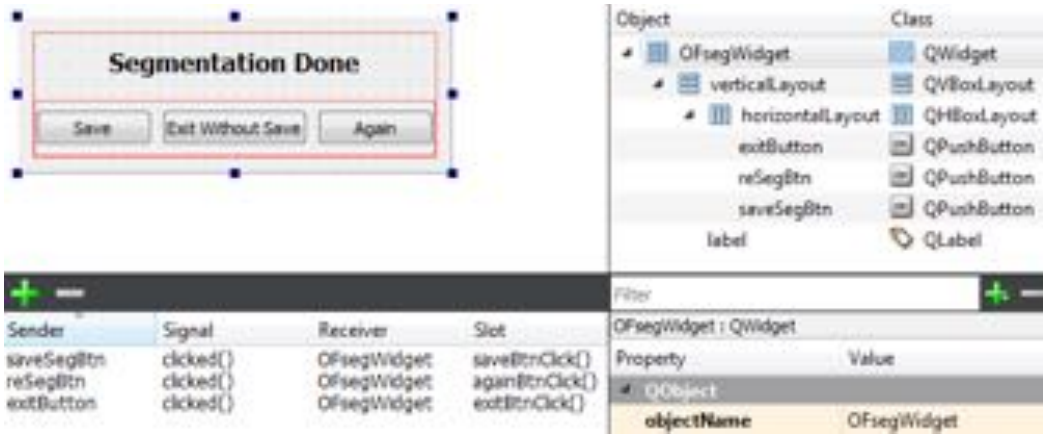


Figura 6.22. Ventana hecha con Qt Creator para indicar que proceso se aplica al contorno segmentado.

Al guardar el contorno se crea un spline de 120 puntos equidistantes, controlando la curvatura de éste con el método *vtkKochanekSpline.* Los puntos generados son operados para obtener las coordenadas correspondientes sobre la imagen y guardarlas en el vector *segPoints*. Si el corte actual es el último regresa al corte en que estaba y despliega el flujo óptico sobre el contorno correspondiente.

La representación del flujo óptico sobre el corte se genera mediante la función *createSegVectors*, que funciona igual que *createAllVectors* cuya única diferencia esta toma las coordenadas de *segPoints* para seleccionar en donde y que flechas

se crean junto con sus barras de referencia para guardarlos en *segVectorActors* y *segScalarBarActors*.

El despliegue se activa automáticamente y activa al radiobutton *onSegmentRBtn* con su función asociada *vectSegRBtnClicked* para indicar que en lugar de visualizar el flujo óptico en toda la imagen, únicamente hará en el contorno dado, siempre y cuando *viewOFBox* esté seleccionado.

6.6.6.1. Gráfica de Magnitudes

Independientemente de si la ventana de flujo óptico recibe el modo M o no, el *QVTKwidget* de la parte inferior, *mModewidget,* puede desplegar una gráfica de las magnitudes de cada vector sobre el contorno seleccionado y su promedio. Si el modo M fue incluido, la gráfica lo usa de fondo para poder relacionarlo con los valores mostrados.

Con la secuencia de contornos generada, el checkbox *graphicBox* se habilita para poder seleccionarlo y de esa forma obtener la gráfica de las magnitudes correspondientes. La función con la que se comunica es *graphicBoxClicked,* que a su vez manda a llamar a la función *fillPlot* para que con ayuda del método *vtkChartXY* despliegue la gráfica correspondiente y con el de *vtkTable* para generar una tabla con las magnitudes de cada vector. Los renglones y columnas de cada vector en la tabla se pasan a un objeto del método *vtkPlot* para comunicarse con *vtkChartXY* indicándole que genere una línea por vector sobre la gráfica desplegada. La imagen de modo M es pasada como textura a través del método *vtkBrush* y se le da una opacidad de 255 para visualizarse.

El ordenamiento los vectores dependen del sentido en que se dibuja la línea del contorno. Por lo tanto, para que los valores de cada una de las líneas en la gráfica le pertenezcan a un solo vector, todos los trazos de la secuencia deben ser hacerse en el mismo sentido.

La magnitud promedio para cada corte se calcula dentro del ciclo que asigna las magnitudes de cada vector a la tabla y la asigna como si fuera un vector extra. Para diferenciar la magnitud de los vectores de la magnitud promedio, a las primeras se los dibuja en verde y a la segunda en azul.

Capítulo 7
Pruebas y Resultados

7.1. Manipulación de volúmenes 4D

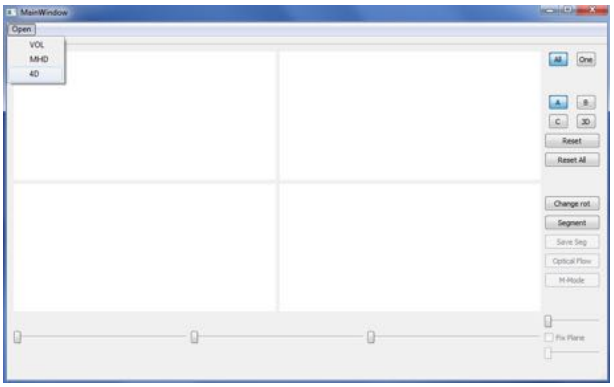
La interfaz principal consta de una pestaña en la parte superior para seleccionar el tipo de archivo que se desea abrir, cuatro *QVTKwidgets* para visualizar un volumen desde distintas vistas y distintos elementos para interactuar con el. Junto con la interfaz inicial también hay un cuadro de diálogo que devuelve información de la imagen y de la manipulación que se está realizando.

Al seleccionar la pestaña *4D* contenida en *Open*, se abre una ventana para elegir un archivo de extensión .vol que contenga una imagen 4D. Una vez seleccionado, el cuadro de diálogo y la interfaz despliega el contenido. Esto se muestra en la Figura 7.1.

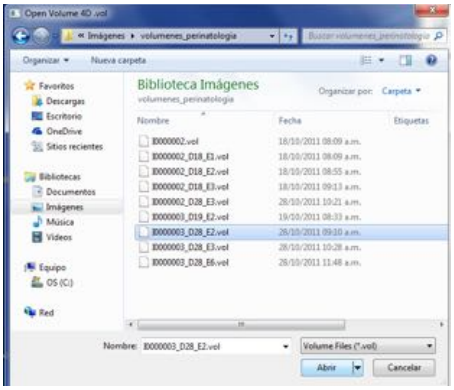
La imagen 4D se despliega como una secuencia de volúmenes mostrando un volumen por vez, como se muestra en la Figura 7.1. (d), donde se despliegan los tres planos positivas del volumen y el volumen. Con los botones *A, B, C* y *3D* se decide en qué vista se pueden realizar las siguientes acciones:

- Combinar todas las vistas en una sola para mostrar la que está seleccionada, al oprimir *One*.
- Definir un punto de rotación distinto al centro dando clic a *Change rot*.
- Rotar sobre un punto con los tres sliders de la parte inferior.
- Desplegar otro plano del volumen actual con el slider que está encima de *Fix Plane*.
- Navegar entre los volúmenes de la secuencia con el slider de la esquina inferior derecha,para este caso todas las vistas son modificadas.
- Reiniciar la vista con *Reset* y todas las vistas con *Reset All.*
- Segmentar manualmente alguno de los planos con Segment.
- Interactuar con el despliegue a través del mouse:
  - Acercar y Alejar al girar el scroll o moviendo el mouse con el botón derecho presionado.
  - Cambiar de posición al mover el mouse manteniendo el scroll oprimido.
  - Modificar el contraste moviendo el mouse con el botón izquierdo apachurrado.

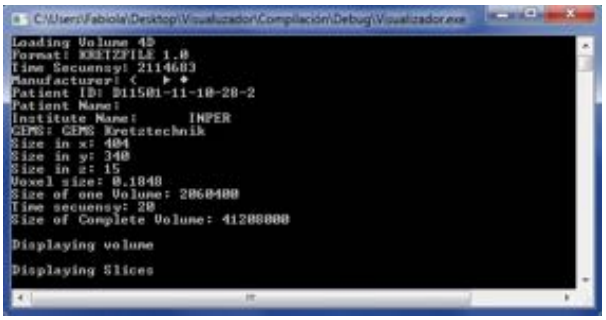
Activando *Fix Plane* con la vista superior izquierda seleccionada, genera una secuencia de cortes con la posición actual del plano mostrado en la vista. Las rotaciones y la navegación entre planos del mismo volumen se desactivan mientras que pasa lo contrario con los botones de *Optical Flow* y *M-Mode*.



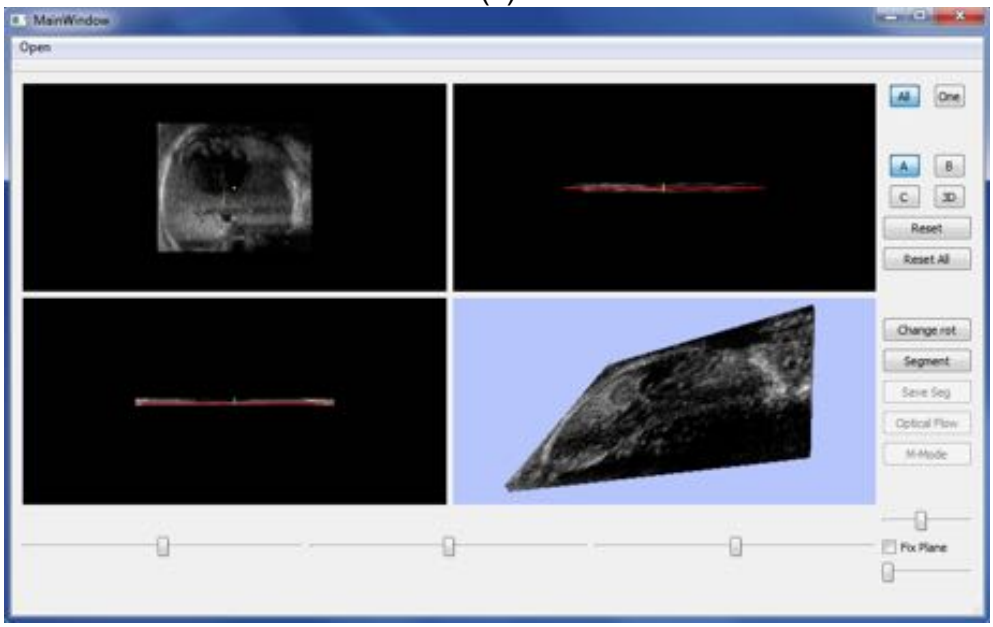
(a)



(b)



(c)



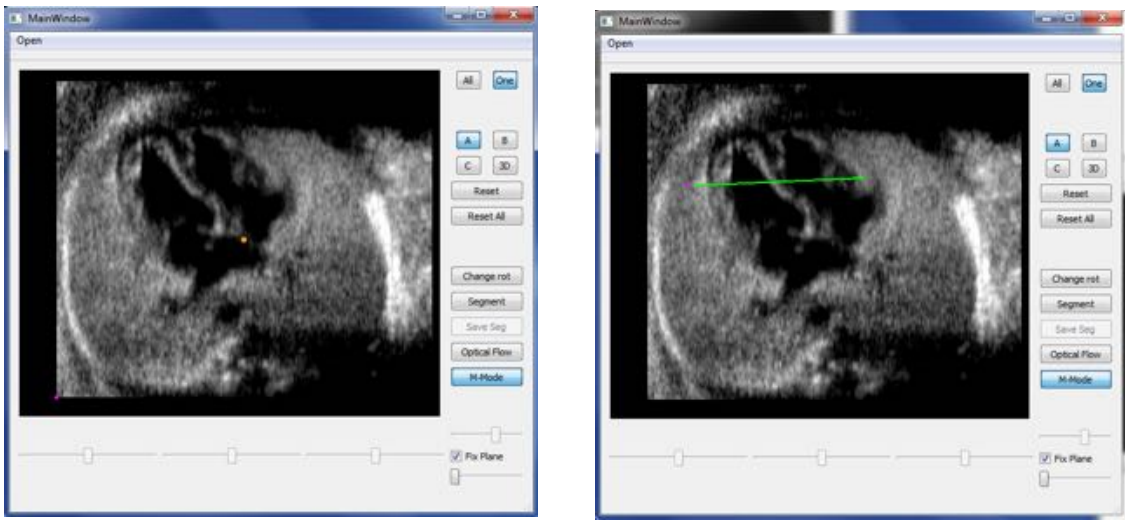
(d)

Figura 7.1. Archivo de una imagen 4D con extensión .vol en la interfaz inicial. (a) Selección la pestaña *4D*. (b)Ventana de selección de archivo .(c) Cuadro de diálogo con información de la imagen 4D contenida. (d) Despliegue de la imagen 4D en la interfaz inicial.

Figura 7.2. Despliegue de un corte seleccionado en toda la ventana de la interfaz inicial.

7.2. Modo M

Al presionar el botón *M-Mode* en la interfaz inicial, se dibuja un punto violeta en la esquina inferior izquierda de la imagen para indicar que ya se puede trazar una línea. Ya que se requiere una línea recta, esta debe de trazarse mediante la selección de dos puntos para ser aceptada. Los puntos se toman dando clic con el scroll del mouse sobre la imagen y para indicar que un punto seleccionado es el último se oprime al mismo tiempo la tecla Crtl. En la Figura 7.3. se muestra la generación de la línea para el modo M.



(a) (b)

Figura 7.3. Trazo de una línea recta para calcular el modo M. (a) Trazado habilitado. (b) Línea dibujada.

El resultados se visualiza en una nueva ventana y permite:

- Repetir el modo M hasta 10 veces al modificar *Repeat*.
- Invertir la imagen del modo M sobre el eje Y al activar *Inverted Image*.
- Invertir colores del modo M con *Inverted Colors*.
- Interactuar con la imagen a través del mouse, igual que con la interfaz inicial.
- Seleccionar el inicio y fin de de un periodo al elegir *Select Period,* dar clic sobre las posiciones deseadas y desactivar el botón.
-

Estas opciones se muestra en la Figura 7.4.

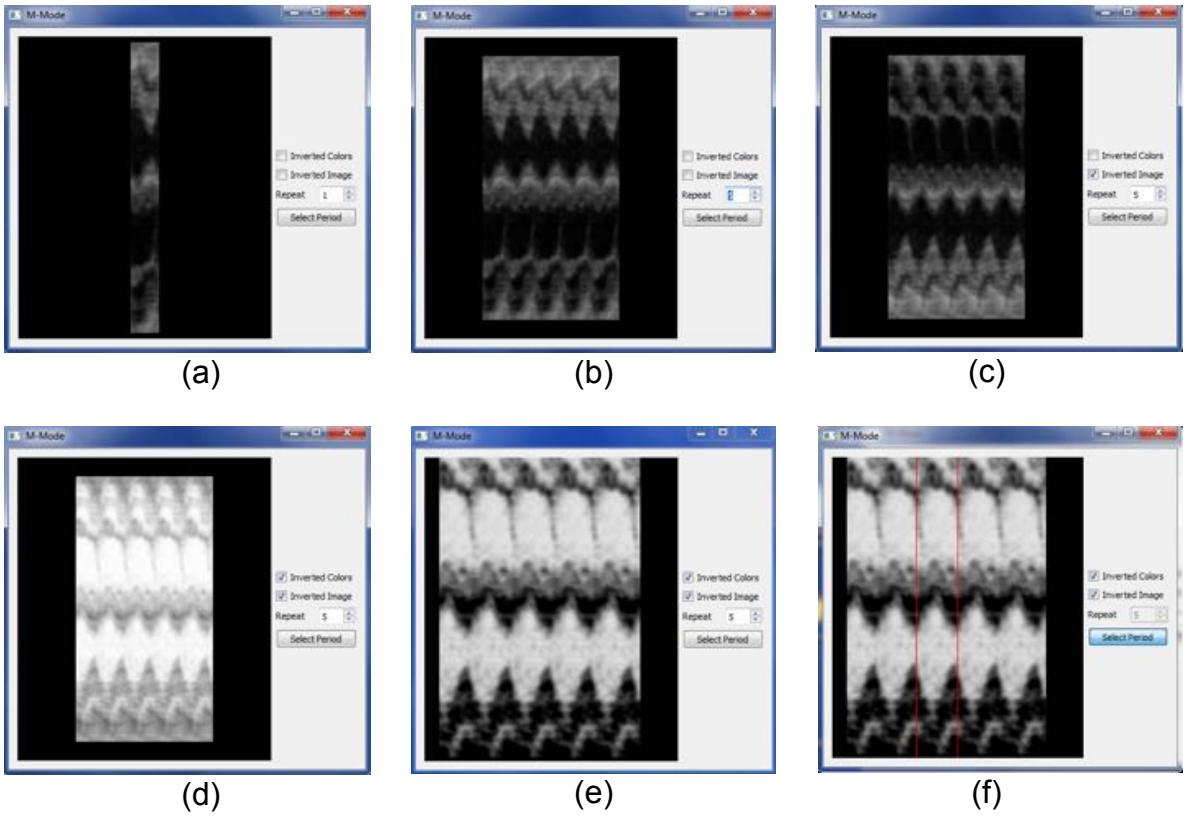


Figura 7.4. Interfaz de modo M. (a) Inicialización de la interfaz. (b) Repetición del modo M. (c) Inversión de la imagen del modo M. (d) Inversión de los colores del modo M. (e) Modificación de posición, distancia y contraste de la imagen. (f) Selección del inicio y fin de un periodo sobre el modo M.

7.3. Flujo Óptico

7.3.1. Comparación de Resultados

La medida de desempeño más comúnmente usada en flujo óptico es el error angular (AE) [45]. El ángulo en el espacio entre un vector de flujo (u , v) y uno de ground truth (u_{GT} , v_{GT}) es el AE y puede calcularse obteniendo el producto punto de los vectores anteriormente mencionados, dividiéndolo entre el producto de sus longitudes y luego aplicarle el coseno inverso.

El código generado por Vargas-Quintero [2] es para Matlab y tomando en cuenta que éste software y el conjunto de librerías ITK y VTK no tienen sus métodos implementados de forma idéntica, es más conveniente comparar las implementaciones descritas en esta Tesis en Matlab.

Se calcula la estimación de movimiento con la versión de Vargas-Quintero [2] y con la de ésta Tesis en Matlab, sobre los pares de imágenes mostradas en las Figuras 7.5. y 7.6. de tamaño 388 x 584, para comparar sus ground truth con los resultados obtenidos con con los siguientes parámetros:

- Constante de escalamiento: 0.95
- Variable para la función Ψ : ε = 0.01
- Variable para la restricción de suavizado: α = 50
- Peso de restricción para la intensidad y gradiente constantes: γ = 90
- Parámetro de homogeneidad: β = 1
- Número máximo de iteraciones en el método SOR: 20
- Número de escalas: 20
- Número de iteraciones en cada escala: 15

Los ground truth se genera a partir de la referencia de la Figura 7.7.

(a) (b) (c)

Figura 7.5. Flujo óptico de la secuencia Dimetrodon. (a) Imagen 10. (b) Imagen 11. (c) Ground truth [45].

(a) (b) (c)

Figura 7.6. Flujo óptico de la secuencia Hydragea. (a) Imagen 10 .(b) Imagen 11 . (c) Ground truth [45].

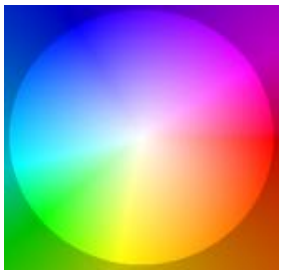
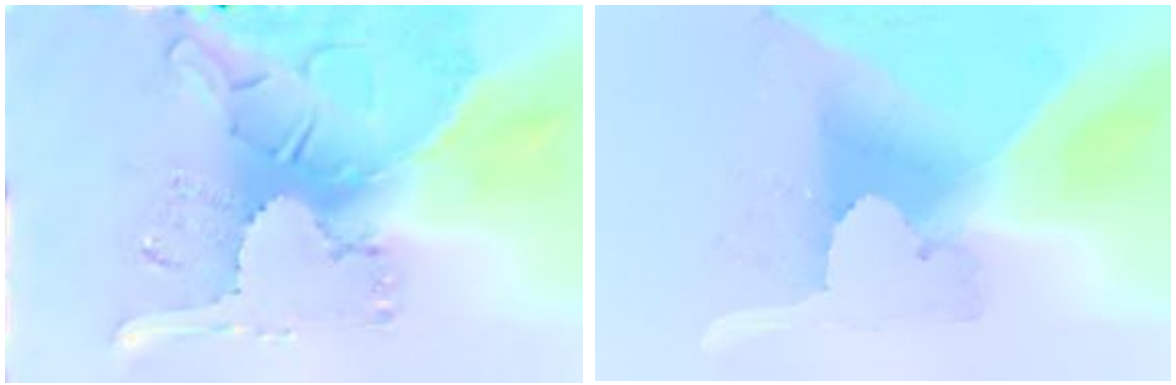


Figura 7.7. Referencia de coordenadas con color [45].

Los resultados y errores de ambas implementaciones se muestran las Figuras 7.8. y 7.9. Los errores comparan los resultados contra los ground truth de las Figuras 7.5. (c) y 7.6. (c) y entre ambos resultados.

En la Figura 7.8., el resultado obtenido de la implementación de Vargas-Quintero [2] se destacan más los contornos. A excepción de esto, en ambas secuencias los errores de los resultados contra el ground truth se ven muy similares, concordando con lo encontrado entre el error entre ambas ya que éste parece el menor de los tres.

