Universidad de Granada

DEPARTAMENTO DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

DEPARTAMENTO DE ELECTRÓNICA Y TECNOLOGÍA DE COMPUTADORES

Implementación en hardware reconfigurable de un modelo de flujo óptico robusto

TESIS DOCTORAL

Guillermo Botella Juan

Granada, 2007

Editor: Editorial de la Universidad de Granada
Autor: Guillermo Botella Juan
D.L.: Gr. 1358 - 2007
ISBN: 978-84-338-4381-4

Profesor D. Eduardo Ros Vidal, Profesor Titular de Universidad, D. Manuel Rodríguez Álvarez, Profesor Titular de Escuela Universitaria ambos del Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada y D. Antonio García Ríos, Profesor Titular del Departamento de Electrónica y Tecnología de Computadores de la Universidad de Granada,

CERTIFICAN:

Que la memoria titulada **"Implementación en hardware reconfigurable de un modelo de flujo óptico robusto"**, ha sido realizada por D. Guillermo Botella Juan bajo nuestra dirección en los departamentos de :Arquitectura y Tecnología de Computadores y Electrónica y Tecnología de Computadores, ambos de la Universidad de Granada para optar al grado de Doctor en CC. Físicas y a la mención especial de Doctor Europeo en CC. Físicas por la Universidad de Granada.

Granada, a 25 de Mayo de 2007

Fdo: Dr.Eduardo Ros Vidal	Fdo: Dr. Manuel Rodríguez Álvarez	Fdo: Dr Antonio García Ríos
Director de la Tesis	Director de la Tesis	Director de la Tesis

Universidad de Granada

DEPARTAMENTO DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

DEPARTAMENTO DE ELECTRÓNICA Y TECNOLOGÍA DE COMPUTADORES

Implementación en hardware reconfigurable de un modelo de flujo óptico robusto

Memoria presentada por

Guillermo Botella Juan

Para optar al grado de

DOCTOR EUROPEO EN CIENCIAS FÍSICAS

POR LA UNIVERSIDAD DE GRANADA

Fdo.: Guillermo Botella Juan

A la memoria de mi tío Jorge.

A mis padres.

"Sabes que tu tiempo es limitado,
entonces, no lo malgastes viviendo la
vida de otra persona. No te dejes
atrapar por el dogma de los demás, es
decir, no sigas al pie de la letra los
pensamientos y las conclusiones de otra
gente. No dejes que el ruido de las
opiniones de los otros, ahogue tu propia
voz interior. Y lo más importante de todo,
ten siempre el valor de seguir lo que dicta
tu corazón y tu intuición. Éstos de algún modo,
ya saben lo que realmente quieres llegar
a ser en la vida.
Todo lo demás, es absolutamente secundario."

(Steve Jobs (1955), fundó Apple Computer en 1976.
Actualmente preside la productora de películas de animación Pixar)

AGRADECIMIENTOS

Deseo expresar en primer lugar, mis más sentidos agradecimientos a mis 3 directores, los profesores de la Universidad de Granada: el Dr. Eduardo Ros Vidal, el Dr. *Manolo* Rodríguez Álvarez, el Dr. Antonio García Ríos, por su dirección, asesoramiento en todo momento, su apoyo, comprensión, paciencia y la estrecha colaboración mantenida a lo largo de estos seis interminables años, en definitiva, por enseñarme a investigar, y al catedrático D. Francisco José Pelayo Valle y a la Dra. Begoña del Pino Prieto a quienes hago extensivos todos los agradecimientos anteriores.

A mis amigos y compañeros de los departamentos de Arquitectura y Tecnología de Computadores y Electrónica y Tecnología de Computadores de la Universidad de Granada, tanto en la Facultad de Ciencias como en la Escuela Técnica Superior de Ingeniería Informática y de Telecomunicaciones, por todo el apoyo y afecto recibido, resultando todos los años pasados en Granada como *inolvidables*.

Al profesor Dr. Alan Johnston de la *University College of London*, al Dr. Jason Dale, Dr. Sovira Tan, Dr. Glynn Cowe, Dra. Szonia Durant, Dr. Derek Arnold y en especial al Dr. Chris Müller con quién tan buenas migas hice en tierras sajonas, seguro que tal y como era su deseo, estará viajando por cualquier parte del globo…y a Mr. John Drapper, un verdadero *crack*…a todos ellos, por su hospitalidad, amistad y su ayuda en los más variopintos asuntos científicos, técnicos, burocráticos...