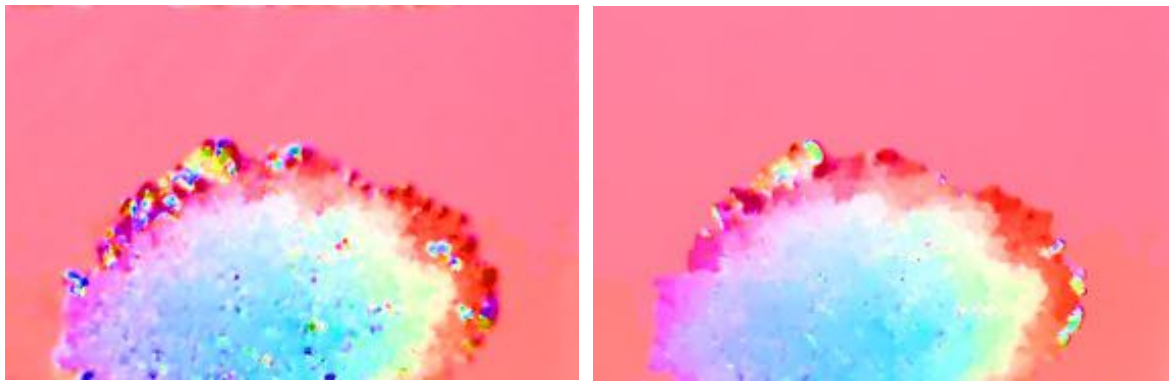


(a) (b)

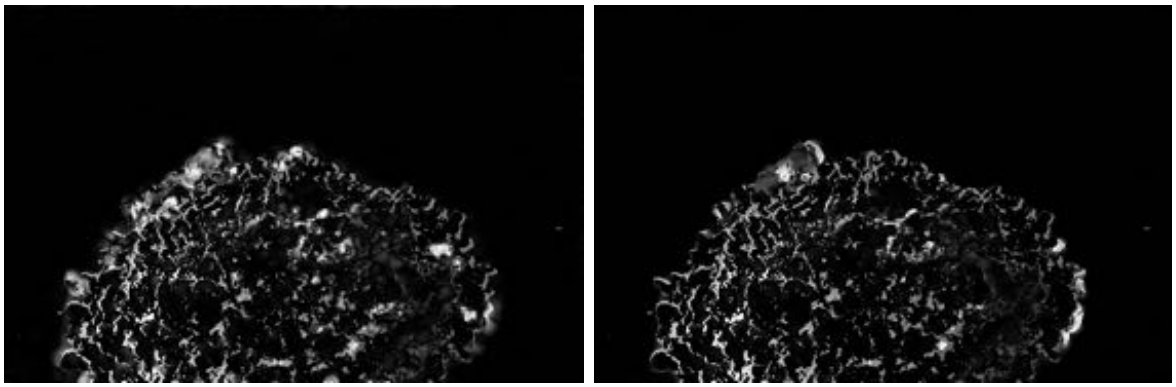
(c) (d)

(e)

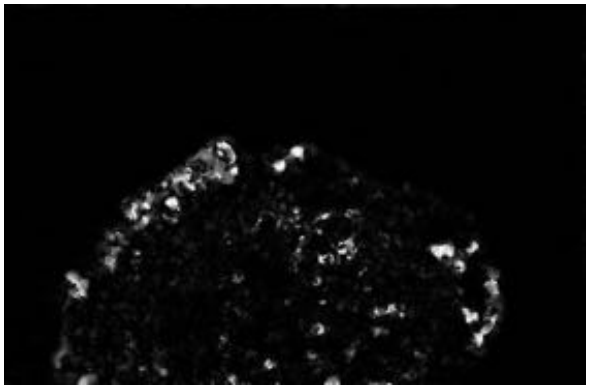
Figura 7.8. Resultados de la estimación de movimiento entre el par de imágenes mostrado en 7.5. (a) Vargas-Quintero. (b) Implementación propuesta. (c) AE de Vargas-Quintero respecto al Ground truth. (d) AE de la implementación propuesta respecto al Ground truth. (e) AE de la implementación propuesta respecto a Vargas-Quintero.



(a) (b)



(c) (d)



(e)

Figura 7.9. Resultados de la estimación de movimiento entre el par de imágenes mostrado en 7.6. (a) Vargas-Quintero. (b) Implementación propuesta. (c) AE de Vargas-Quintero respecto al Ground truth. (d) AE de la implementación propuesta respecto al Ground truth. (e) AE de la implementación propuesta respecto a Vargas-Quintero.

Los errores angulares promedio y tiempo de ejecución en las secuencias son:

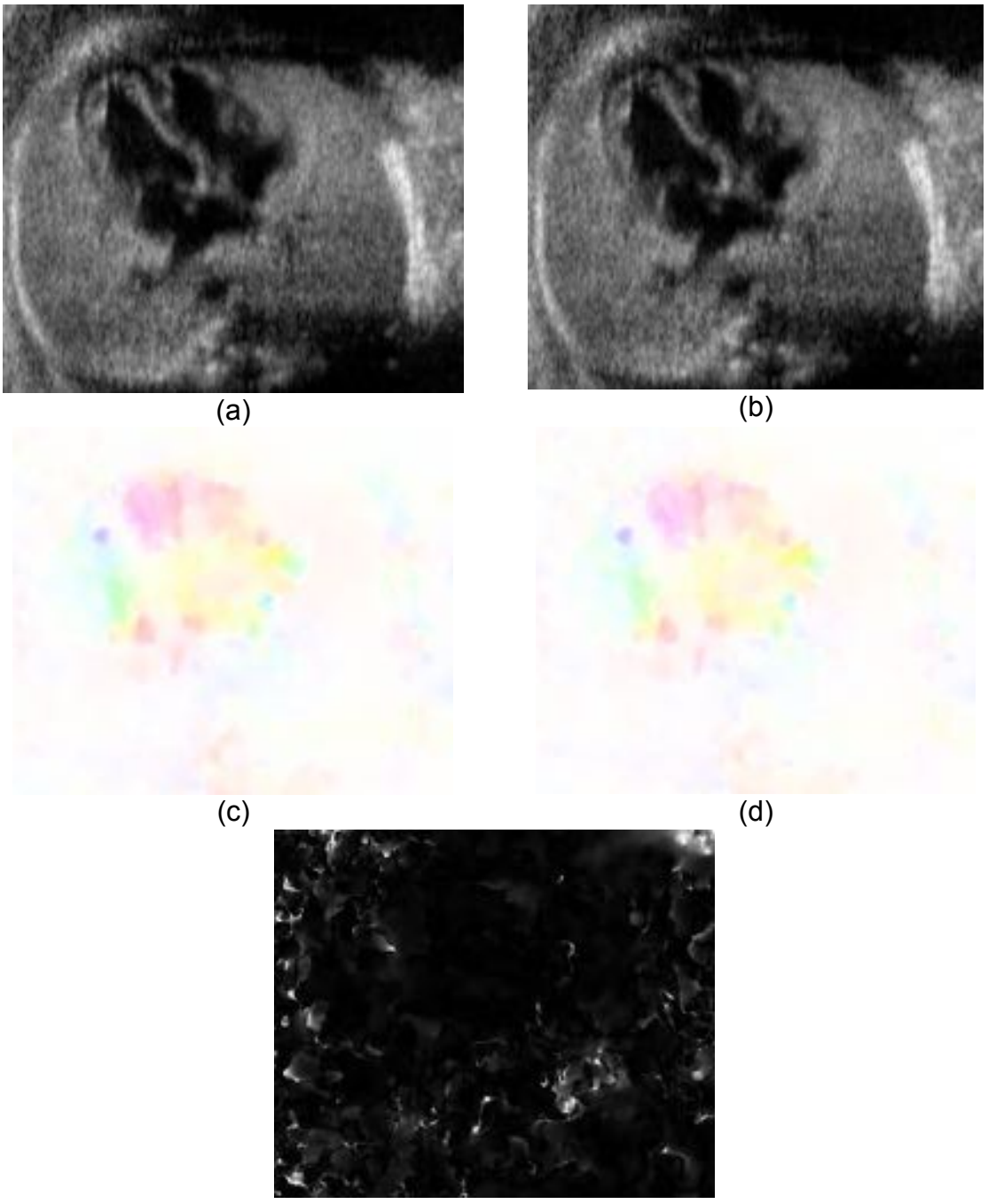
Secuencia	Dimetrodon	Hydragea
AE promedio de Vargas-Quintero respecto al Ground truth [°]	3.8964	9.9398
AE promedio de la implementación propuesta respecto al Ground truth [°]	2.587	7.8908
AE promedio de la implementación propuesta respecto a Vargas-Quintero [°]	3.0453	4.7544
Tiempo de ejecución de Vargas-Quintero [s]	377.0892	416.9077
Tiempo de ejecución de la implementación propuesta [s]	314.5913	359.0661

De manera que la implementación propuesta en ésta tesis se parece más al resultado esperado y se logra en menos tiempo.

Para imágenes de ultrasonido se modificaron los parámetros usados para que fuesen:

- Constante de escalamiento: 0.95
- Variable para la función Ψ : ε = 0.01
- Variable para la restricción de suavizado: α = 10
- Peso de restricción para la intensidad y gradiente constantes: γ = 1
- Parámetro de homogeneidad: β = 1
- Número máximo de iteraciones en el método SOR: 30
- Número de escalas: 3
- Número de iteraciones en cada escala: 3

y se aplicaron sobre un par imágenes consecutivas de ultrasonido. En las Figuras 7.10. (a) y (b) se muestra el par de imágenes utilizado, él cual es tomado de una secuencia de cortes de un volumen ecocardiográfico fetal. Ya que éstas imágenes no poseen un ground truth sólo se obtiene el error entre los resultados, los resultados y el error pueden observarse en las Figuras 7.10. (c), (d) y (e)



(a) (b) (c) (d) (e)

Figura 7.10. Resultados de la estimación de movimiento entre un par de imágenes consecutivas de una secuencia de cortes de un volumen ecocardiográfico fetal. (a) Primera imagen de la secuencia. (b) Segunda imagen de la secuencia. (c) Vargas-Quintero. (d) Implementación propuesta. (e) AE de la implementación propuesta respecto a el ground truth.

La diferencia visual entre las Figuras 7.10. (a) y (b) es la intensidad de color en la zona correspondiente a la estructura en movimiento sin embargo la Figura 7.10. (c) muestra que fuera de la estructura es donde hay más discrepancia. Para éste caso, el error angular promedio es de 9.7026[°] y los tiempos que tomó calcular los resultados son 19.7945[s] para Vargas-Quintero [2] y 17.2029[s] para la implementación propuesta.

Otra manera de comparar visualmente ambos resultados se muestra en la Figura 7.11., donde ambas estimaciones se muestran con flechas superpuestas, siendo las flechas azules las que corresponden al resultado con la implementación de Vargas-Quintero y las rojas para la implementación de ésta Tesis.

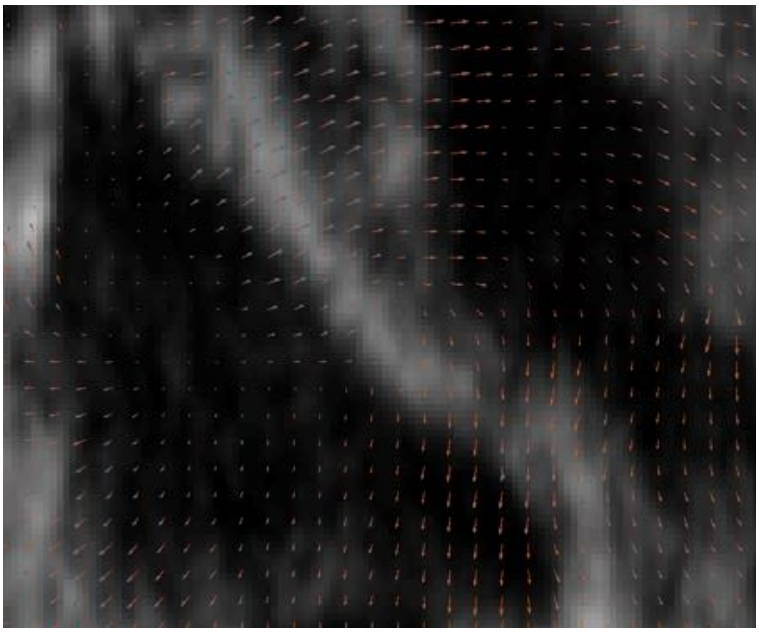


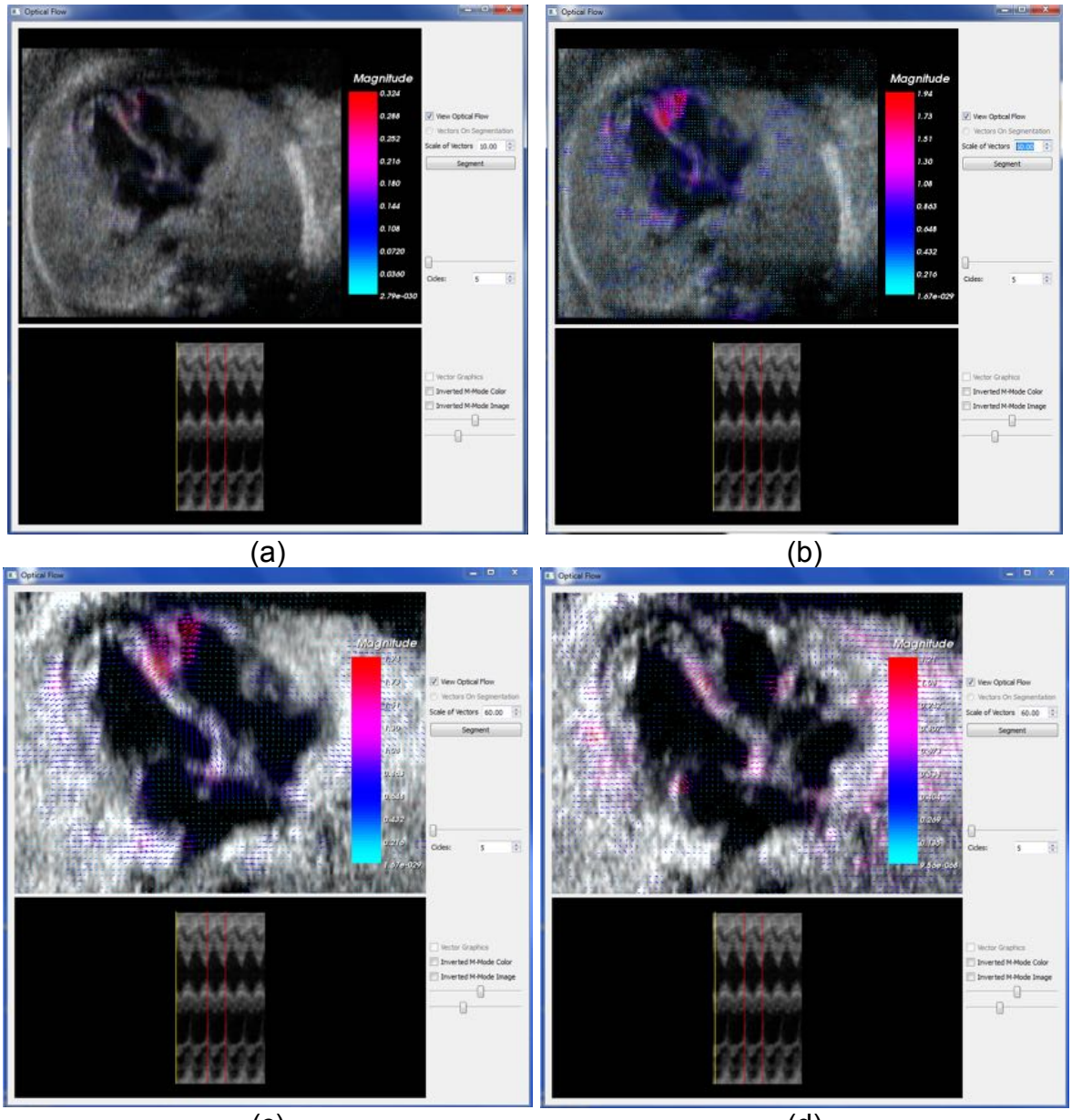
Figura 7.11. Resultados de la figura 7.10. (a) y (b) en forma de flechas, azul para Vargas-Quintero y rojo para la implementación propuesta, sobre parte de la imagen 7.9.(a).

La estimación de movimiento utilizada consiste en resolver un funcional que trabaja con los coeficientes de Hermite. Siendo que el funcional que se implementa es el que propone Vargas-Quintero en [2], la principal diferencia entre ambas implementaciones es el método de obtención de los coeficientes de Hermite.

Por un lado Vargas-Quintero [2] utiliza filtros de Hermite en dos dimensiones para obtener los coeficientes de Hermite y por el otro, en la implementación de esta tesis se ocupa el algoritmo de la Transformada rápida de orden múltiple. Mientras que para generar los filtros se pueden variar los parámetros de ventana la

Gaussiana, el algoritmo de la Transformada Rápida usa únicamente las secuencias [1,1] y [1,-1] por lo que a pesar de tener resultados similares para la estimación de movimiento, no son exactamente iguales..

7.3.2. Flujo Óptico en la Interfaz



(a) (b)

(c) (d)

Figura 7.12. Interfaz del flujo óptico. (a) Despliegue inicial. (b) Vectores magnificados 60 veces. (c) Manipulación del corte. (d) Avance de dos cortes respecto a lo mostrado en (c).

Oprimiendo el botón *Optical Flow* de la interfaz inicial, se estima el flujo óptico de la secuencia de imágenes y se despliega en otra ventana sobre su corte correspondiente. Debajo, se dibuja el modo M, este último mantiene las funciones para invertir la imagen y los colores, mientras que ocupa dos sliders para para manipular el inicio y fin del periodo seleccionado, estos se muestran como líneas rojas.

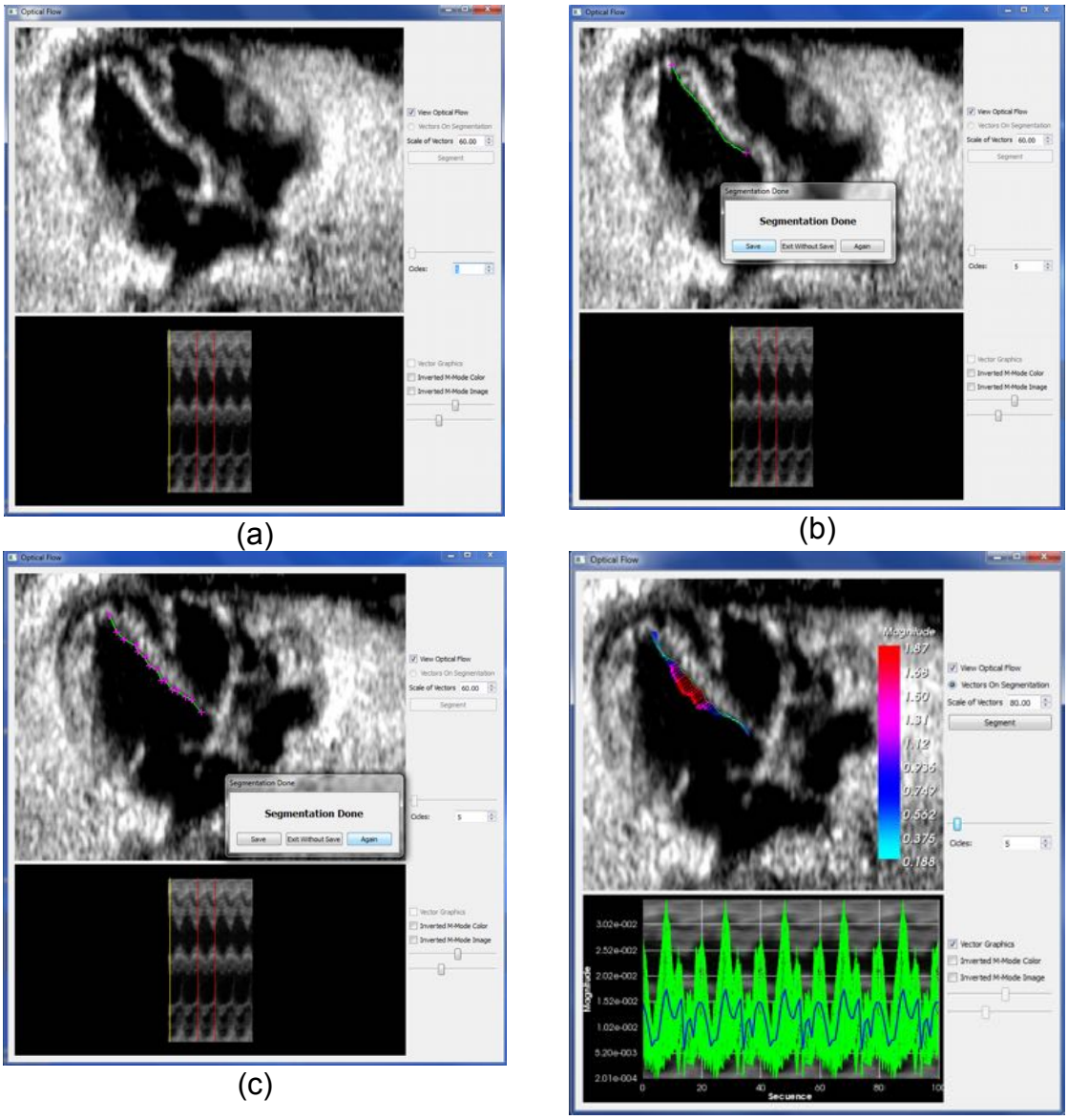
Al aumentar el modo M mediante *Cicles*, la secuencia se puede repetir tantas veces como el modo M y se muestra el corte correspondiente a la posición del slider de la parte superior, en el modo M la posición es mostrada con una línea amarilla. El número mínimo de ciclos es uno aunque puede ser mayor si el fin del periodo seleccionado en la ventana de modo M lo excede, y el máximo es 10.

El corte mostrado puede manipularse con el mouse de la misma manera que con la interfaz inicial mientras que el flujo óptico de la imagen puede ocultarse al desactivar *View Optical Flow*. Las flechas que lo representan tienen la capacidad de escalar su tamaño con *Scale of Vectors*. En la Figura 7.12. se muestra la interfaz inicial del flujo óptico con las características anteriormente descritas.

Si se quiere analizar algún contorno se habilita el trazado de una línea con *Segment* y se procede a dibujarla con el mouse a mano alzada o con puntos de control, en ambos casos el delineado debe seguir el mismo sentido para que la gráfica sea coherente con la selección. Inmediatamente después de presionar el botón, la secuencia se posiciona sobre su primer corte y el flujo óptico se oculta.

Cuando el usuario termina la línea aparece una ventana para decidir que hacer con el trazo. Si se guarda, automáticamente se muestra el corte siguiente para repetir el procedimiento, si no se guarda, se borra lo dibujado para poder volver a realizarlo sobre el corte actual, o si se opta por salir de la segmentación sin guardar, los contornos generados se eliminan y regresa a la visualización anterior al procedimiento.

Al guardar el contorno de la última imagen de la secuencia, termina la segmentación y se habilita y activa el radiobutton *Vectors On Segmentation* para mostrar el flujo óptico sobre el contorno en lugar de sobre la imagen entera. El checkbox *Vector Graphics* también se habilita para poder generar una gráfica de las magnitudes de los vectores sobre el contorno a lo largo de la secuencia mostrando en verde las líneas correspondientes a las magnitudes de los vectores y en azul su promedio. En la Figura 7.13. se ejemplifica una segmentación y su gráfica correspondiente en la interfaz del flujo óptico



(a) (b)

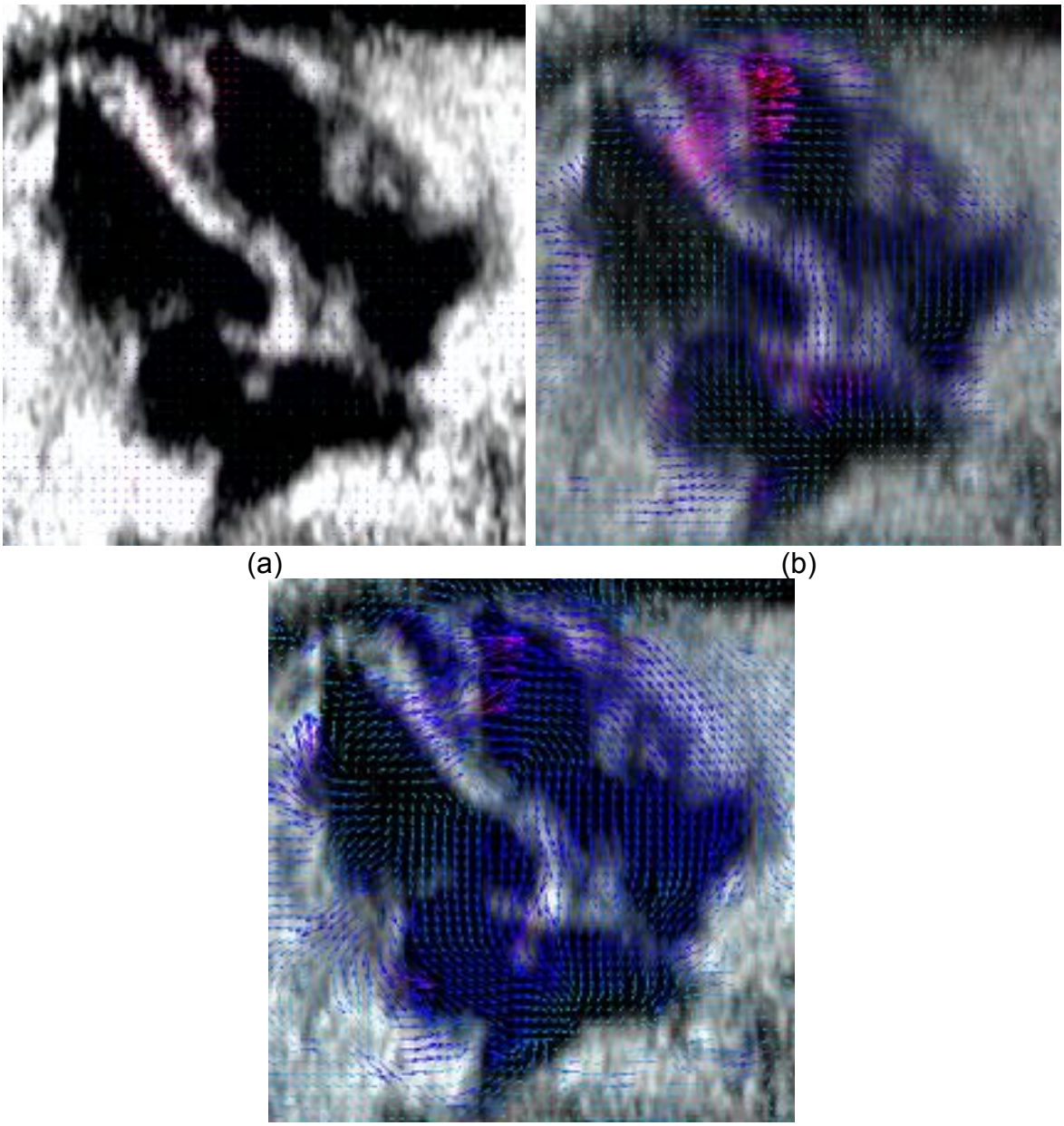
(c) (d)

Figura 7.13. Secuencia de contornos y su gráfica en la interfaz de flujo óptico. (a) Activación de trazado. (b) Línea hecha a mano alzada. (c) Línea hecha con puntos de control. (d) Flujo óptico sobre un contorno y gráfica de magnitudes del flujo óptico sobre la secuencia de contornos.

La gráfica tiene de fondo el modo M cuando este existe y mantiene las funciones para invertir los colores y la imagen pero no de modificar el periodo ni mostrar las posiciones de los sliders. Cuando el modo M no se realiza, el fondo se muestra en negro. Independientemente de que exista o no el modo M, la gráfica es afectada por el número de ciclos. Las magnitudes no dependen del aumento o disminución

del tamaño de las flechas dado por *Scale of vectors* pero si del tamaño del píxel, el cual se lleva acarreando desde el inicio de la visualización.

7.3.3. Variaciones del Número de Escalas e Iteraciones



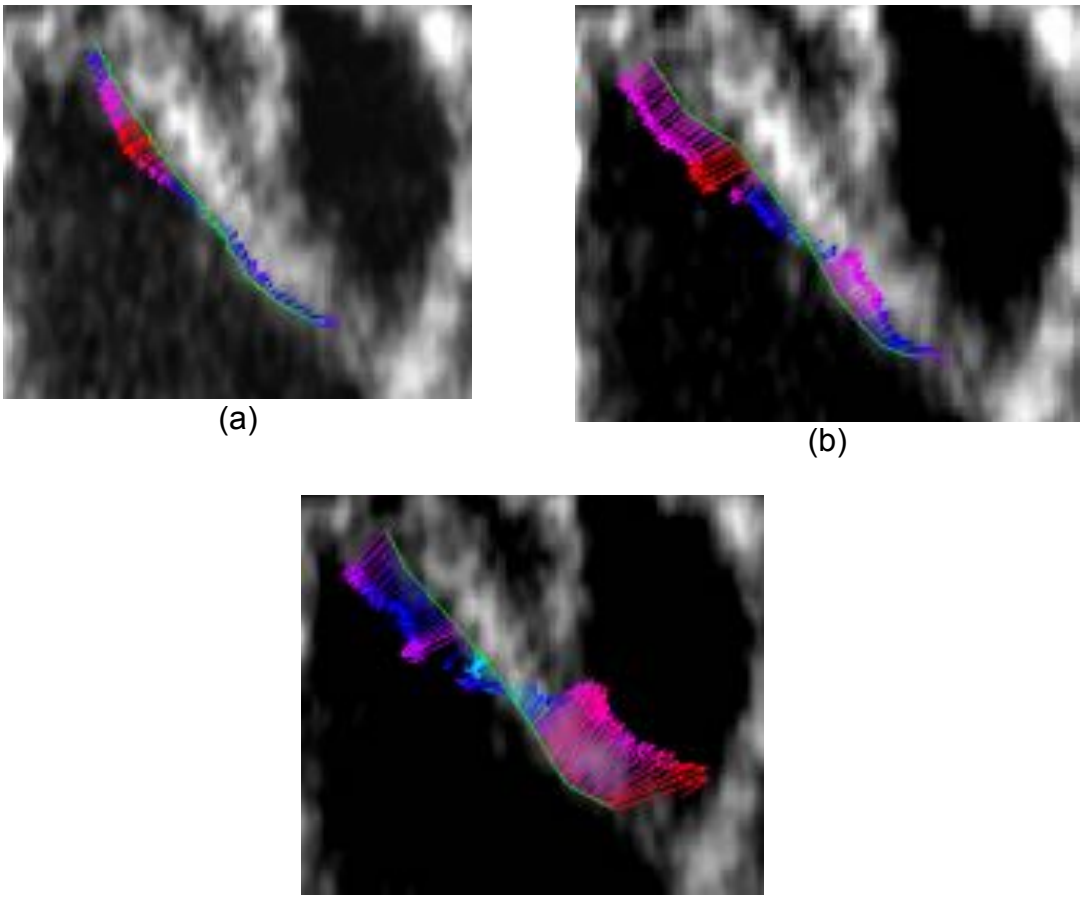
(a) (b)

(c)

Figura 7.14. Flujo óptico del primer corte de una secuencia utilizando una iteración por escala y aumentando la magnitud de los vectores 10 veces. (a) Con una escala. (b) Con cuatro escalas. (c) Con 10 escalas.

Dentro del código principal están definidos el número de escalas y el de sus iteraciones correspondientes. El uso de un número de interacciones y/o escalas mayor a uno mejora la aproximación del flujo óptico a costa de realizar una mayor cantidad de procesos, lo que conlleva utilizar más tiempo.

En la Figura 7.14. se observa que al aumentar el número de escalas, el tamaño de los vectores crece, las magnitudes son más similares entre sí dentro del mismo corte ya que el color es más uniforme, y las direcciones que muestran las flechas se vuelven menos rígidas pero disminuye la cantidad de zonas con poco o ningún flujo, las cuales son resultado de la restricción de regiones homogéneas.



(a) (b) (c)

Figura 7.15. Flujo óptico de un contorno utilizando una iteración por escala (a) Con una escala y aumentando la magnitud 80 veces. (b) Con cuatro escalas y aumentando la magnitud 20 veces. (c) Con 10 escalas y aumentando la magnitud 20 veces.

En la Figura 7.15., al aumentar la cantidad de escalas, las direcciones de las flechas varían más entre sí y las magnitudes cambian más abruptamente. Las gráficas con las que se relacionan lo reflejan en la Figura 7.16. al aumentar el número de picos y valles.

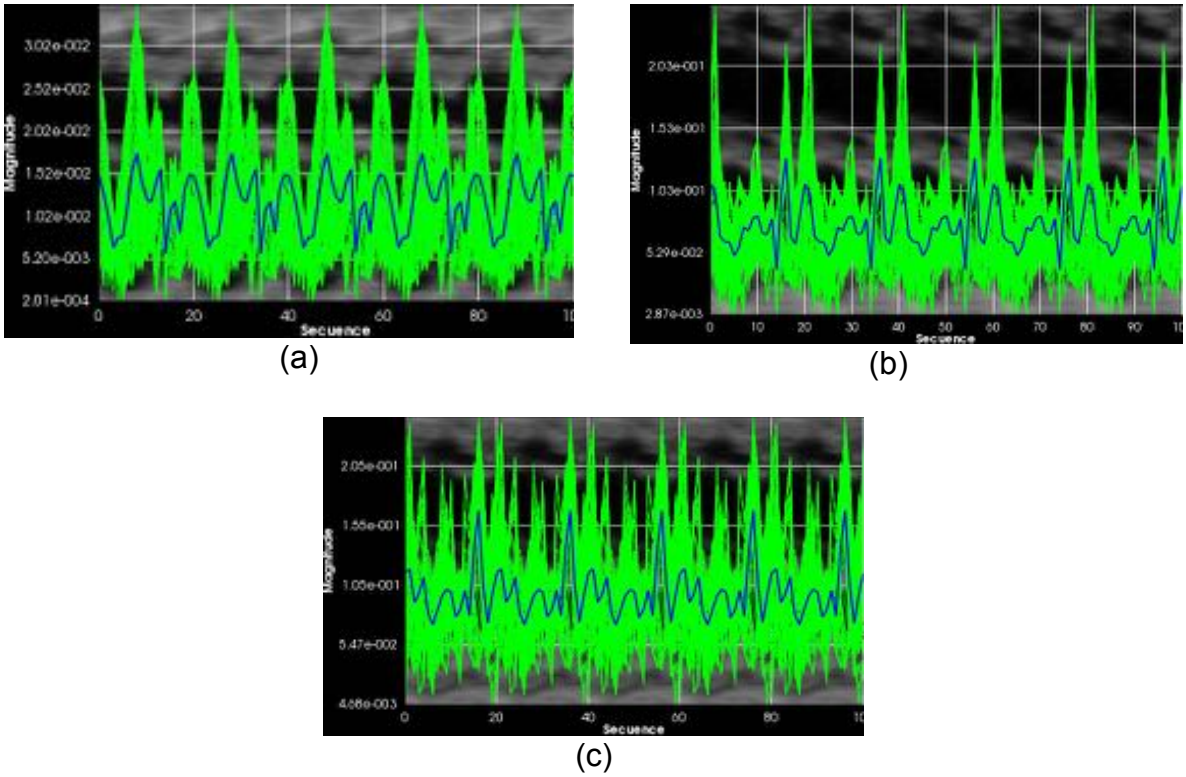
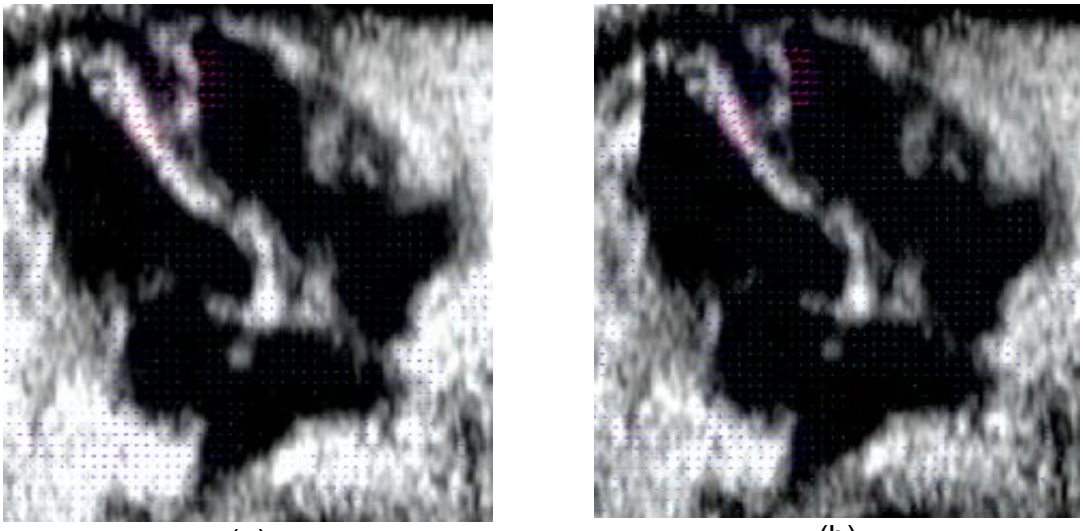


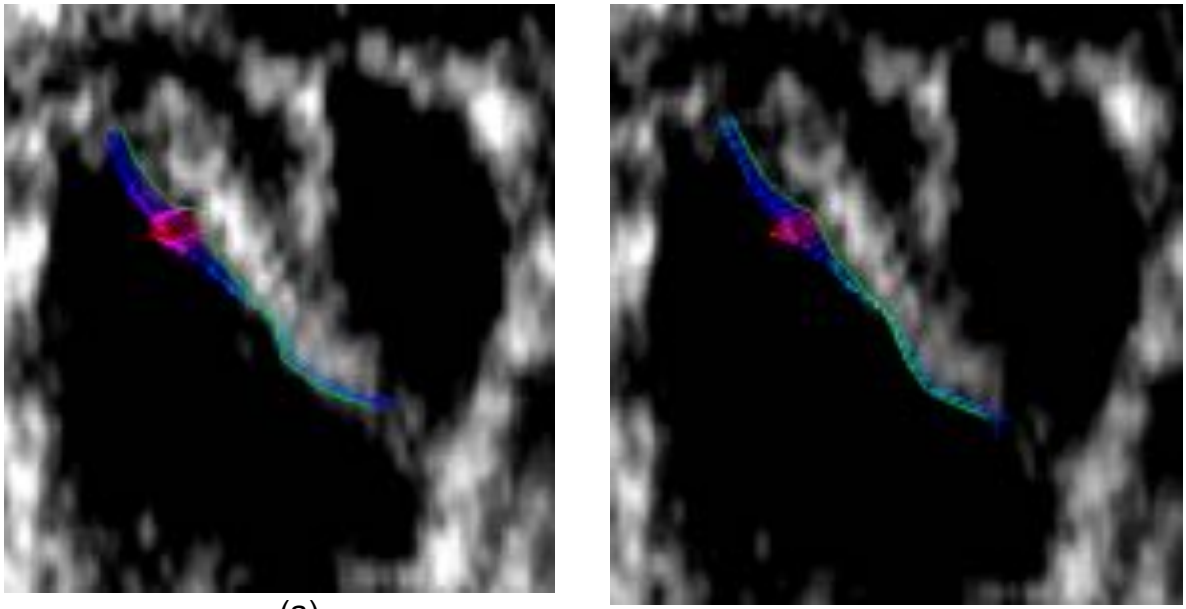
Figura 7.16. Gráfica de las magnitudes de flujo óptico sobre una secuencia de contornos utilizando una iteración por escala y repitiendo cinco veces la secuencia (a) Con una escala. (b) Con cuatro escalas. (c) Con 10 escalas.

Al igual que en el aumento en escalas, cuando se hacen más iteraciones dentro de la misma escala de imagen, las flechas también crecen aunque no tan drásticamente y el resultado es muy similar, como se observa en la Figura 7.17. También se obtiene una mejor definición de las zonas que presentan menor o ningún flujo.



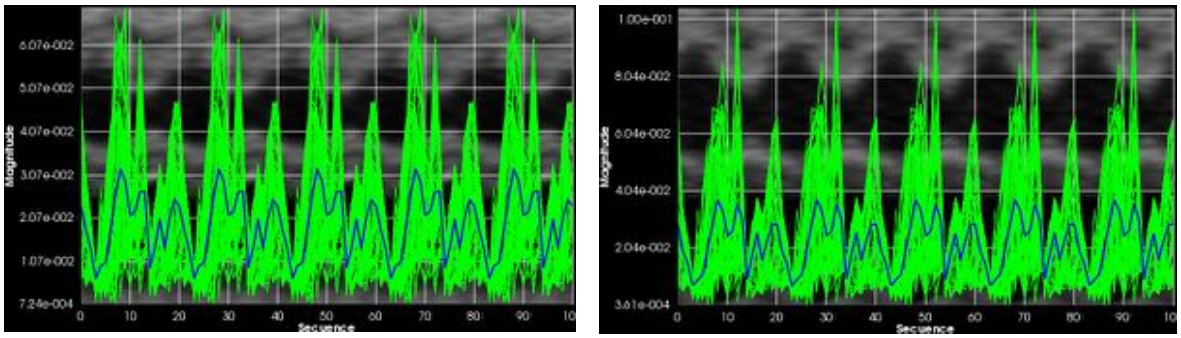
(a) (b)

Figura 7.17. Flujo óptico sobre el primer corte de una secuencia utilizando 1 escala y aumentando la magnitud de los vectores 10 veces. (a) Con 4 iteraciones. (b) Con 10 iteraciones.



(a) (b)

Figura 7.18. Flujo óptico sobre un contorno utilizando una escala. (a) Con cuatro iteraciones y aumentando la magnitud 80 veces. (b) Con 10 iteraciones y aumentando la magnitud 60 veces.



(a) (b)

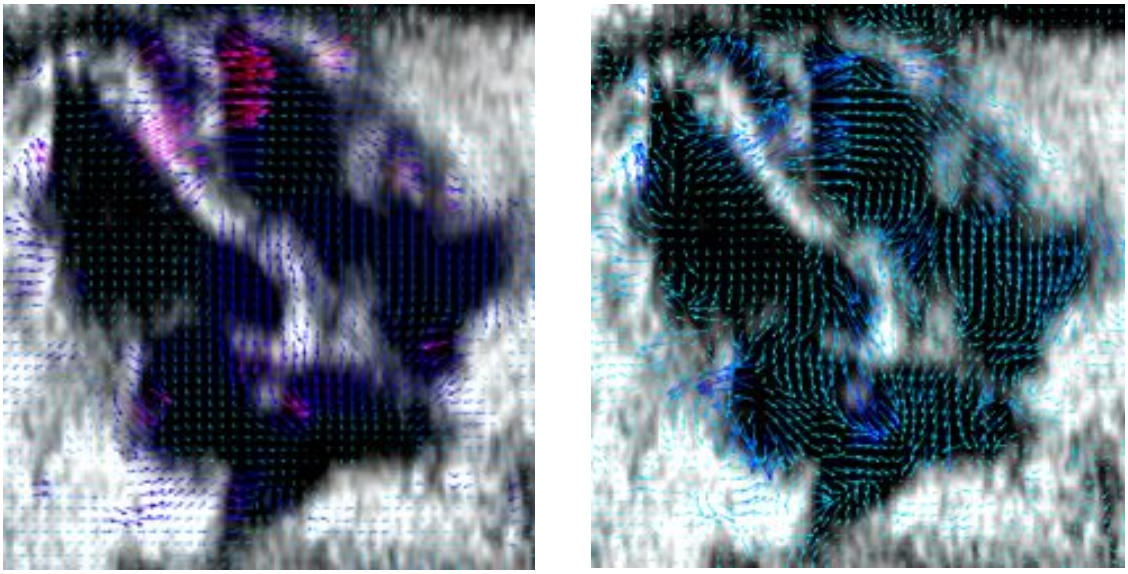
Figura 7.19. Gráfica de las magnitudes de flujo óptico sobre una secuencia de contornos utilizando una escala y repitiendo cinco veces la secuencia. (a) Con cuatro iteraciones. (b) Con 10 iteraciones.