A mis compañeros y amigos del Departamento de Arquitectura de Computadores y Automática de la Universidad Complutense de Madrid, por su gran ayuda, singularmente a la catedrático Dª Milagros Fernández Centeno quién me brindó una acogida especialmente calurosa, cuando me incorporé a este departamento hace unos 3 años.

Agradezco en el plano personal a mis padres y hermanos, sin su arrojo incondicional, confianza en mí y permanente ánimo, este trabajo no habría sido posible nunca.

Quisiera tener un recuerdo especial a la memoria de María y de mi tío Jorge, quienes siempre pelearon hasta el final, ellos me enseñaron que incluso a pesar de estar en las peores circunstancias, siempre existen unos valores más allá del cada vez más frecuente oportunismo.

También agradezco, finalmente, a todos mis amigos, repartidos por tantos y tan variados sitios, cuya lista sería muy extensa… al escribir estas líneas, estáis todos en mi memoria, ya que de alguna manera, este trabajo también es vuestro.

Es obligado destacar que esta investigación se ha financiado gracias a 3 proyectos, siendo 2 de ellos europeos: IST-2001-32114, QLK5-CT-1999-50523, y uno nacional, correspondiente al proyecto del Ministerio de Educación y Ciencia DPI 2004-07032. En el capítulo VI de esta memoria se explica el alcance y la relación de cada uno de ellos con este trabajo.

Capital de las Españas, *Mayrīt*/ ciudad de la Alhambra, *Ilíberis*, mayo de 2007.

ÍNDICE

GLOSARIO DE TÉRMINOS

A continuación se detalla un glosario de abreviaturas que aparecen a lo largo de esta memoria, a fin de facilitar la lectura de la misma.

ALU Arithmetic Logic Unit, unidad aritmético lógica.

ASIC Application Specific Integrated Circuits, circuito integrado de aplicación específica.

CCD Charge-Coupled Device, dispositivo de cargas (eléctricas) interconectadas.

CLB Cell Logic Block, bloque de celda lógica.

CMOS Complementary Metal-Oxide Semiconductor, proceso metal-óxido semiconductor complementario.

CPU Central Processing Unit, unidad central de proceso.

DK Design Kit, kit de diseño perteneciente a Celoxica.

DMA Direct Memory Access, acceso directo a memoria.

DSP Digital Signal Processor, procesador digital de señal.

EDIF Electronic Design Interchange Format, formato de intercambio de diseño electrónico.

FFT Fast Fourier Transform, transformada rápida de Fourier.

FIR Finite Impulse Response, respuesta impulsiva finita.

FMRI Functional magnetic resonance imaging, imagen de resonancia magnética funcional.

FPGA Field Programable Gate Array, matriz de puertas programables por campos.

GELT **(**Función) Gaussiana en el logaritmo del tiempo.

IEEE Institute of Electrical and Electronics Engineers, instituto de ingenieros eléctricos y electrónicos (es la mayor asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas).

IIR Infinite Impulse Response, respuesta impulsiva infinita**.**

ICA Independient Component Analisis, análisis de componentes independientes.

IP Intelectual Property, propiedad intelectual.

JTAG Join Test Action Group, grupo para acciones coordinadas sobre test.

MAC Multiply and Accumulate, multiplicar y acumular.

McGM Modelo Multicanal de Gradiente, modelo psicofísico fuertemente bioinspirado de detección del movimiento creado en la UCL por Alan Johnston & Chris Benton, objeto de la implementación de este trabajo de investigación.

MIMD Multiple Instructions Multiple Data, múltiples instrucciones para múltiples datos. (según la taxonomía de Flynn).

MMX MultiMedia eXtensions, conjunto de instrucciones multimedia.

LGN Latheral Geniculate Nucleous, núcleo geniculado lateral.

LOG Laplacian Of the Gaussian, operador laplaciana de gausiana.

LUT Look-Up Table, tabla de consulta.

PCI Peripheral Component Interconnet, estándar para conexión de componentes.

PET Positron Emission Tomography, tomografía de emisión de positrones**.**

SOC System On a Chip, sistema (embebido) en el chip.

RAM Random Access Memory, memoria de acceso aleatorio**.**

MCE Motion Constraint Equation, ecuación de constricción de movimiento.