Las iteraciones ayudan a estabilizar la estimación de movimiento dentro de la escala por lo que las direcciones en las que apuntan los vectores sobre el contorno en la Figura 7.18. casi no cambian pero si la relación entre sus magnitudes. Razón por la cual, las formas de las gráficas resultantes de la Figura 7.12 son muy similares entre sí y con la de 7.16.(a), que también ocupa una sola escala.

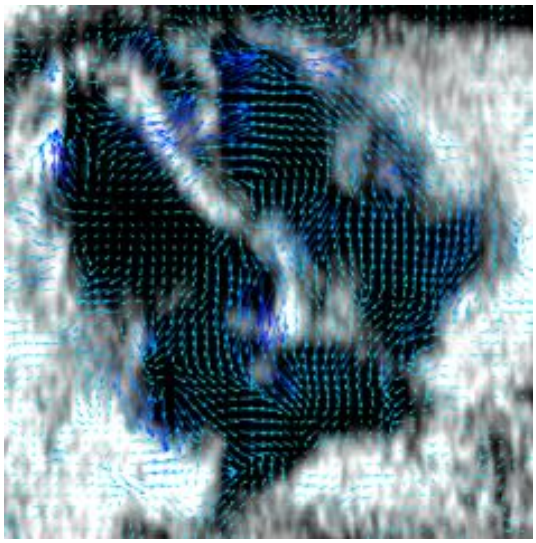
Un flujo fluido pero estable que preserve las zonas con poco o nada de flujo, es el objetivo de combinar el aumento de escalas con el de iteraciones. La Figura 7.20. muestra estas características en mayor o menor medida. A diferencia del aumento exclusivo de la cantidad escalas, entre los incisos de esta Figura, las flechas difieren poco en tamaño, aunque conservan la característica de homogeneizar las magnitudes de todo el corte, disminuyendo las zonas con poco o ningún flujo.

Mientras que en el flujo óptico de todo el corte se distinguen mejor las características asociadas al aumento de escala, sobre el contorno se destacan las asociadas al uso de más iteraciones ya que las flechas de las diferentes imágenes de la Figura 7.14 forman tendencias muy similares entre sí mediante su dirección y magnitud.

Las gráficas de la Figura 7.22. se parecen más a la de la Figura 7.16.(c), correspondientes al aumento en el número de escalas, que a las de la Figura 7.19. las cuales sólo modifican sus iteraciones. Sin embargo, las gráficas de 7.22. presentan entre si las mismas tendencias, esto es coherente con las características vistas en los contornos. Un pico se destaca entre todos los demás cada vez que se aumenta la cantidad de escalas y los valles tienden a alinearse con el aumento de iteraciones.

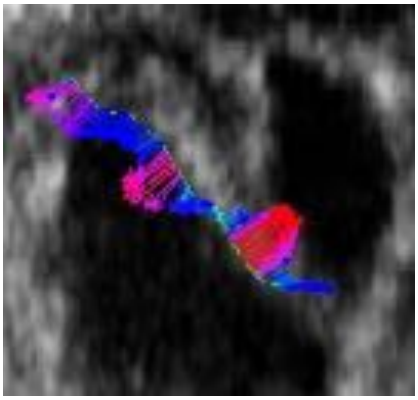


(a) (b)

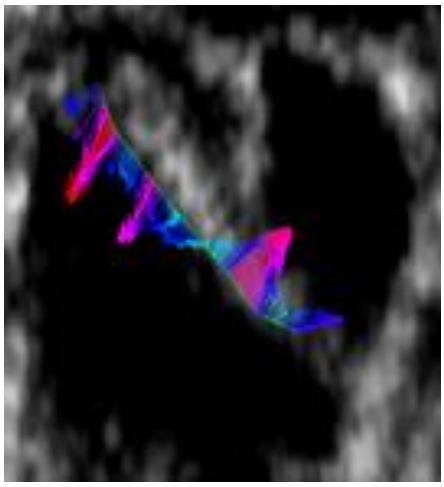


(c)

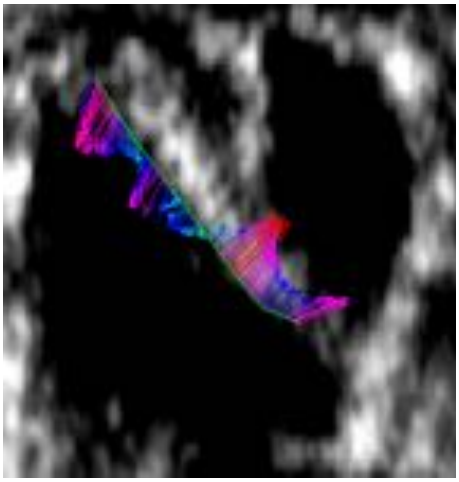
Figura 7.20. Flujo óptico sobre el primer corte de una secuencia aumentando la magnitud de los vectores 10 veces. (a) Con cuatro escalas y 45 iteraciones por escala. (b) Con 10 escalas y 10 iteraciones por escala. (c) Con 10 escalas y 45 iteraciones por escala.



(a)

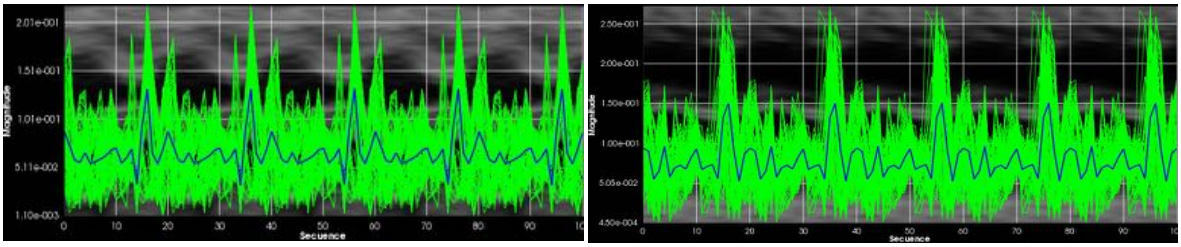


(b)

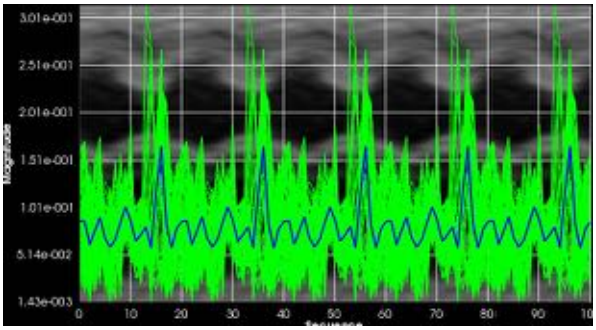


(c)

Figura 7.21. Flujo óptico sobre un contorno. (a) Con cuatro escalas, 45 iteraciones por escala y aumentando la magnitud 30 veces. (b) Con 10 escalas, 10 iteraciones por escala y aumentando la magnitud 20 veces. (c) Con 10 escalas y 45 iteraciones por escala y aumentando la magnitud 20 veces.



(a) (b)

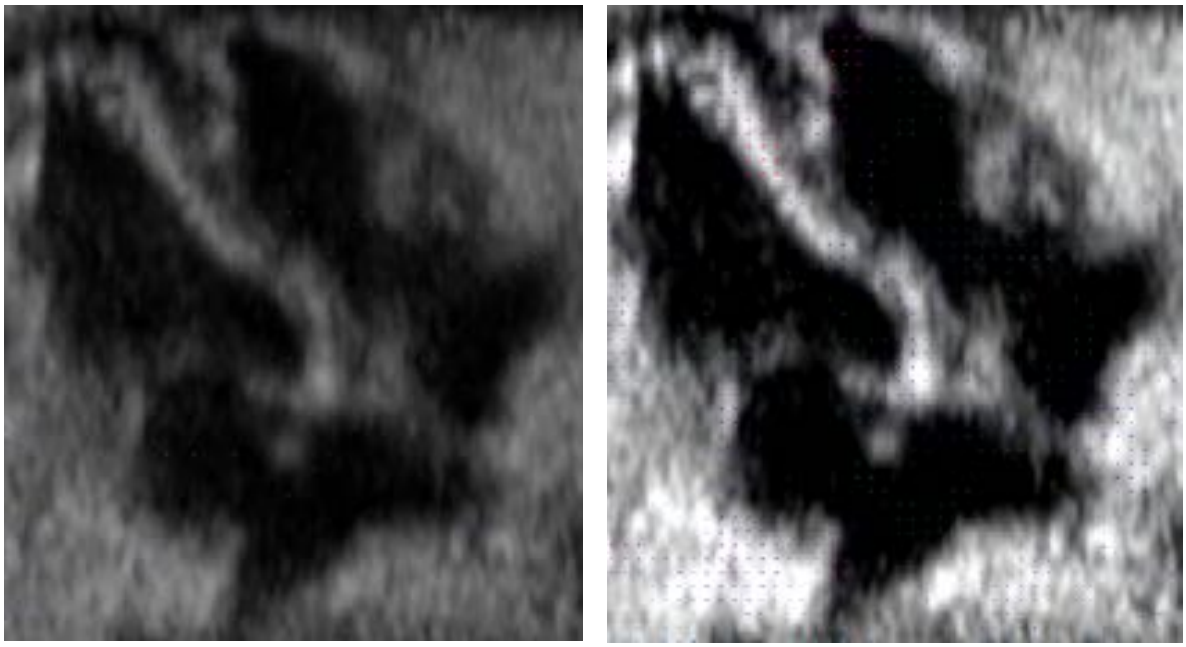


(c)

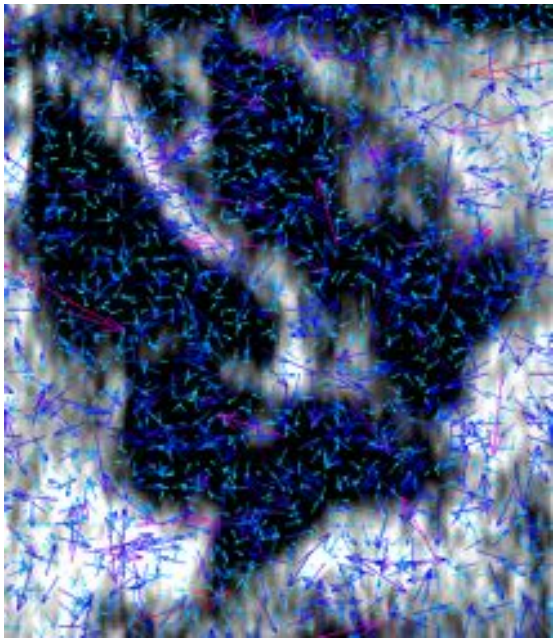
7.22. Gráfica de las magnitudes de flujo óptico sobre una secuencia de contornos utilizando 45 iteraciones por escala. (a) Con cuatro escalas y 45 iteraciones por escala. (b) Con 10 escalas y 10 iteraciones por escala. (c) Con 10 escalas y 45 iteraciones por escala.

Dentro de cada iteración se realiza el método SOR, que aproxima los valores asociados a su estimación de movimiento. Los vectores cambian su magnitud y dirección al aumentar las iteraciones dentro de la escala y dentro del método SOR, esto es mostrado en las Figuras 7.23., 7.24., 7.25. y 7.26.

Las iteraciones en la escala y en SOR no son equivalentes. Las Figuras 7.23.(a) y 7.24.(a) muestran el flujo óptico en el corte calculado con sólo una iteración en SOR, generando una estimación muy burda en comparación con 7.23.(b) y 7.24.(b) que utilizan 10. Con una estimación pobre, sus inexactitudes se acarrean en las siguientes iteraciones de la escala actual, empeorando en lugar de mejorar, como se observa en la Figura 7.23.(c).

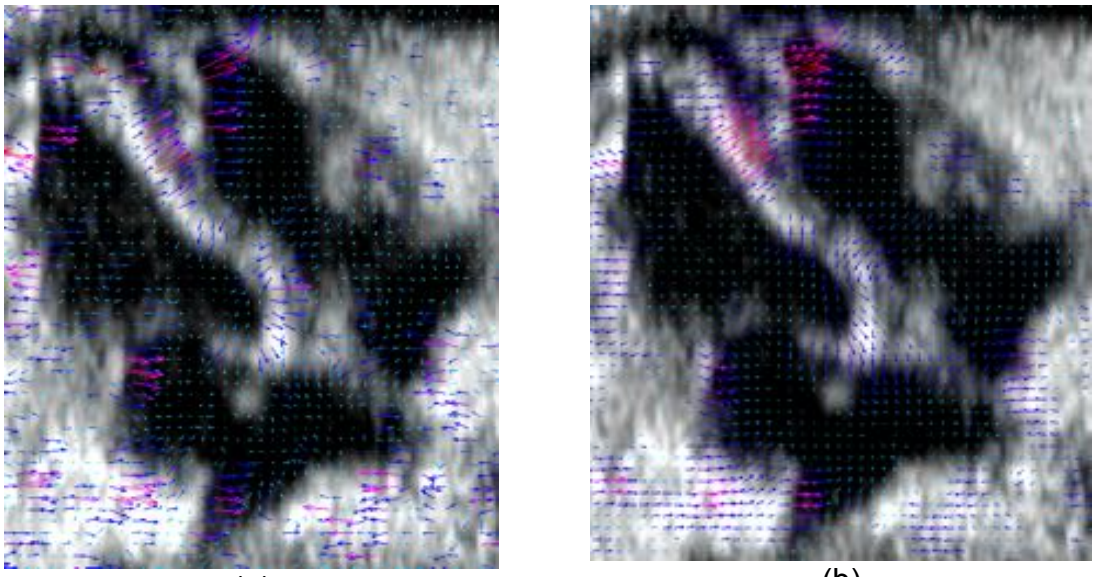


(a) (b)



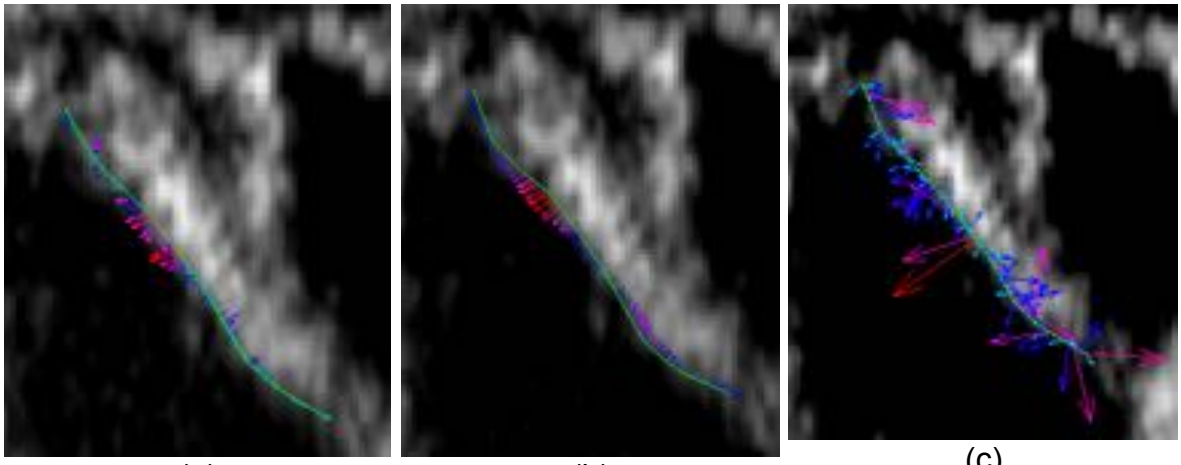
(c)

Figura 7.23. Flujo óptico del primer corte de una secuencia utilizando una escala y aumentando la magnitud de los vectores 10veces. (a) Con una iteración por escala y una iteración en el método SOR. (b) Con una iteración por escala y 10 iteraciones en el método SOR. (c) Con 10 iteraciones por escala y una iteración en el método SOR.



(a) (b)

Figura 7.24. Flujo óptico de las Figuras 7.23. (a) y (b) aumentando su magnitud. (a) Aumentado la magnitud de 7.23.(a) 480 veces. (b) Aumentado la magnitud de 7.23.(b) 80 veces.



(a) (b) (c)

Figura 7.25. Flujo óptico sobre un contorno utilizando una escala. (a) Con una iteración por escala, una iteración en el método SOR y aumentando la magnitud de los vectores 480 veces. (b) Con una iteración por escala, 10 iteraciones en el método SOR y aumentando la magnitud de los vectores 80 veces. (c) Con 10 iteraciones por escala, una iteración en el método SOR y aumentando la magnitud de los vectores 10 veces.

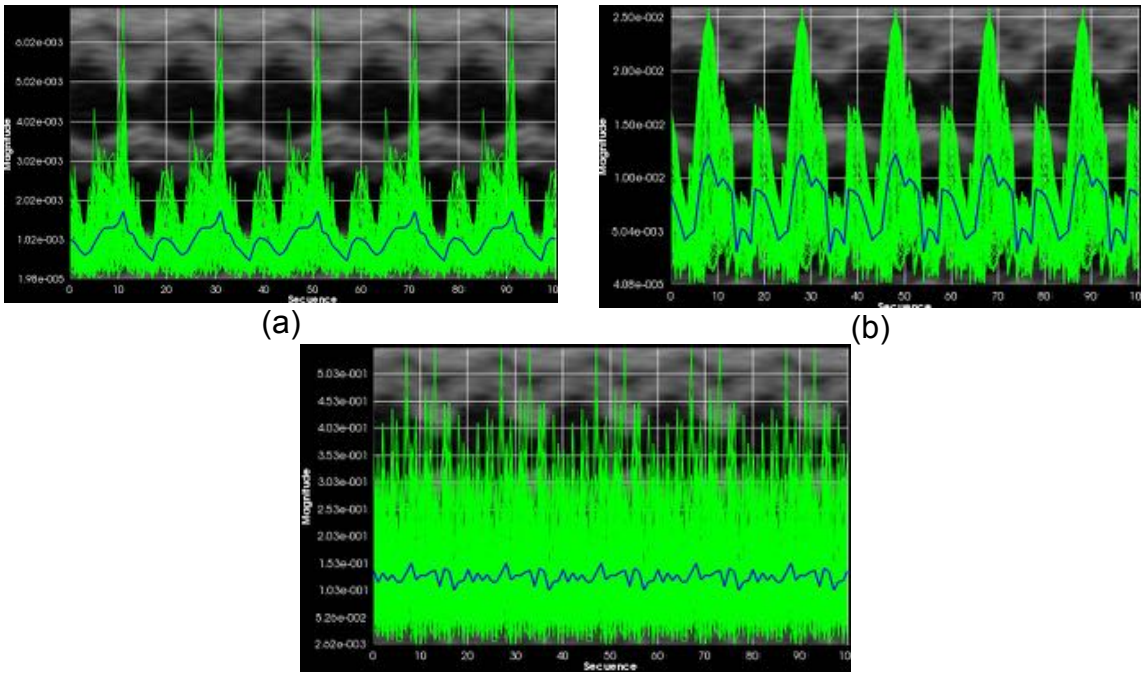
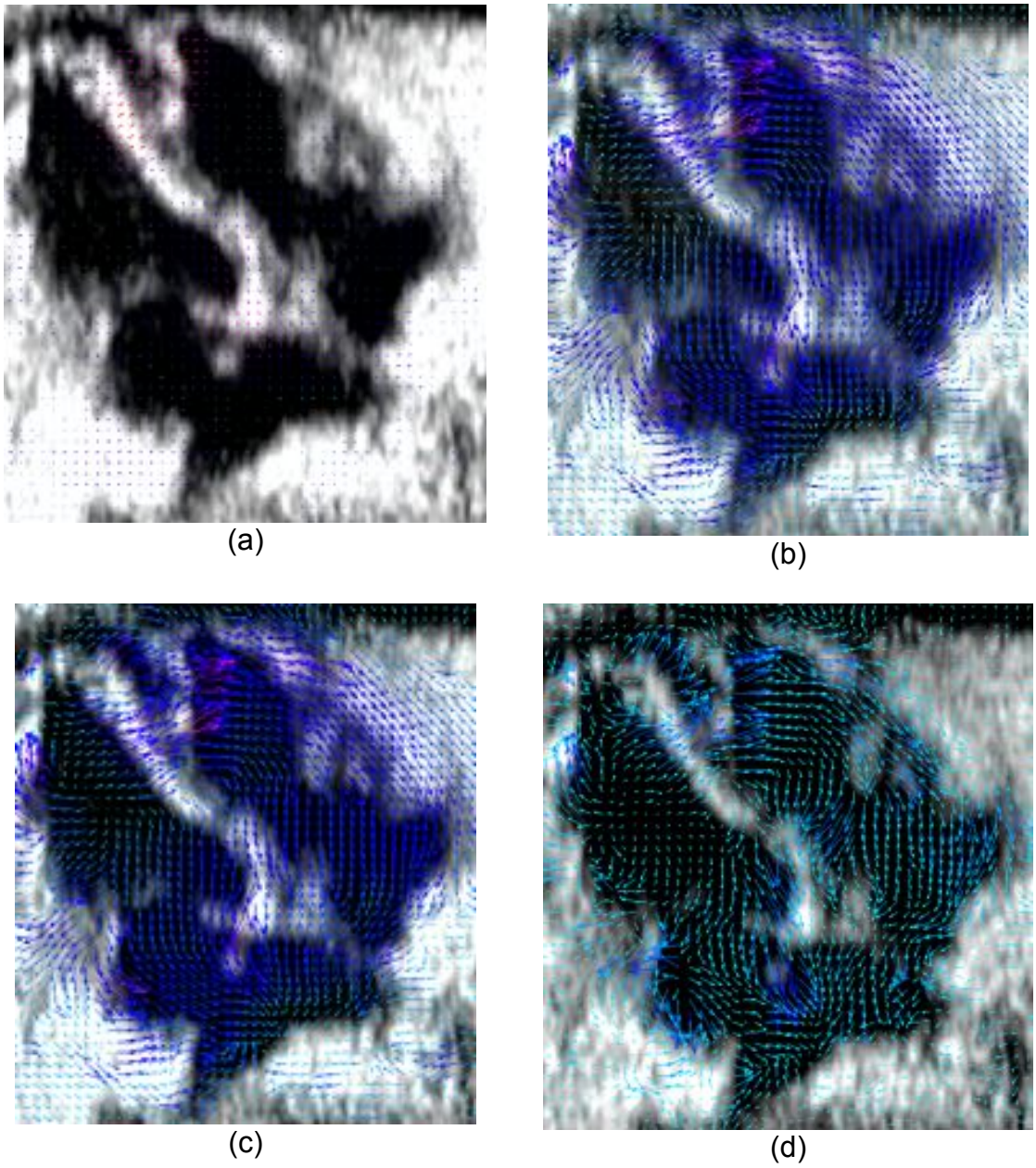


Figura 7.26. Gráfica de las magnitudes de flujo óptico sobre una secuencia de contornos utilizando una escala. (a) Con una iteración por escala y una iteración en el método SOR. (b) Con una iteración por escala y 10 iteraciones en el método SOR. (c) Con 10 iteraciones por escala y una iteración en el método SOR.

Tanto los contornos y gráficas de las Figuras 7.25. y 7.26. muestran en (c) que al aumentar las iteraciones por escala sin una buena estimación sólo resulta en valores erráticos. Por lo que al no asignar suficientes iteraciones al método SOR es mejor no realizar varias escalas como se muestra en 7.26.(a) donde el mejor resultado de estos 3 es (b), ya que es más similar al contorno y gráfica del inciso (a) de las Figuras 7.15. y 7.16. que tiene 30 iteraciones en SOR.

Al comparar las Figuras 7.27., 7.28. y 7.29. con los flujos ópticos y gráficas que coinciden en sus números de escalas e iteraciones pero con 30 ciclos como máximo en SOR, se encuentra que los cambios son demasiado sutiles como para ser visualmente significativos. Ya que, en comparación con las otras Figuras con escalas e iteraciones correspondientes, los flujos sobre la imagen y el corte presentan las mismas coloraciones y direcciones, las flechas sobre los contornos tienen las mismas tendencias y sus gráficas asociadas tienen las mismas formas y, a excepción de 7.29.(a) respecto a 7.16.(a), tienen los mismos rangos.



(a) (b) (c) (d)

Figura 7.27. Flujo óptico sobre el primer corte de una secuencia aumentando la magnitud de los vectores 10 veces. (a) Con una escala, una iteración por escala y 1350 iteraciones máximas en SOR. (b) Con 10 escalas, una iteración por escala y 300 iteraciones máximas en SOR. (c) Con 10 escalas, una iteración por escala y 1350 iteraciones máximas en SOR. (d) Con 10 escalas, 10 iteraciones por escala y 1350 iteraciones máximas en SOR.

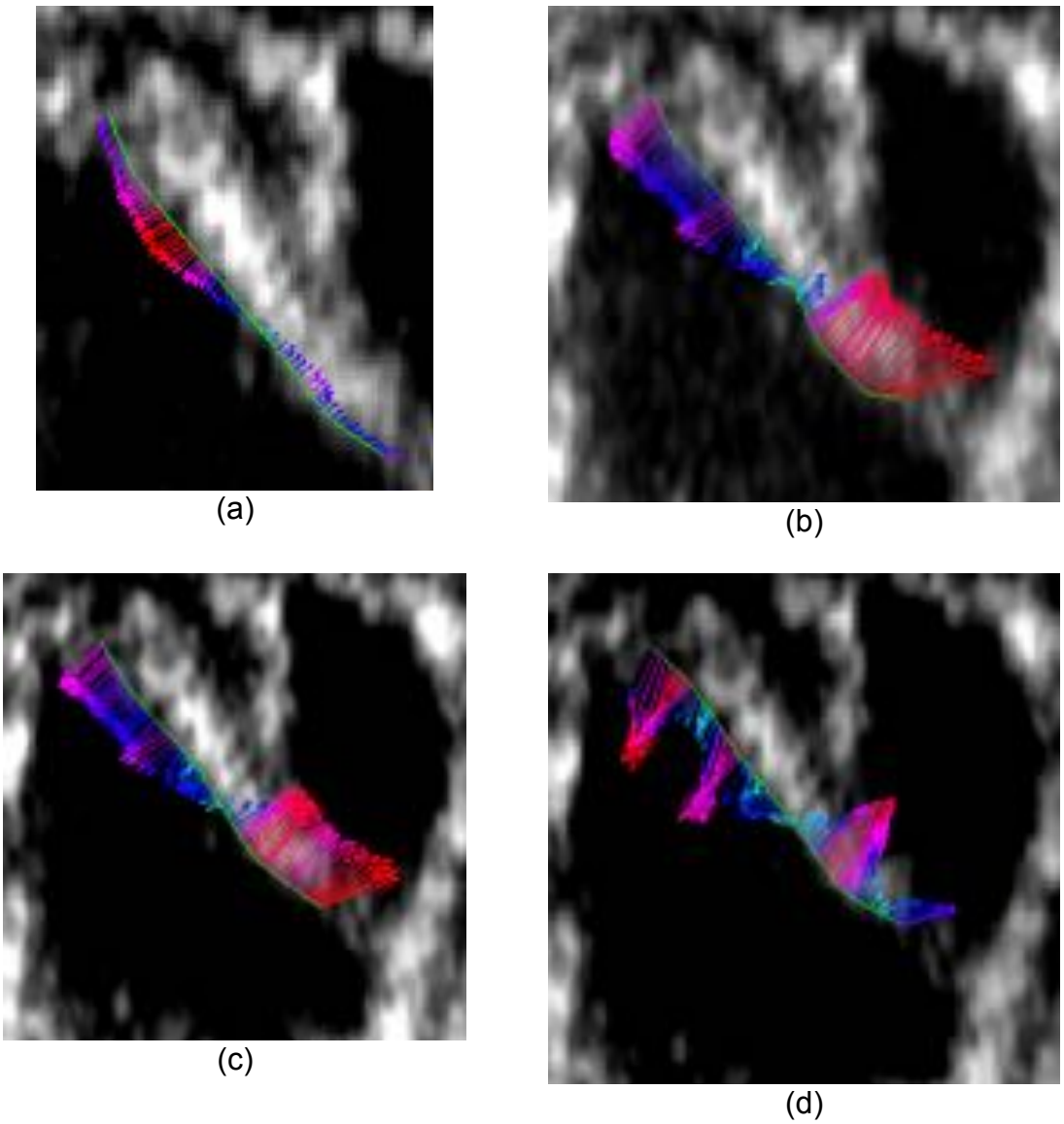


Figura 7.28. Flujo óptico sobre un contorno. (a) Con una escala, una iteración por escala, 1350 iteraciones máximas en SOR y aumentando la magnitud de los vectores 80 veces. (b) Con 10 escalas, una iteración por escala, 300 iteraciones máximas en SOR y aumentando la magnitud de los vectores 20 veces. (c) Con 10 escalas, una iteración por escala, 1350 iteraciones máximas en SOR y aumentando la magnitud de los vectores 20 veces . (d) Con 10 escalas, 10 iteraciones por escala, 1350 iteraciones máximas en SOR y aumentando la magnitud de los vectores 20 veces.

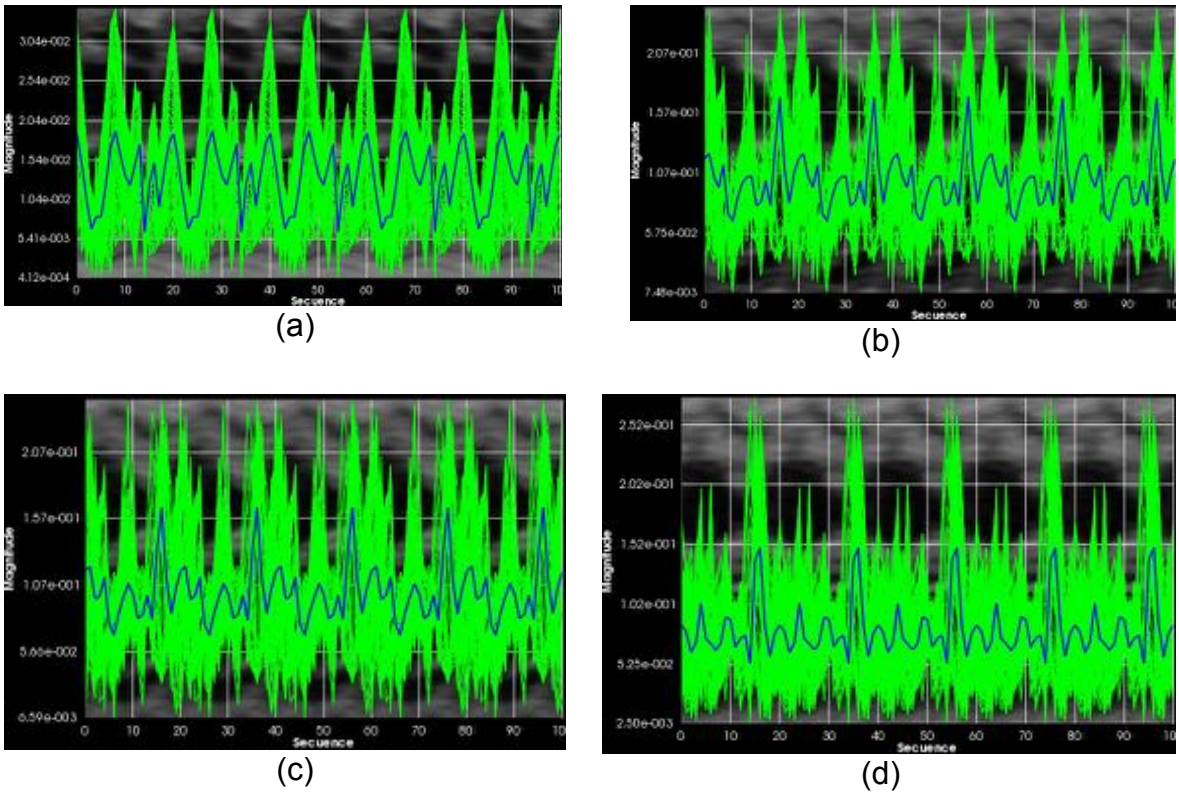


Figura 7.29. Gráfica de las magnitudes del flujo óptico sobre una secuencia de contornos. (a) Con una escala, una iteración por escala y 300 iteraciones máximas en SOR. (b) Con 10 escalas, una iteración por escala y 300 iteraciones máximas en SOR. (c) Con 10 escalas, una iteración por escala y 1350 iteraciones máximas en SOR. (d) Con 10 escalas, 10 iteraciones por escala y 1350 iteraciones máximas en SOR.

Una vez que se estabiliza la estimación, el aplicar más iteraciones en el método SOR no genera cambios significativos independientemente de la cantidad de escalas e iteraciones asociadas.

Capítulo 8
Conclusiones

En la presente tesis se muestra el desarrollo todo un sistema de análisis de imágenes de ultrasonido fetal que incluye la visualización de las imágenes y los resultados de sus procesamientos, la manipulación de las imágenes, la obtención del modo M, la implementación del algoritmo de flujo óptico mediante los coeficientes de Hermite de las imágenes y la segmentación manual de un contorno de interés sobre éste.

Las imágenes 4D y sus datos son leídos por el sistema para poder visualizarse y manipularse por el usuario, de tal manera que el corte más adecuado para el análisis sea elegido por un experto. Con ello se genera la secuencia de imágenes 2D con la que se hacen todos los procesamientos descritos en éste desarrollo.

Para el cálculo del modo M, el usuario debe seleccionar el sitio en donde quiere que éste sea obtenido ya dependiendo de las estructuras por las que pase, mostrará la información correspondiente.

El método de flujo óptico se aplica sobre pares de imágenes por lo que hay que configurar la secuencia en que toma a los cortes. A cada corte se le obtienen los coeficientes de Hermite y de ellos se obtienen los ángulos de máxima energía para generar los coeficientes rotados de Hermite y sus derivadas. Ya que la estimación de movimiento parte de los coeficientes de Hermite de las imágenes, su resultado varía dependiendo de como son obtenidos estos coeficientes.

Para la presente Tesis, se desarrolló el algoritmo para la estimación de derivadas parciales en dos dimensiones hasta tercer orden que da resultados equivalentes a calcular los filtros de Hermite en dos dimensiones y posteriormente aplicárselos a las imágenes pero en menor tiempo. Es por esto que la imagen necesita un procesamiento antes para que las orillas tengan con qué hacer las operaciones, y después para obtener los coeficientes de la Transformada de Hermite del mismo tamaño que la imagen original. El algoritmo también puede aplicarse a la obtención de elementos matemáticos directa o parcialmente que requieran derivadas como el gradiente.

En la implantación realizada, la utilización de la Transformada rápida de Hermite da mejores resultados que la utilización de los filtros de Hermite, como se observa en los errores y los tiempos que se obtuvieron de la estimación de movimiento.

Los parámetros e iteraciones involucradas en el flujo óptico dependen del tipo de imágenes con las que se trabaja y el resultado que se desea obtener.

Dependiendo de los valores que se les asignen la magnitud de los vectores, el suavizado del flujo y los grandes desplazamientos se ven afectados en mayor o menor medida además de que el tiempo que tarda efectuar el cálculo aumenta cuando el número de iteraciones y la constante de escalamiento aumentan.

El conjunto de todos los elementos generados, componen la interfaz que es una pequeña parte de lo se espera sea una vital herramienta en el análisis fetal realizado en el Instituto Nacional de Perinatología.

El conjunto de herramientas proporcionado por las librerías ITK y VTK son parte esencial de este trabajo, brindan una gran variedad de métodos de procesamiento y visualización que pueden implementarse de distintas maneras dependiendo de lo que se requiera. De forma similar, Qt simplifica eficientemente la integración de interfaces gráficas y su interacciones programadas en C++.

8.1. Trabajo Futuro

El equipo usado comúnmente en el Instituto Nacional de Perinatología para tomar las ecocardiografías es el que genera los archivos con extensión .vol sin embargo existen otros formatos en los que se guardan las imágenes médicas por lo que se necesita implementar la lectura y visualización para ellos.

Del modo M es posible obtener información del ciclo cardíaco que le es útil al médico. Si se implementan más herramientas a su interfaz, se pueden generar valiosos datos para el análisis del corazón.

El filtro que estima el movimiento está diseñado para calcularlo entre 2 imágenes. Ya que se aplica en toda una secuencia, es posible agilizarlo al modificarlo para tomar todas las imágenes y reutilizar los coeficientes de la Transformada de Hermite y algunos valores que se generan a partir de ellos.

El hacer contornos manuales para todas las imágenes de una secuencia es tedioso, tardado y dependiente del médico. Se pretende que el proyecto sustituya esto por una segmentación automatizada de estructuras cardíacas como el ventrículo izquierdo.

El flujo óptico varía dependiendo de los parámetros asignados por lo que necesita validarse para definir los más óptimos según los criterios que los médicos ocupen.

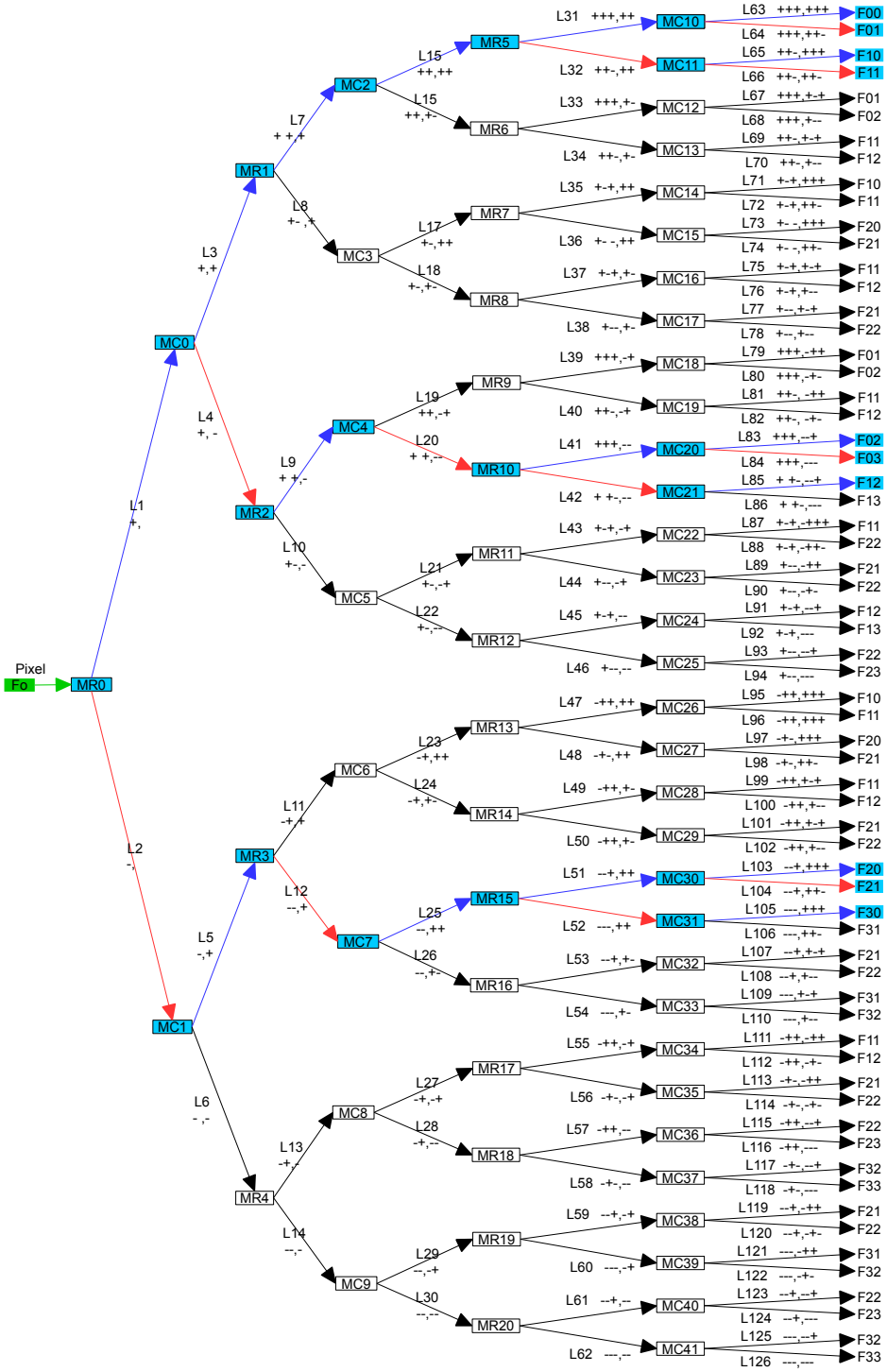
El flujo óptico sobre una secuencia de contornos se toma del obtenido para la imagen entera pero si ya se cuenta con el contorno definido, es probable que sea más práctico calcular la estimación de movimiento únicamente con los píxeles

seleccionados.

Para el análisis cardíaco existen métricas definidas que se tiene contemplado integrar para procesar los datos obtenidos de la estimación de movimiento. Sus resultados se visualizarán en la gráfica de la interfaz y otras, con más elementos de interacción.

El sistema requiere que el modo M sea calculado antes que la estimación de movimiento para poder integrarlo a la ventana de flujo óptico. El modo M es mucho más rápido de obtener que la estimación de movimiento por lo que habría que anexar este proceso a la interfaz de flujo óptico en caso de que se requiera recalcularlo e integrarlo.

Anexo A
Diagrama del Árbol Completo para el Cálculo de los Coeficientes de la Transformada de Hermite de Orden 3 en 2 Dimensiones



Este diagrama representa el desarrollo del árbol completo del algoritmo de estimación de derivadas parciales en dos dimensiones de primer, segundo y tercer orden con $N=3$ resaltando el camino más corto de las derivadas a utilizar en la obtención de los coeficientes de Hermite, necesarios para el cálculo del flujo óptico.

Para seleccionar las derivadas a utilizar, primero se identifican las derivadas que no se repiten, en este caso son F00, F30, F03. De las ramas recorridas para las derivadas que ya se tienen, se selecciona el camino más corto a las faltantes.

Las derivadas repetidas, ya sea que fueron o no elegidas e independientemente de la sección del árbol en que se encuentran, son equivalentes.

donde:

- Fo es el píxel de entrada.
- MR son los registros de almacenamiento para el retraso de los renglones.
- MC son los registros de almacenamiento para el retraso de las columnas.
- L representa a los registros de almacenamiento de los resultados intermedios.
- +, -: se apilan de izquierda a derecha indicando que operación se efectúa (adición o sustracción) con el resultado intermedio. Antes de la coma para MR y después para MC. El número de signos – indica el orden de la derivada.
- F00 a F33 son las derivadas parciales resultantes.