PAL Phase Alternating Line, linea alternada de fase, (hace referencia a la forma de transmisión de televisión analógica más usada en el mundo).

SNR Signal Noise Ratio**,** relación señal ruido.

RGB Red Green Blue, Decomposición de colores en rojo, verde y azul.

ROI Region of Interest, región de interés.

RTL Register Transfer Level, nivel de transferencia de registros.

SIMD Single Instruction Multiple Data, un sola intrucción y múltiples datos (según la taxonomía de Flynn).

SSE Streaming SIMD Extensions, extensiones SIMD.

SRAM Static Random Access Memory, memoria de acceso aleatorio estática.

V1 Primary Visual Cortex, corteza visual primaria., es la zona visual del córtex mejor y más estudiada de todas, estando especializada en reconocimiento de patrones.

V5-MT Visual area (V5), Visual area (Middle temporal), region del córtex visual que juega un papel fundamental en varios campos como el procesamiento de movimiento, la transformación de señales de movimiento en percepciones y la conducción de algunos movimientos oculares.

VHDL Very high-speed integrated circuit Hardware Description Lenguaje, lenguaje para descripción de circuitos integrados de muy alta velocidad.

VLSI Very Large Scale Integration, escala muy grande de integración..

VLIW Very Long Instruction Word, palabra de instrucción muy larga.

BREVE INTRODUCCIÓN

El tratamiento y procesado de señal es un campo que tiene ya una larga y rica historia, que se relaciona con un inmenso conjunto de disciplinas y que ha cobrado una gran importancia dentro de ellas. Dentro de este campo, la visión artificial es especialmente relevante debido a la gran cantidad de aplicaciones y repercusión de éstas en el mundo que nos rodea.

Perteneciente a la visión artificial, encontramos la detección de movimiento, un tema que a pesar de llevar investigándose unos 40 años está prácticamente en un estado incipiente, careciendo los sistemas existentes más avanzados de características que atesoran los mamíferos, además de que los recursos computaciones requeridos son ingentes y a menudo inabordables cuando se requiere tiempo real.

El objetivo del presente trabajo de investigación es el estudio e implementación en *hardware* configurable de un modelo de flujo óptico robusto.

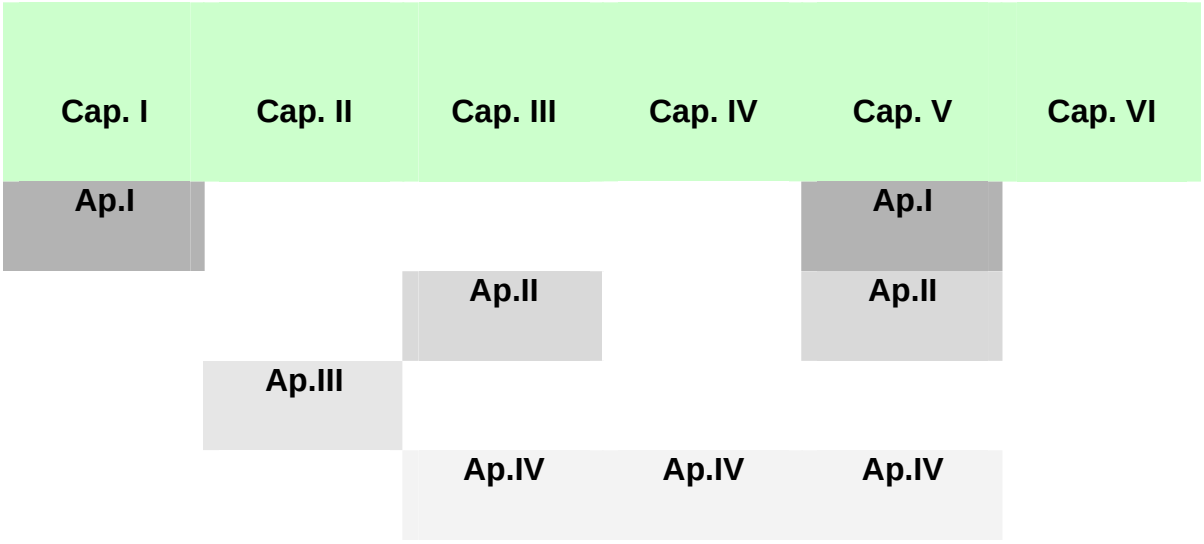
El *hardware* configurable tiene una serie de características propias, como bajo coste, prototipado rápido, capacidad de modificación del diseño *a posteriori*, que lo hacen apropiado frente a otras alternativas. Respecto al modelo de flujo óptico robusto (McGM), ha sido diseñado desde hace unos 20 años por Alan Johston y Chris Benton [JOH95] mediante continuas versiones y mejoras, añadiendo este trabajo algunas aportaciones propias. Se ha escogido este modelo por su robustez y fuerte bioinspiración, incorporando capacidades inherentes a los humanos, dando cuenta, por ejemplo, de algunas ilusiones ópticas sólo percibidas por éstos [AND03][JOH94]. La gran contrapartida a todas estas cualidades, es su elevada complejidad computacional, no existiendo hasta la presente ninguna publicación en *hardware* configurable cuya complejidad sea ni siquiera un orden de magnitud inferior a la requerida por el sistema completo.