Anexo B
Configuración de Qt4 para 64 bits

El archivo de instalación para este proyecto es qt-opensource-windows-x86-vs2008-4.8.7, que es compatible con Visual Studio 2008 y con la librería de VTK 5.10.1. Ya instalado Visual Studio, se procede a instalar Qt. Este framework también cuenta con un wizard para su instalación.

Las versiones de Qt 4.x (y de forma particular 4.8.7 para el sistema en desarrollo) sólo vienen inicializadas para 32 bits por lo que posteriormente a la instalación se reconfigura y recomplila para ser compatible con una aplicación de 64 bits [46]. Para esto, se comienza con la instalación de Visual Studio Add-in 1.1.11 para Qt4. Este es un complemento de Qt4 para Visual Studio que también tiene su propio wizard de instalación, como se muestra en la Figura B.1.

Figura B.1. Wizar de Instalación de Visual Studio Add-in 1.1.11

Con el complemento instalado se procede a abrir el archivo qmake.config localizado en la carpeta de instalación de Qt y se cambian las líneas:

```
DEFINES                    += UNICODE WIN32
QMAKE_COMPILER_DEFINES  += _MSC_VER=1500 WIN32
```

por:

DEFINES += UNICODE WIN64 QT_LARGEFILE_SUPPORT
QMAKE_COMPILER_DEFINES += _MSC_VER=1500 WIN64

Esta modificación se muestra en la Figura B.2.

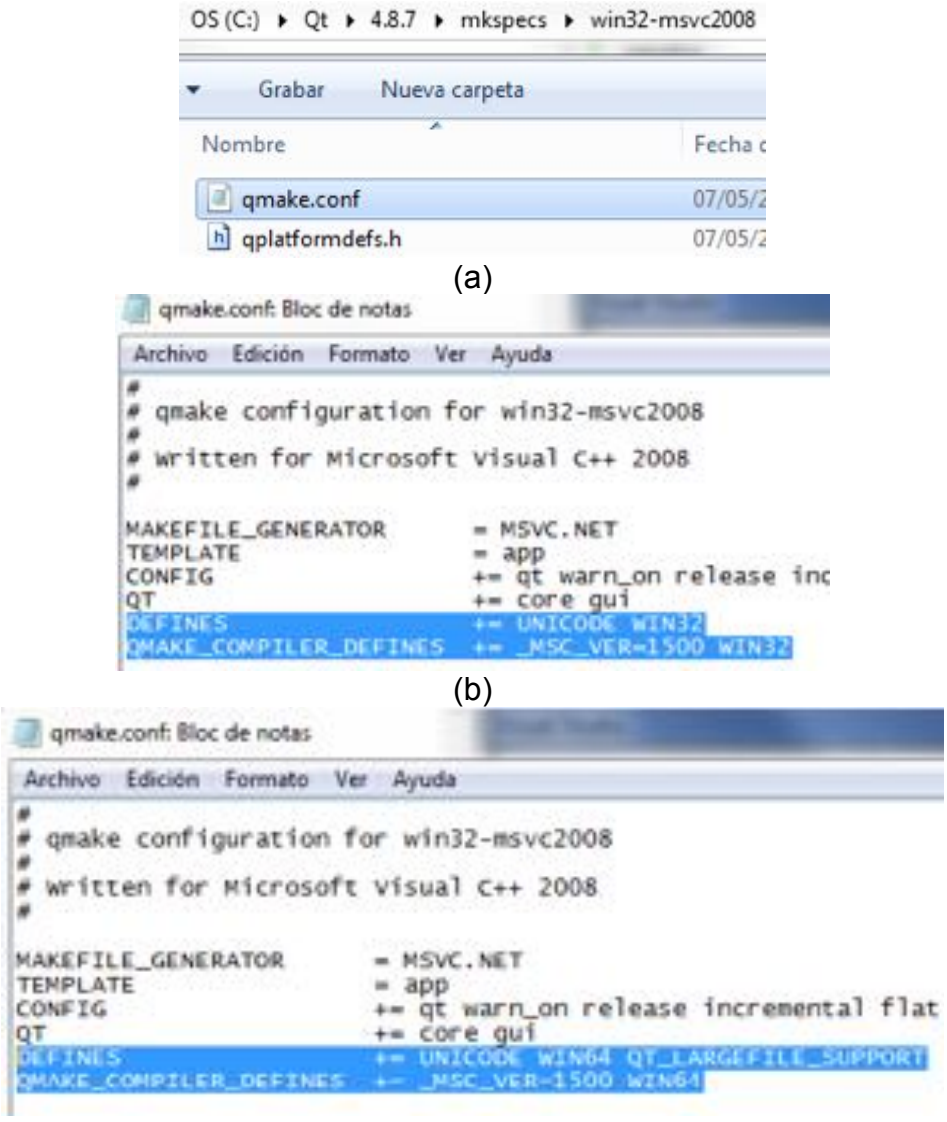


Figura B.2. Modificación del archivo qmake.config. (a) Archivo qmake.config localizado en la carpeta de instalación de Qt. (b) Líneas a cambiar. (c) Líneas cambiadas.

Después de modificar el archivo se recopila Qt a través de la línea de comandos de Visual Studio con el compilador Visual Studio 9 2008 Win64, para reconfigurarlo a 64 bits en lugar de 32 se siguen los siguientes pasos:

1. Abrir la línea de Comandos desde Visual Studio (figura B.3.):
    1. Desplegar la pestaña de herramientas.
    2. Seleccionar Visual Studio 2008 Command Promt.

Figura B.3. Línea de Comandos de Visual Studio 2008.

2. Especificar el entorno de Visual Studio 2008 para 64 bits (figura B.4.):
    Seleccionar “Visual Studio 2008 x64 Win64 Command Promt.lnk“. El nombre de la ventana cambia para indicar que que se está usando Win64 para un entorno de 64 bits.

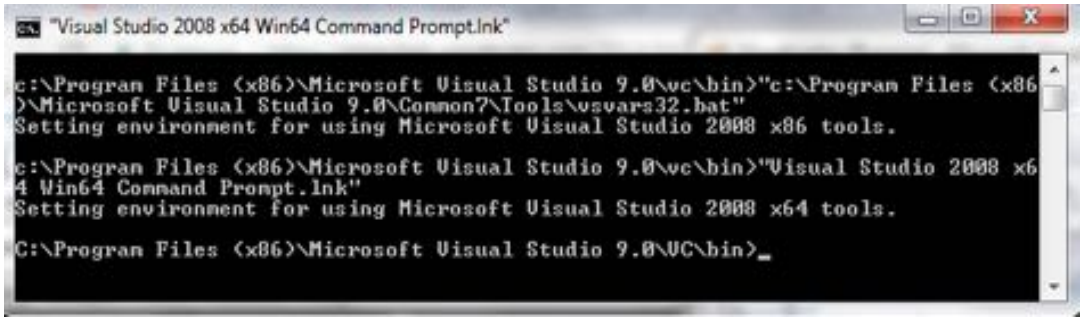


Figura B.4. Línea de comandos de Visual Studio 2008 con entorno a 64 bits.

3. Limpiar Qt:
    1. Posicionarse en la carpeta donde se instaló Qt.
    2. Teclear el comando nmake con la opción distclean (figura B.5.).

Figura B.5. Limpiar Qt desde la línea de comandos.

4. Reconfigurar Qt (figura B.6.):

   En la carpeta actual escribir el comando configure seguido de los parámetros -debug-and-release y -platform win32-msvc2008 si se desea se le pueden agregar más opciones [47] dependiendo de como se quiera configurar.

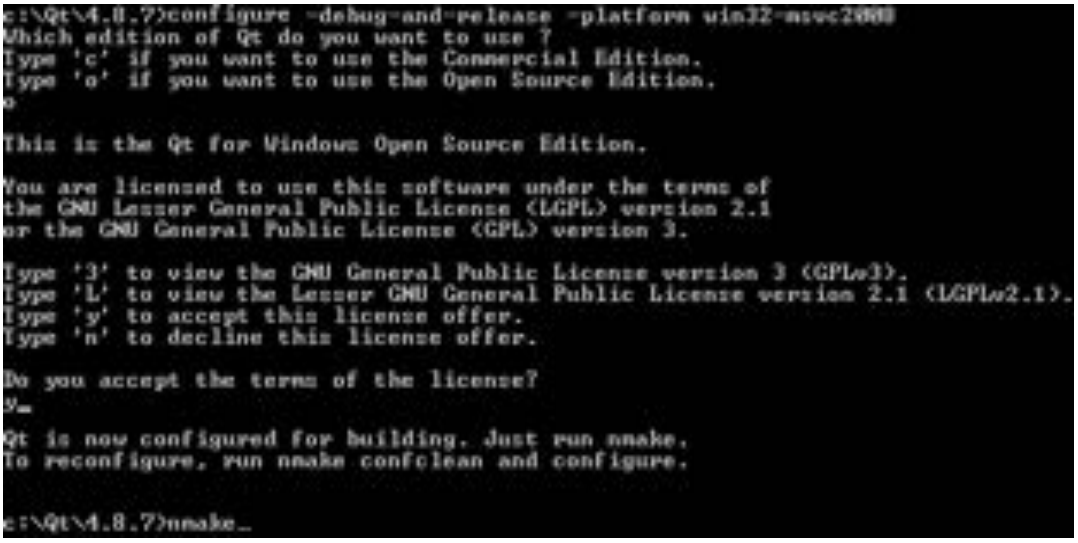


Figura B.6. Reconfiguración de Qt desde línea de comandos.

A lo largo del proceso aparece la advertencia de que “QT_LARGEFILE_SUPPORT”, como se muestra en la Figura B.7., no está definida previamente pero esto no afecta la reconfiguración [48].

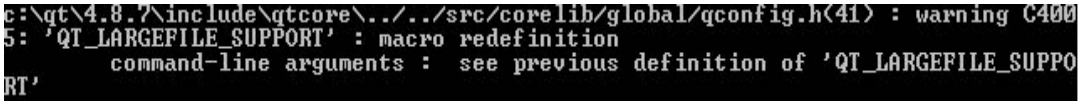


Figura B.7. Advertencia “QT_LARGEFILE_SUPPORT”.

Anexo C
Compilación de ITK y VTK

Para el sistema a desarrollado en la presente tesis el compilador que se utiliza es Visual Studio 9 2008 Win64, perteneciente a Visual Studio. El proceso de compilación de ITK y VTK junto con su arquitectura y aplicación están explicados detalladamente en sus guías respectivas [41], [42].

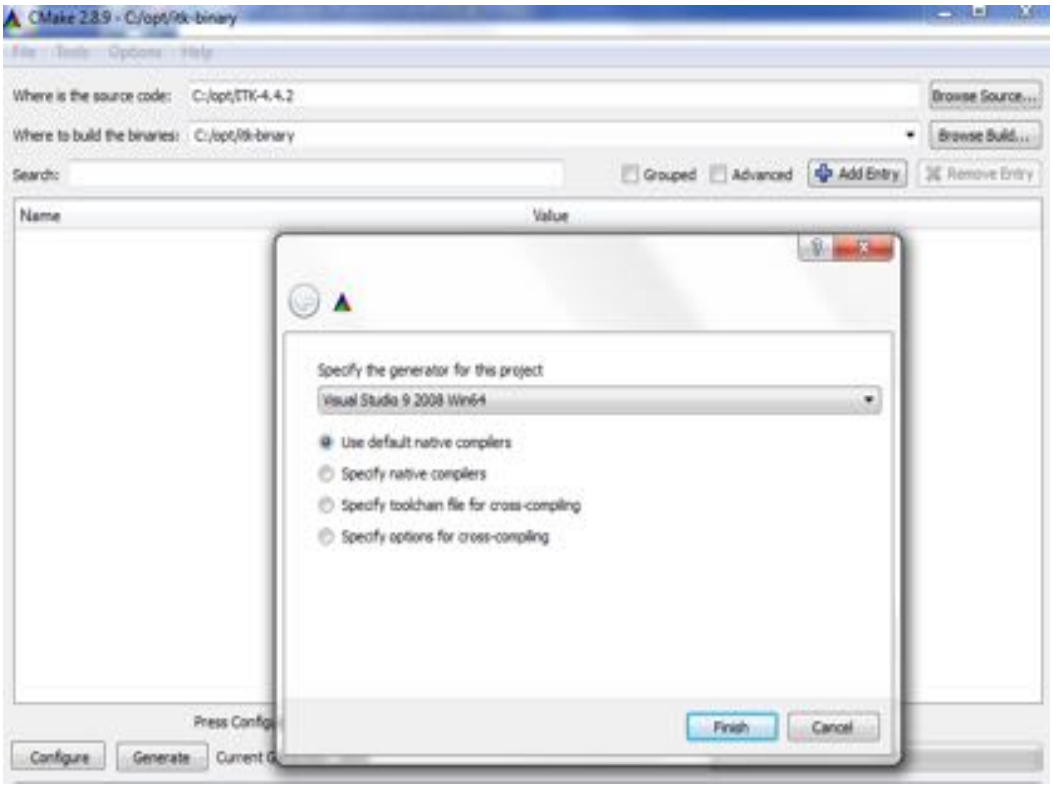


Figura C.1. Selección del compilador Visual Studio 9 2008 Win64 para CMake.

Pasos realizados para la generación de las librerías de ITK:

1. Descargar ITK y descomprimir su contenido en una carpeta.

2. Crear dentro de la raíz del sistema, la carpeta destinada a almacenar la librería.

3. Abrir CMake, indicando las carpetas origen (donde se están los archivos descomprimidos de ITK) y destino (donde se almacena el resultado de la compilación).

4. Oprimir el botón de configurar y seleccionar el compilador Visual Studio 9 2008 Win64 (figura C.1.).

5. Seleccionar las propiedades requeridas y se presiona nuevamente el botón de configurar. Si lo hace correctamente aparecerá el texto Configuring done (figura C.2.), en caso contrario aparecerá en rojo el error.

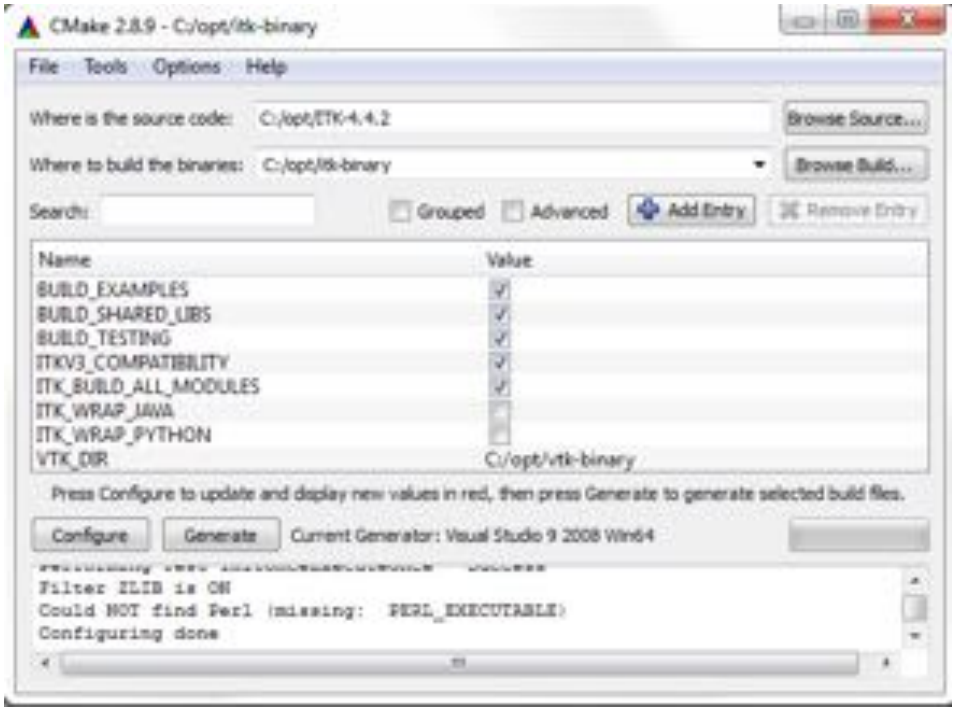
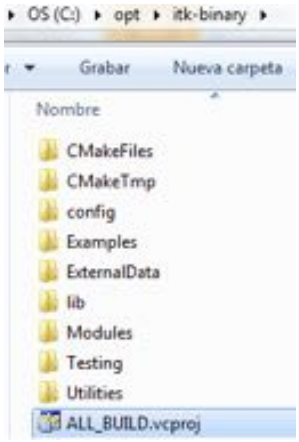


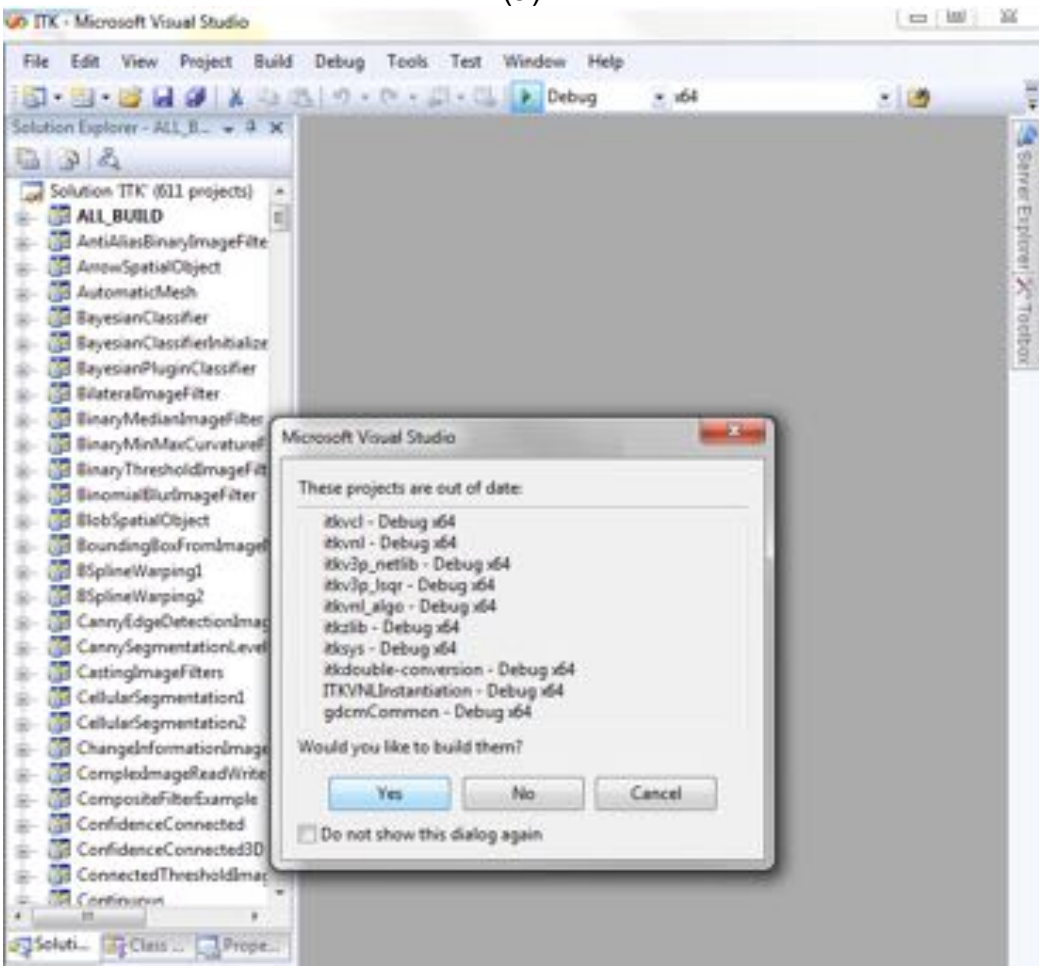
Figura C.2. Configuración de la librería ITK en CMake.

6. Oprimir el botón de generar. Si lo hace correctamente aparecerá el texto Generating done, en caso contrario aparecerá en rojo el error.

7. Una vez generado correctamente, con Visual Studio se abre el archivo creado ALL_BUILD, localizado en la carpeta destino y se compila. Esto se muestra en la Figura C.3.



(a)



(b)

Figura C.3. Compilación de ITK en Visual Studio. (a) archivo ALL_BUILD creado por CMake. (b) Compilación del archivo ALL_BUILD.

8. Se agrega la dirección de la carpeta Debug que se generó en la compilación con Visual Studio a la variable del sistema PATH (figura C.4.).

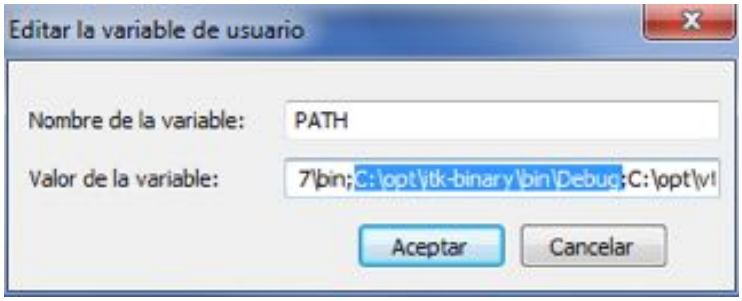


Figura C.4.Carpeta Debug de ITK en variable de entorno PATH.

Así como se genera la librería ITK, se procede a crear VTK siguiendo los mismos pasos seleccionando las propiedades mostradas a continuación:

Figure C.5. Propiedades para VTK en Cmake.

Posteriormente a la instalación de ITK y VTK se regenera la librería de ITK para hacela compatible VTK, para esto se abre CMake con las carpetas que se usaron al generar la librería por primera vez, se reconfigura y regenera agregando la opción Module_ITKVtkGlue, después se recompila con Visual Studio. Como ya esta agregada la carpeta Debug a la variable de entorno PATH ya no se modifica.