La presente memoria se ha organizado en 6 capítulos y 4 apéndices de apoyo, que creemos, servirán para aclarar conceptos que pueden quedar difusos debido a su naturaleza tangencial con este trabajo.

En el capítulo I se hablará de sistemas bioinspirados, el tipo de problemas al que nos enfrentamos y la tecnología al efecto que usaremos. En el capítulo II revisaremos el estado del arte de algunos de los métodos existentes para la estimación de flujo óptico. En el capítulo III se profundizará en la presente implementación *software* del modelo McGM.. Llegado a este punto, puede ser útil consultar: el apéndice I referido a sistemas configurables en general, el apéndice II donde se explica el modelo utilizado, mediante un diagrama de flujo jerárquico diseñado al efecto y se justifican su robustez y principales características matemáticamente, además del apéndice III, donde se expone su inspiración biológica y la naturaleza de los operadores usados.

Seguimos con el capítulo IV, donde construiremos una plataforma para poner a prueba el modelo y analizar la exactitud frente a la eficiencia (puente entre *hardware* y *software*). En el capítulo V diseñaremos varias arquitecturas especializadas en la FPGA usada, describiendo cada una de las etapas de diseño, estructura de datos y comunicación y evaluación de resultados. El capítulo VI se centrará en una discusión y conclusión global de este trabajo, remarcando las principales aportaciones y líneas futuras.

El apéndice IV trata de ilustrar muy brevemente el resultado de aplicar el sistema diseñado en sus diferentes versiones, *software*, *semihardware* y *hardware* a secuencias reales. Por último, la bibliografía se ha escrito lo más exhaustiva y completa posible, siguiendo el formalismo de notación usual.



Propuesta para la consulta de los apéndices, en relación con el tema de interés en cada capítulo

BRIEF INTRODUCTION

Signal processing is a field of rich and long history, which is related to a big set of topics and has grown in importance within them. Inside of this research field, artificial vision is specially relevant due to the great quantity of its applications and their repercussion in our real world. Within artificial vision, motion detection is a topic that is in an incipient state despite of more than 40 years of research. Even the best algorithms still need a set of characteristics that are found in the mammalians, with huge computational resources required and usually not affordable when real-time is required.

The aim of this research work is a reconfigurable *hardware* implementation of a robust optical flow model. Reconfigurable hardware possesses a set of characteristics that includes low cost, rapid prototyping, possibility of changing the final design, which are valuable features when compared to other alternatives.

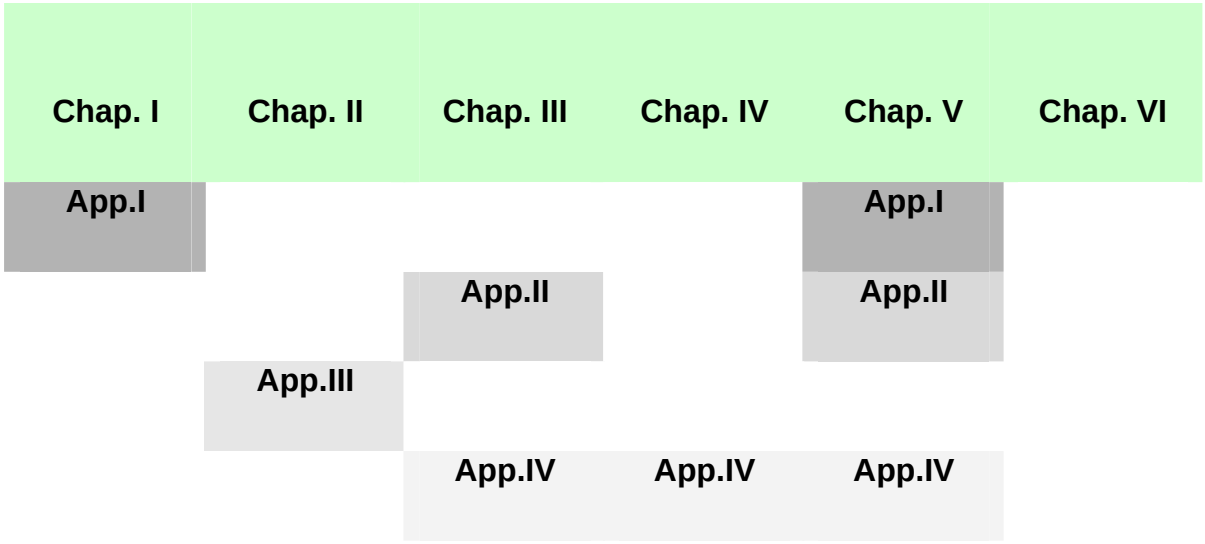
The robust optical flow model (McGM) has been under development by Johnston *et al* [JOH95] for 20 years, with the present research work adding some new contributions. This model has been chosen because of its robustness and strong bioinspiration, as it will be discussed over the next chapters. An example of this is the ability to justify some optical illusions [AND03][JOH94] only perceived by humans. The drawback of this system is its huge computational complexity, not appearing until now any publication about optical flow using reconfigurable *hardware* whose complexity was at least one order of magnitude less than this required by the complete system.

This work is organized in six chapters and four support appendices that will clarify some concepts which could be diffuse due to their tangential nature to this work.

In the first chapter we will discuss about bioinspired systems, the kind of problem to be treated and the available technology. In chapter II we revise the motion estimation state of

art, while in chapter III our *software* implementation of McGM model is explained. At this point, it can be useful to check appendix I, which is related to general reconfigurable systems, while appendix II explains the model with a hierarchical flow chart and accounts for its characteristics. Appendix III deals with the bioinspiration attributes of the model in terms of its filters and parameters.

We follow with chapter IV, where we build a platform that will test the model and analyse its accuracy *versus* efficiency (this being a bridge between *hardware* and *software*). In chapter V we will design some specialized architectures for the FPGA used, describing each one of the design stages, data structure and communication, metrics and result evaluation. In chapter VI, a global discussion will be made, remarking the main contributions and future lines. Appendix IV tries to represent briefly the results of applying the system designed in its different versions, pure s*oftware*, *semihardware* and *hardware*, to real sequences. Finally, the used bibliography has compiled as exhaustive and complete as possible following the usual formalism.



Proposed reading scheme for linking the appendixes and the chapters in relationship with the topic field.

CAPÍTULO I

Introducción y conceptos generales

1.1. Introducción

La visión representa el sentido más primario e importante para los seres vivos que disponen de percepción visual. Permite extraer información continua sobre el entorno, reconociendo objetos a través de sus formas, colores y movimiento. Se percibe profundidad y distancia a través de texturas, sombras, estereópsis, color, además, mientras se observa el movimiento de todo lo que rodea al observador, se realizan cálculos exactos en tiempo real muchas veces inconscientemente, se predicen eventos futuros (tiempo hasta un impacto, por ejemplo) y se evitan situaciones potencialmente peligrosas.

Sin embargo, la percepción visual tiene un gran coste asociado, siendo un ejemplo el sistema del propio ser humano, en el que el 40 % del área cortical del cerebro se dedica a procesar información visual y el 70% de la información de entrada al cerebro proviene de los ojos. Este hecho resalta la importancia y gran cantidad de recursos que se necesitan para procesar la información visual, incluido el cerebro, que tiene que trabajar arduamente para estructurar y tratar toda esta información.

Intentar procesar el volumen de toda esta información puesta en juego y comprender su complejidad, ha sido la principal motivación hasta nuestros días de infinidad de investigadores. Aunque se ha avanzado mucho en visión estereoscópica, reconocimiento de patrones, seguimiento, navegación, etc., muy pocos modelos todavía se acercan a la robustez y eficiencia de los sistemas biológicos.