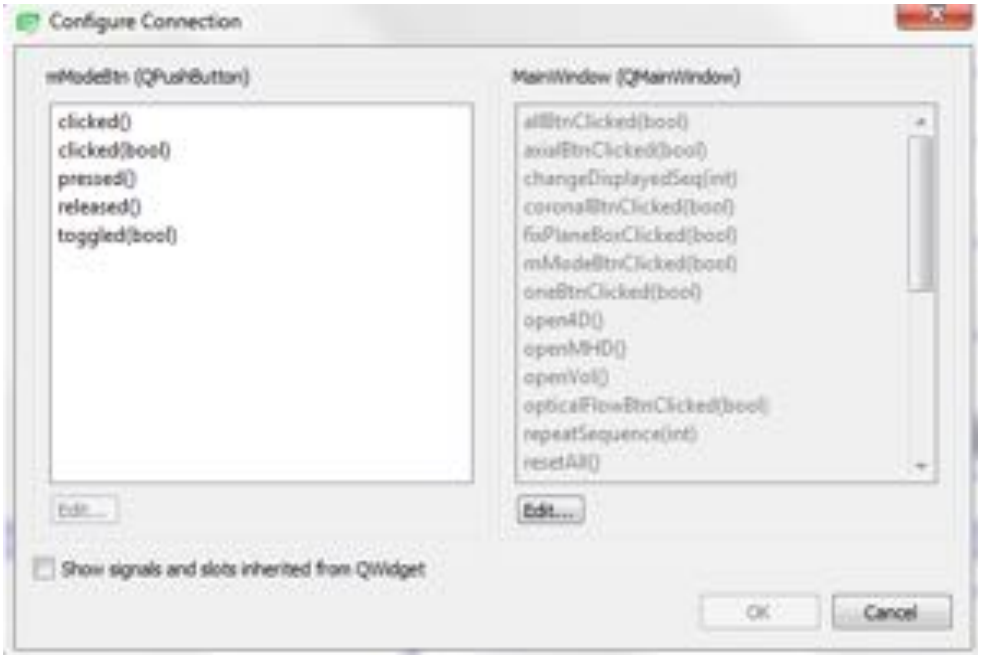
Con las librerías hechas para desarrollo del sistema, al igual que estas, se utiliza CMake para generar el proyecto de Visual Studio con la diferencia de que no es necesario seleccionar ninguna opción adicional en CMake.

Anexo D
Creación de Elementos de Interacción en una Interfaz

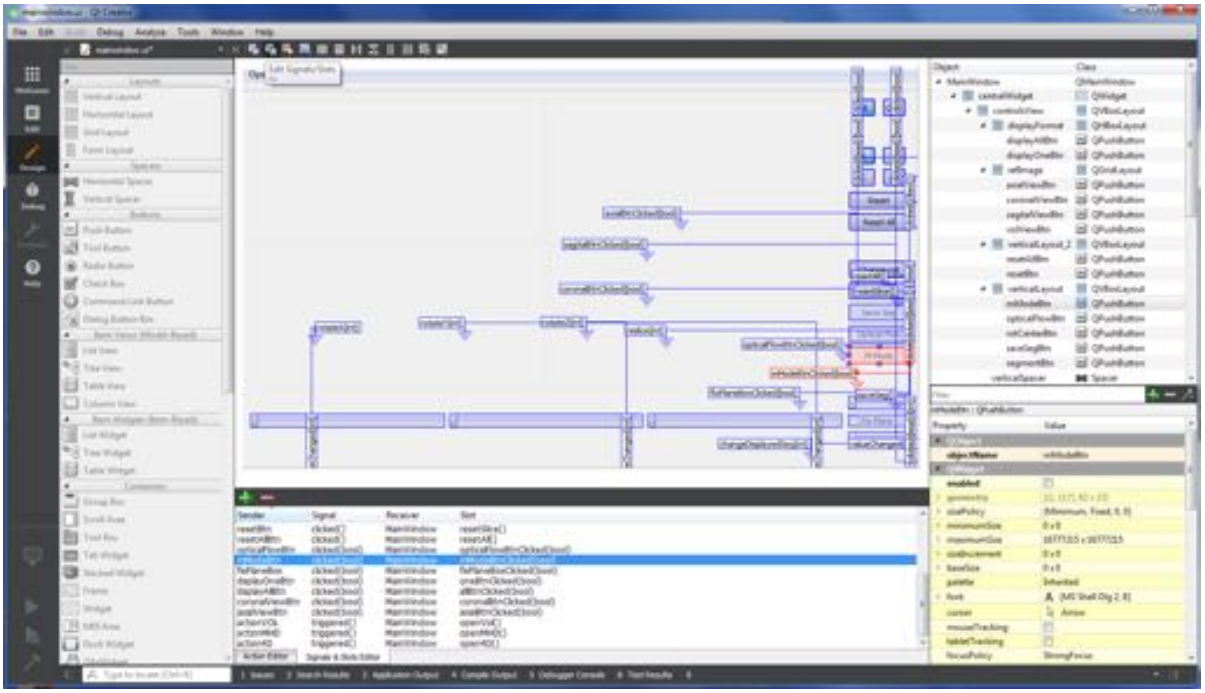
Los elementos de interacción se crean en los archivos .ui con el software Qt Creator y se ligan a una función perteneciente de la clase asociada a la ventana que puede o no recibir algún paramento dependiendo del tipo y utilización del elemento.

Qt Creator cuenta con distintas herramientas que pueden ser integradas a la interfaz para interactuar con ella y organizar los elementos que contienen. Se asignan arrastrándolas una por una desde el menú donde están expuestas hasta la zona perteneciente a la interfaz y soltarla en el lugar que se le quiere asignar. Una vez creados los elementos para la interacción, se asocia cada uno a una función definida dentro de la clase de su ventana. En la Figura D.1. se muestra la interfaz principal en Qt Creator con sus señales asociadas.

La comunicación con la función se crea en Qt Creator, asignando una señal al elemento mediante la activación de la opción Edit Signals/Slots para ver las señales asociadas a cada elemento y poder modificarlas o agregar alguna. Para esto último se se arrastra el mouse mientras se presiona el botón izquierdo desde el elemento y se suelta el botón cuando aparezca un símbolo similar al símbolo eléctrico de tierra. Inmediatamente después, aparece una ventana que dependiendo del elemento, en la parte izquierda muestra el tipo de señales que puede tener para elegir una de ellas y en el otro extremos, cuando ya se haya elegido el tipo de señal, se escoge la función con la que se relacionara. En caso de que la función asociada no se encuentre puede declararse con la opción de Edit.



(a)

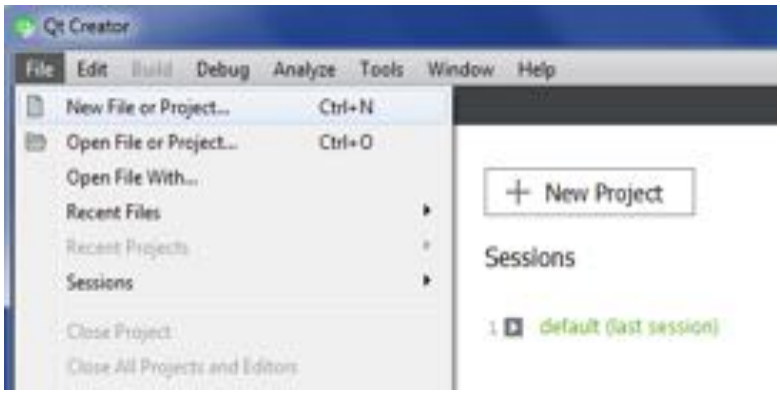


(b)

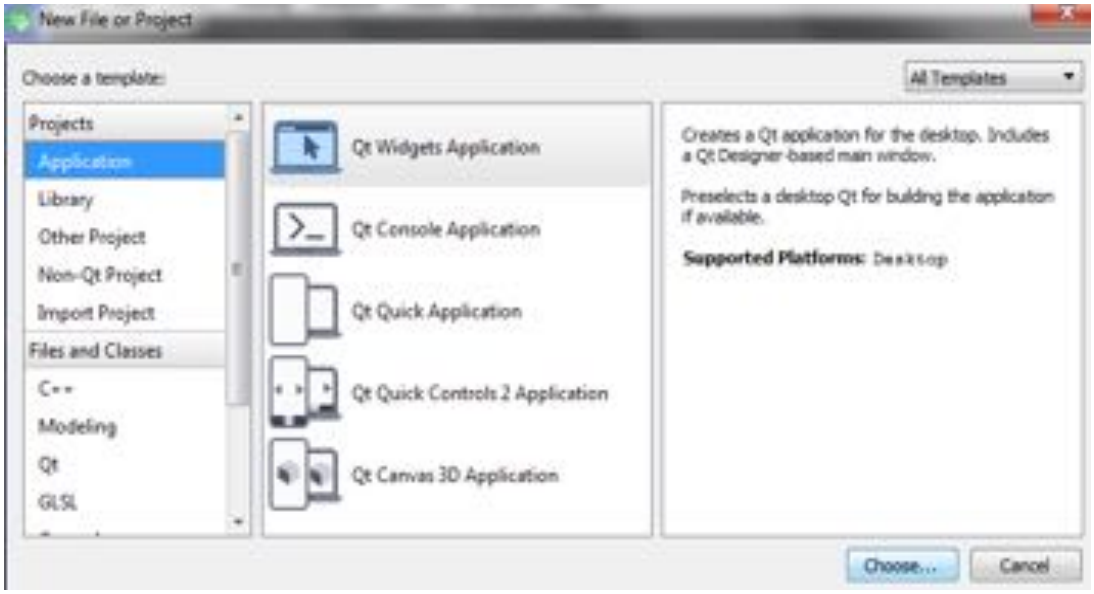
Figura D.1. Edición de las señales de los elementos de la interfaz con Qt Creator. (a) Ventana para la asignación de señales. (b) Interfaz con señales asociadas.

Anexo E
Generación de una ventana de Qt asociada a VTK

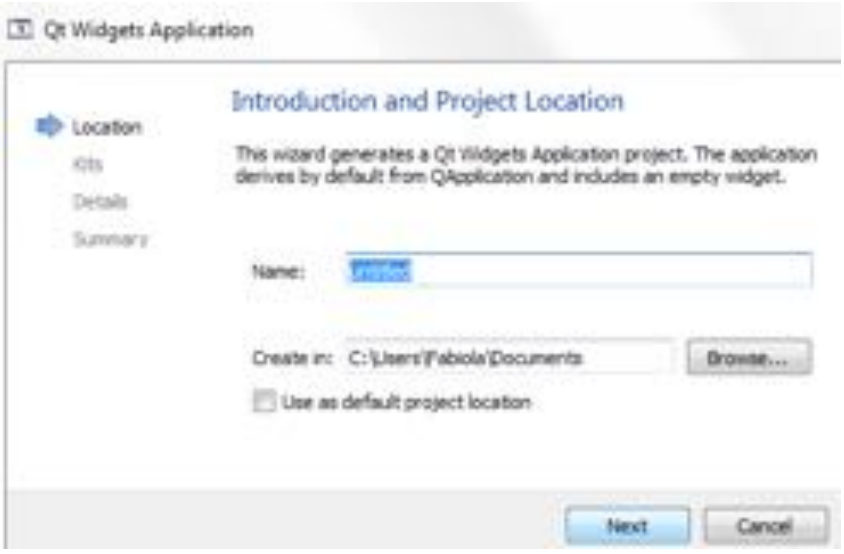
Usando de ejemplo la creación de la ventana del modo M para visualizarlo sin interferir con lo mostrado en la interfaz inicial, se crea una nueva interfaz con sus clases correspondiente. Esta recibe el modo M cuando es creada. En la Figura E.1. se muestra la secuencia para crearla con Qt Creator.



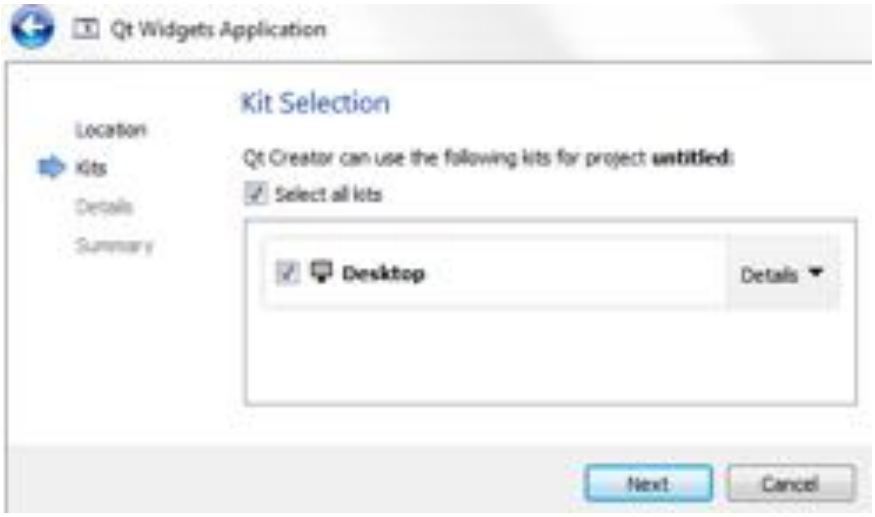
(a)



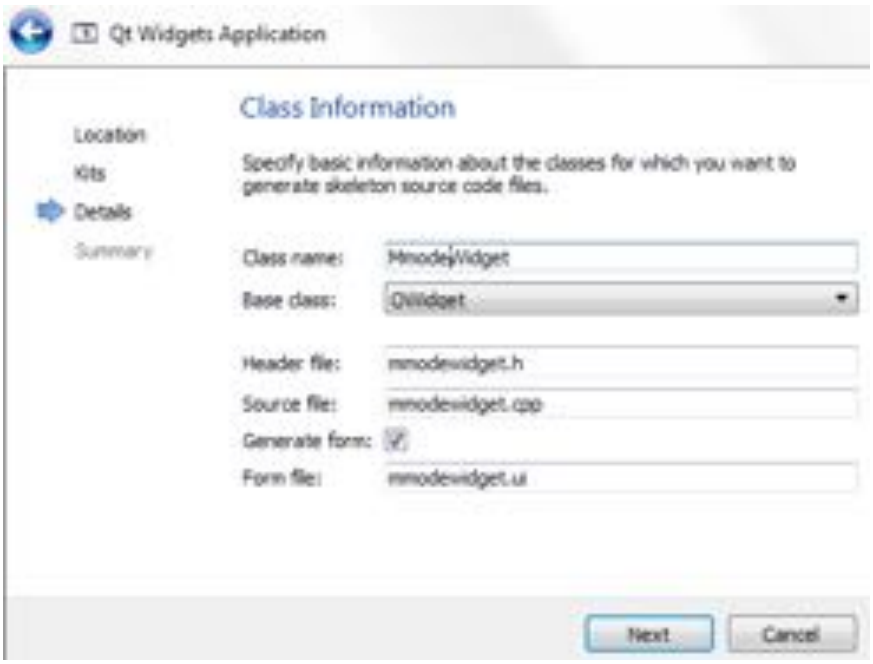
(b)


Qt Widgets Application
Introduction and Project Location
Location
Kits
Details
Summary
This wizard generates a Qt Widgets Application project. The application derives by default from QApplication and includes an empty widget.
Name:
Create in:
C:\Users\Fabiola\Documents
Browse...
Use as default project location
Next
Cancel

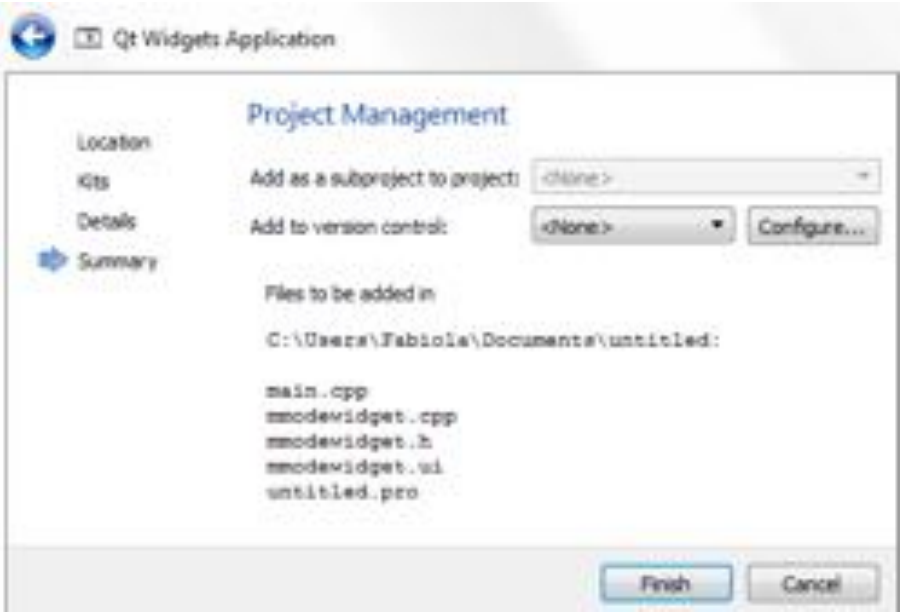
(c)


Qt Widgets Application
Kit Selection
Location
Kits
Details
Summary
Qt Creator can use the following kits for project untitled:
Select all kits
Desktop
Details
Next
Cancel

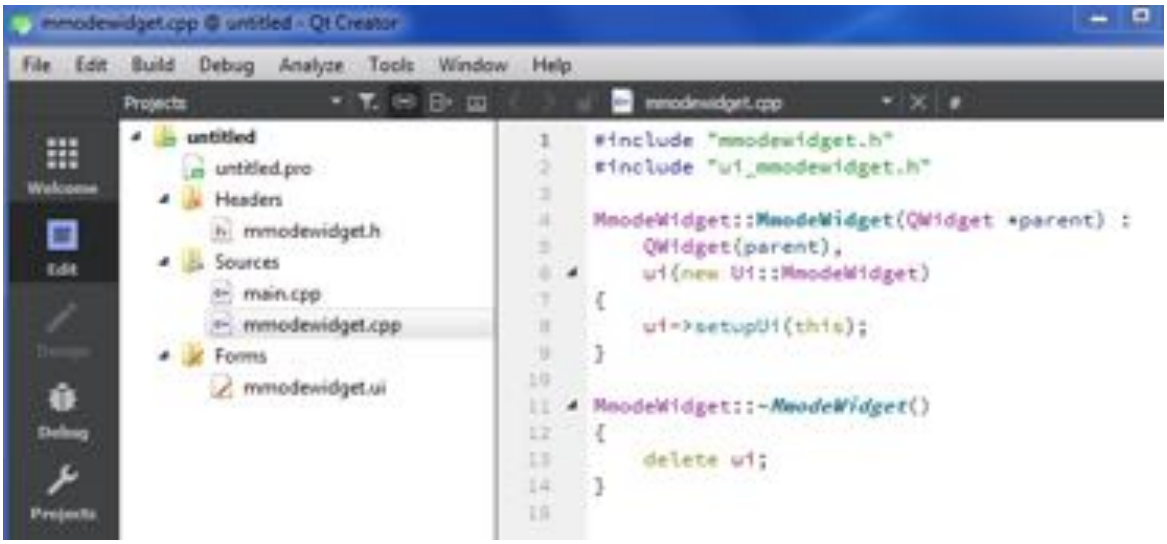
(d)


Qt Widgets Application
Location
Kits
Details
Summary
Class Information
Specify basic information about the classes for which you want to generate skeleton source code files.
Class name:
Mmodewidget
Base class:
QWidget
Header file:
mmodewidget.h
Source file:
mmodewidget.cpp
Generate form:
Form file:
mmodewidget.ui
Next
Cancel

(e)


Qt Widgets Application
Location
Kits
Details
Summary
Project Management
Add as a subproject to project:
<None>
Add to version control:
<None>
Configure...
Files to be added in
C:\Users\Fabiola\Documents\untitled:
main.cpp
mmodewidget.cpp
mmodewidget.h
mmodewidget.ui
untitled.pro
Finish
Cancel

(f)

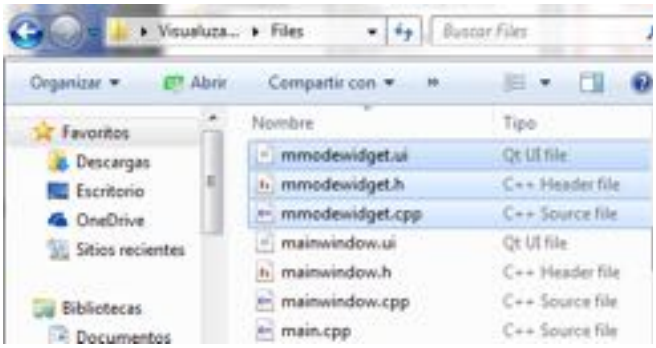


(g)

Figura E.1. Creación de la ventana del modo M con Qt Creator. (a) Creación de un nuevo proyecto. (b) Selección del tipo de aplicación. (c) Nombre y localización del proyecto. (d) Selección del Kit *Desktop*. (e) Nombre de la clase y de sus archivos correspondientes. (f) Resumen de las características generadas. (g) Contenido del proyecto hecho por Qt.

Para generar una nueva ventana se selecciona la opción *New File or Project* en Qt Creator con la opción de de una plantilla de proyectos de Aplicación de tipo *Qt Widgets Aplication* con el Kit *Desktop*. El nombre y localización del proyecto no son importantes mas si el la clase, la cual pertenece a *Qwidget* y genera los archivos con extensión .h, .cpp y .ui con su mismo nombre.

Para el modo M, a la clase creada se le da el nombre de *MmodeWidget* y sus archivos correspondientes son *mmodewidget.h, mmodewidget.cpp* y *mmodewidget.ui.*



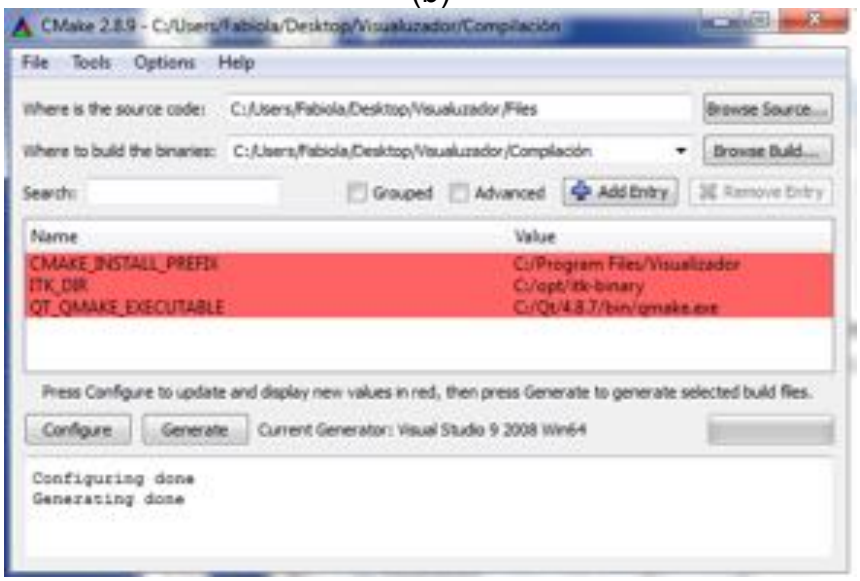
(a)

CMakeLists.txt: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
# Find QT
FIND_PACKAGE(Qt4 REQUIRED)
INCLUDE( ${QT_USE_FILE} )

INCLUDE_DIRECTORIES(
  ${CMAKE_CURRENT_BINARY_DIR}
  ${CMAKE_CURRENT_SOURCE_DIR}
)

SET(AppSrcs main.cpp mainwindow.cpp QVTKImagewidgetCommand.cpp vtkTracerInteractorStyle.cpp Openvol.cpp Open4D.cpp mmodewidget.cpp)

SET(AppHeaders mainwindow.h QVTKImagewidgetCommand.h vtkTracerInteractorStyle.h Openvol.h Open4D.h mmodewidget.h)

SET(AppUI mainwindow.ui mmodewidget.ui)

# for generate qt aditional files
QT4_WRAP_UI(UISrcs ${AppUI})
QT4_WRAP_CPP(MOCSrcs ${AppHeaders} )

SOURCE_GROUP("Resources" FILES
  ${AppUI}
)
```

(b)

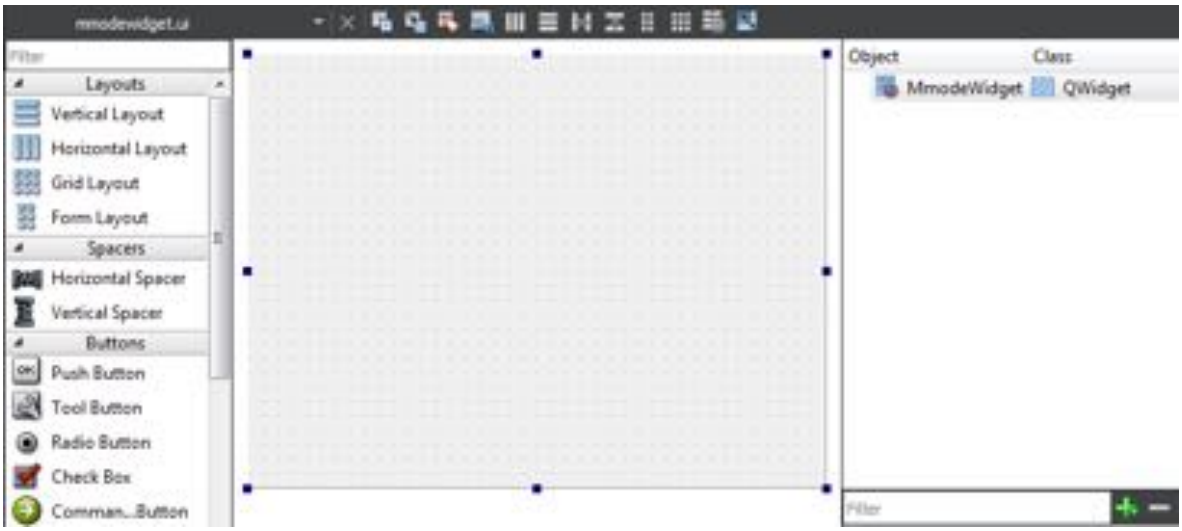


(c)

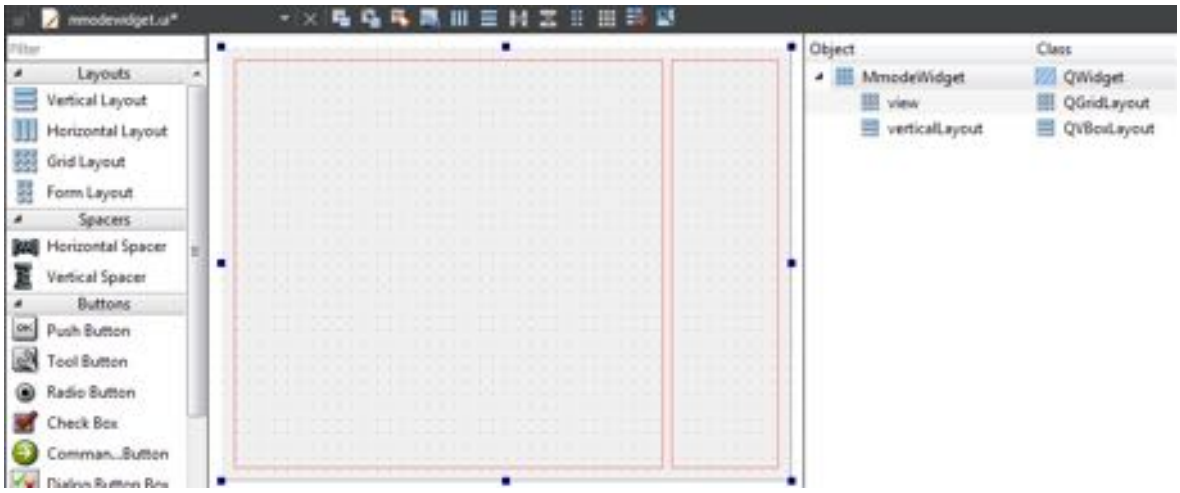
Figura E.2. Regeneración del proyecto de Visualización para incluir a la ventana de mono M. (a) Incorporación de los archivos *mmodewidget.h, mmodewidget.cpp* y *ui_mmodewidget.h* en la carpeta de los programas del sistema. (b) Inclusión de los archivos añadidos en (a) dentro de *CMakeList.txt*. (c) Regeneración del sistema con CMake.

Los archivos mencionados anteriormente son extraídos de la carpeta del proyecto y colocados en la que contiene los archivos de la interfaz de visualización. Lo que queda del archivo de Qt puede eliminarse. En *CMakeList.txt* se agregan los nombres de los nuevos archivos y se vuelve a generar el proyecto con CMake para que sean incluidos. El encabezado *mainwindow.h* se debe incluir en *mmodewidget.h* y, *mmodewidget.h* y *ui_mmodewidget.h* en *mainwindow.cpp*. Con lo anterior ya hecho, se compila el proyecto con Visual Studio. En la Figura E.2. se muestra la inclusión de una nueva ventana en la interfaz principal.

La interfaz creada se desplegar en Qt Creator y se muestra como un cuadro gris con nada más que puntos referencia, aquí es donde diseña su contenido. Antes de poner cualquier interactor, se agrega uno o más Layouts para distribuir de forma ordenada los elementos depositados dentro de ellos, puede ser distribuidos horizontal y/o verticalmente. Esto se puede observar en la Figura E.3.



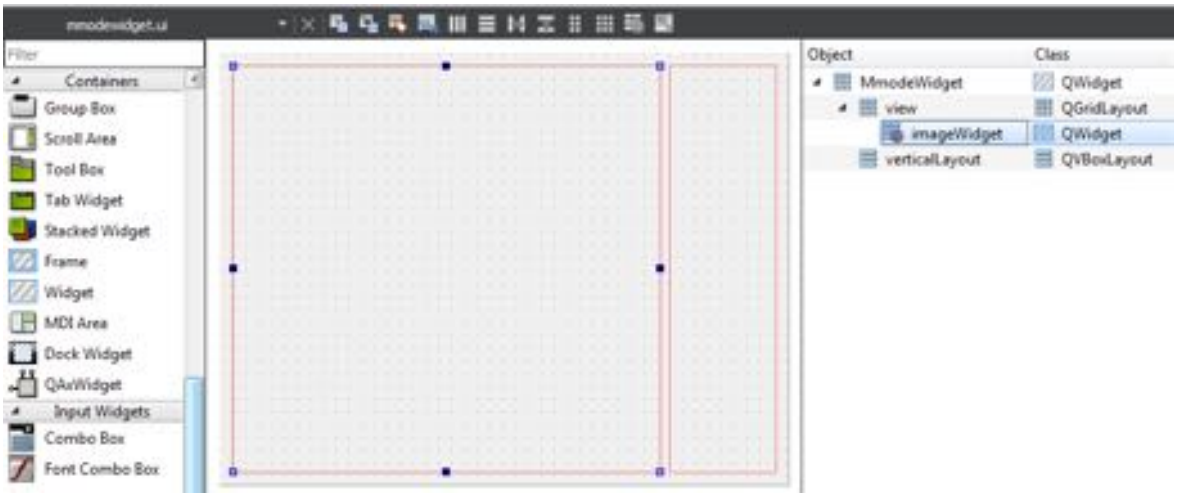
(a)



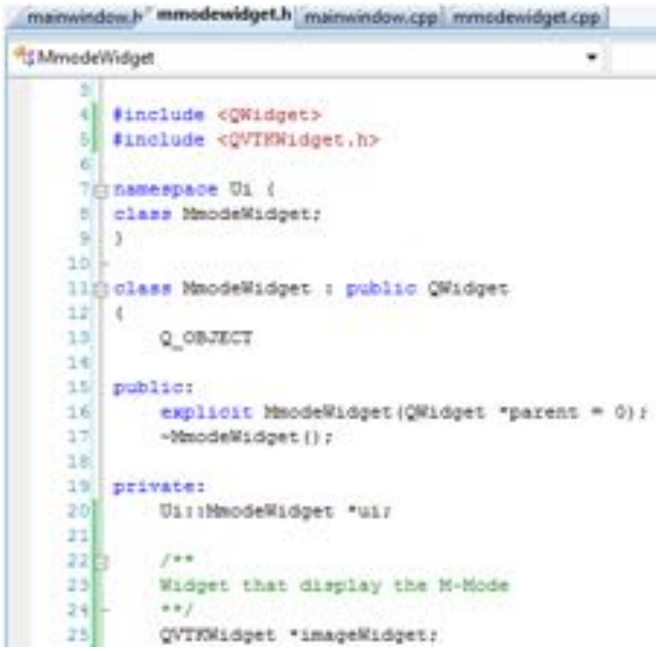
(b)

Figura E.3. Diseño de la distribución en la interfaz del modo M. (a) Ventana básica inicial. (b) Ventana con Layouts.

Los objetos de tipo *QWidget* son las ventanas donde se despliega el renderizado pero como se esta empleando VTK para ello se debe agrega el método *QVTKWidget* en el archivo .h de la clase y definir la vista como un objeto de *QVTKWidget* con el mismo nombre que se le asignó al crearlo. Para este caso se define la vista como *QVTKWidget *imageWidget*. Esta modificación se ve en la Figura E.4.



(a)



(b)

Figura E.4. Objeto *imageWidget* de la clase *Qwidget* pasa a pertenecer a la clase *QVTKWidget.* (a) Objeto *imageWidget* añadido en la interfaz. (b) Definición de *imageWidget* como *QVTKWidget.*

Referencias

[1] E. Moya-Albor, E*stimación de Flujo Óptico por medio de la transformada de Hermite* (Tesis doctoral), Universidad Nacional Autónoma de México, Ciudad de México, México, 2013.

[2] L. P. Vargas-Quintero, *Segmentación y Estimación de Movimiento en Ecocardiografía Fetal usando un Modelo de Apariencia Activa Multi-textura y la Transformada de Hermite Rotada* (Tesis doctoral). Universidad Nacional Autónoma de México, Ciudad de México, México, 2016.

[3] M. Hashimoto y J. Sklansky, “Multiple-Order Derivates for Detecting Local Image Characteristics,” *Computer Vision, Graphics, and Image Processing*, vol.39, pp. 28-55, 1987.

[4] A. Bruhn y J. Weickert, “Lucas/Kanade Meets Horn/Schunck: Combining Local and Global Optic Flow Methods,” *International Journal of Computer Vision*, vol. 61 (3), pp. 211-231, 2005.

[5] Insight Segmentation and Registration Toolkit ITK, 2016. URL https://itk.org/.

[6] Virtualization toolkit VTK, 2016. URL https://www.vtk.org/.

[7] D. Ortega y S. Seguel, "Historia del ultrasonido: el caso chileno," *Revista Chilena Radiología, v*ol. 10 (2), pp.89-92, 2004.

[8] C. Pineda, M. Macías y A. Bernal, “Principios físicos básicos del ultrasonido,” *Investigación en Discapacidad,* vol. 1, pp.25-34, 2012.

[9] W. F. Armstrong y T. Ryan, *Feigenbaum's Echocardiography,* 7th Edition, Filadelfia, Pensilvania: Liippcott Williams & Wilkins, 2010, pp. 2, 21, 22, 26.

[10] J. Fineman, R. Clyman, Fetal Cardiovascular Physiology, en: R. Creasy, R. Resnik, J. Iams, C. Lockwood y T. Moore, *Creasy & Resnik´s Maternal-Fetal Medicine: Principles and Practice*, 6th Edition, Filadelfia: Elsevier, 2009.

[11] M. Sklansky, “Fetal Cardiac Malformations and Arrhythmias,” en *Creasy & Resnik´s Maternal-Fetal Medicine: Principles and Practice,* 6th Edition, R. Creasy, R. Resnik, J. Iams, C. Lockwood y T. Moore, Filadelfia :Elsevier, 2009, pp. 308, 307.

[12] F. A. Manning, Imaging in the Diagnosis of Fetal Anomalies, en: R. Creasy, R. Resnik, J. Iams, C. Lockwood and T. Moore. (Ed.), *Creasy & Resnik´s Maternal-Fetal Medicine: Principles and Practice*, 6th Edition. Filadelfia: Elsevier, 2009.

[13] J. B. Martens, “The Hermite Transform-Theory,” *IEEE Transactions on Acoustics, Speech and Signal Processing,* vol. 38 (9), pp. 1595-1606, 1990.

[14] W. T. Freeman y E. H. Adelson, “The design and use of steerable filters,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13 (9), pp. 891-906, 1991.

[15] J. L. Barron, D. J. Fleet y S. S. Beauchemin, “Performance of Optical Flow Techniques,” *International Journal of Computer Vision*, vol. 12 (1), pp. 43-77, 1994.

[16] C. L. Fennema y W. B. Thompson, “Velocity Determination in Scenes Containing Several Moving Objects,” C*omputer Graphics and Image Processing* vol. 9 (4), pp. 301-315, 1979.

[17] B. Horn y B. Schunck, "Determining optical flow," *Artificial Intelligence, v*ol. 17, pp. 185–203, 1981.

[18] H.-H. Nagel, “Constraints for the Estimation of Displacement Vector Fields from Image Sequences,” *Proceedings of the Eighth International Joint Conference on Articial Intelligence*, vol. 2, pp. 945-951, 1983.

[19] H.-H. Nagel, “On the estimation of optical flow: Relations between dierent approaches and some new results,” *Artifical Intelligence* vol. 33 (3), pp. 299-324, 1987.

[20] S. Uras, F. Girosi, A. Verrri y V. Torre, “A Computational Approach to Motion Perception,” *Biological Cybernetics* vol. 60, pp. 79-87, 1988.

[21] B. Lucas y T. Kanade, “An iterative image registration technique with an application to stereo vision,” *Proc. Seventh International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

[22] C. Schnörr, “Determining optical flow for irregular domains by minimizing quadratic functionals of a certain class,” *International Journal of Computer Vision*, vol. 6 (1), pp. 25-38, 1991.

[23] B. Galvin, B. McCane, K. Novins, D. Mason y S. Mills, “Recovering motion fields: An analysis of eight optical flow algorithms,” en: *Proceedings 1998 British Machine Vision Conference*, Southampton, Inglaterra, pp.195-204, 1998.

[24] P. Charbonnier, L. Blanc-F´eraud, G. Aubert y M. Barlaud, “Two deterministic half-quadratic regularization algorithms for computed imaging”. en *Proceedings 1994 IEEE International Conference on Image Processing*, vol. 2. IEEE Computer Society Press: Austin TX, pp. 168-172, 1994.

[25] P. Anandan, “A computational framework and an algorithm for the measurement of visual motion,” *International Journal of Computer Vision*, vol. 2, pp. 283-310, 1989.

[26] M. J Black y P. Anandan, “The Robust Estimation of Multiple Motions: Parametric and Piecewise Smooth Flow Fields,” C*omputer Vision and Image Understanding,* vol .63 (1), pp. 75-104, 1996.

[27] E. Mémin, y P. Pérez, “A multigrid approach for hierarchical motion estimation,” en: *Proc. 6th International Conference on Computer Vision*, Bombay, India, pp. 933-938, 1998.

[28] E. Mémin, y P. Pérez, “Hierarchical estimation and segmentation of dense motion fields,” *International Journal of Computer Vision*, vol. 46 (2), pp. 129-155, 2002.

[29] N. Papenberg, A. Bruhn, T. Brox, S. Didas, y J. Weickert, “Highly accurate optic flow computation with theoretically justified warping,” *International Journal of Computer Vision,* vol. 67, pp.141-158, 2006.

[30] H.-H. Nagel y W. Enkelmann, “An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences,” *Pattern Analysis and Machine Intelligence,* vol. 8, pp. 565–593, 1986.

[31] B. Escalante-Ramirez, J. Silvan-Cardenas y H. Yuen-Zhuo, H, "Optic flow estimation using the Hermite transform," en: *Proceedings of the SPIE in Applications of Digital Image Processing XXVII,* vol. 5558, pp. 632-643, 2004.

[32] H. Liu, T. H. Hong, M. Hermn y R. Chellappa, "A general motion model and spatio-temporal filters for computing optical flow," *International Journal of Computer Vision,* vol. 22, pp. 141-172, 1997.

[33] A. Bruhn, J. Weickert y C. Schnörr, “Combining the advantages of local and global optic flow methods,” en: *Proceedings of the Twentyfourth DAGM Symposiumon Pattern Recognition*, pp. 454-462, 2002.

[34] A. B. Estudillo-Romero, B. Escalante-Ramrez, “Rotation-invariant texture features from the steered Hermite transform,” *Pattern Recognition Letters,* vol. 32 (16), pp. 2150-2162, 2011.

[35] G. Aubert, R. Deriche y P. Kornprobst, "Computing Optical Flow via Variational Techniques," *SIAM J Appl Math,* vol. 60, pp. 156-182, 1999.

[36] N. Carranza-Herrezuelo, A. Bajo, F. Sroubek, C. Santamarta, G. Cristobal, A. Santos, M. Ledesma-Carbayo, "Motion estimation of tagged cardiac magnetic resonance images using variational techniques," *Computerized Medical Imaging and Graphics,* vol. 34, pp. 514–522, 2010.

[37] Introducing Visual Studio, 2016. URL https://msdn.microsoft.com/en-us/library/fx6bk1f4(v=vs.90).aspx.

[38] Visual Studio Guided Tour, 2016. URL https://msdn.microsoft.com/en-us/library/bb514232(v=vs.90).aspx.

[39] Qt, 2016. URL https://www.qt.io/.

[40] Cmake, 2016. URL https://cmake.org/.

[41] H. J. Johnson, M. M. McCormick y L. Ibáñez, Consorcio Insight Software, *The ITK Software Guide, Book 1: Introduction and Development Guidelines, Fourth Edition Updated for ITK version 4.10., 2016.* URL https://itk.org/ItkSoftwareGuide.pdf.

[42] L. S.Avila, U. Ayachit, S. Barré, J. Baumes, F. Bertel, R. Blue, D. Cole, D. DeMarle, B. Geveci, W. A. Hoffman, B. King, K. Krishnan, C. C. Law, K. M. Martin, W. McLendon, P. Pebay, N. Russell, W. J. Schroeder, T. Shead, J. Shepherd, A. Wilson, B. Wylie. The VTK User’s Guide, 11th Edition, 2010. URL https://www.kitware.com/products/books/VTKUsersGuide.pdf.

[43] A. Hadjidimos, “Successive overrelaxation (SOR) and related methods,” *Journal of Computation and Applied Mathematics*, vol. 123, pp. 177-199, 2000.

[44] W. Kahan, *Gauss–Seidel methods of solving large systems of linear equations* (Tesis doctoral), Universidad de Toronto, Toronto, Canada, 1958.

[45] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black y R. Szeliski, “A Database and Evaluation Methodology for Optical Flow”, *International Journal of Computer Vision*, vol. 92 (1), pp. 1-31, 2011.

[46] Setting up Qt 4.8.5 for Visual Studio 2008 x86 and x64, 2016. URL https://mlrecitals.wordpress.com/2014/04/10/qt-4-8-5-for-visual-studio-2008-x86-and-x64/.

[47] Configuration Options, Qt Documentation, 2016. URL http://doc.qt.io/qt-4.8/configure-options.html.

[48] Avertencia QT_LARGEFILE_SUPPORT, 2016. URL http://lists.qt-project.org/pipermail/interest/2014-June/012635.html.