A medida que la tecnología progresa, la potencia de cálculo para sistemas en tiempo real ha mejorado significativamente. Debido a la complejidad que conlleva el proceso, se necesitan soluciones más sofisticadas y delicadas que la mera computación masiva.

A modo de ejemplo, si se realiza un cálculo sencillo, para una imagen con una matriz bidimensional de 1000×1000 componentes, cada uno definido por 8 bits, existen $255^{1,000,000}$ imágenes posibles, aunque la gran mayoría de ellas sean bastante improbables en entornos naturales [HAT98]. Si se construye un sistema de visión que trabaje a 25 Hz (la tasa típica de una video cámara estándar), con la resolución del ejemplo anterior, se tendrían que procesar 25 millones de puntos por segundo. Suponiendo que se pudiera hacer este procesamiento en unas mil operaciones aritméticas elementales, se tendrían que realizar 25.000 millones de operaciones por segundo, haciendo notar con el presente ejemplo, que se está tratando con grandes cantidades de datos y operaciones. A pesar de todas estas dificultades, el ser humano procesa esta enorme tarea de forma continua, robusta, eficiente y sin gran esfuerzo. Este hecho supone un enorme campo para investigar, recogiendo la inspiración del sistema visual del mamífero en general y del humano en particular.

El mundo en el que se está inmerso, presenta un cambio dinámico y continuo, obteniendo a través del movimiento de la escena y demás objetos tentativas y posibles eventos futuros. Explorando de forma activa e interactuando con movimientos del ojo, cabeza y demás partes del cuerpo, se puede usar el movimiento visual para conseguir más información como profundidad, forma, etc. Teniendo en cuenta todas estas capacidades, resulta inmediato que se desee simular y modelar las habilidades del sistema visual humano para poder diseñar sistemas artificiales que procesen y respondan al movimiento.

La principal motivación de este trabajo de investigación es el estudio de viabilidad e implementación eficiente en *hardware* reconfigurable de un algoritmo de estimación de imagen en movimiento (flujo óptico) que sea bioinspirado, es decir, que emule el comportamiento del sistema visual de un ser vivo, con una serie de características que lo hagan único frente a todos los demás. Se introducirá, además, el concepto de ingeniería neuromórfica (inspiración en la naturaleza del sistema nervioso para el diseño de algoritmos y arquitecturas especializadas) y se implementará un modelo específico de percepción visual del movimiento denominado “Modelo Multicanal de Gradiente (*Multi-channel Gradient Model*: *McGM*)” [JOH98]. Se optimizará el modelo existente a través de modificaciones algorítmicas y arquitecturales, diseñando arquitecturas eficientes para las diferentes unidades del sistema, obteniendo al final de este trabajo varias implementaciones específicas en *hardware* basadas en un sistema de co-diseño funcional. Adicionalmente, se muestra que la arquitectura del modelo daría pie a una implementación completa en *hardware* reconfigurable, disponiendo de suficientes recursos para la siguiente etapa de investigación.

1.2. Flujo óptico y movimiento de imágenes

El término “flujo óptico” fue acuñado por James Gibson tras la II Guerra Mundial, cuando trabajaba en el desarrollo de pruebas para pilotos en las Fuerzas Aéreas norteamericanas [GIB47]. Su teoría básica proporcionó el punto de partida para gran parte del trabajo de visión por computador 30 años después [MAR82]. En publicaciones posteriores, Gibson define el gradiente de deformación de la imagen a lo largo del movimiento del observador, concepto que fue bautizado como “flujo óptico” o simplemente “flujo”.

Cuando se mueve un objeto respecto por ejemplo, a una cámara, su proyección bidimensional se mueve con la imagen proyectada (Figura 1.1). La proyección de los vectores de movimiento tridimensionales sobre el detector bidimensional se denomina campo aparente de velocidades o flujo de imagen. El observador no tiene acceso al campo de velocidades directamente, ya que los sensores ópticos proporcionan distribuciones de luminancia, no de velocidad y los vectores de movimiento pueden ser muy diferentes de la distribución de luminancia.

La expresión “aparente”, hace referencia a uno de los mayores problemas en visión artificial al no disponer nunca de la velocidad real, sino de un campo bidimensional denominado campo de movimiento.

Sin embargo, se puede calcular el movimiento de regiones locales de la distribución de luminancia, siendo conocido este campo de movimiento como flujo óptico, proporcionando sólo una aproximación al campo real de velocidades [VER89].

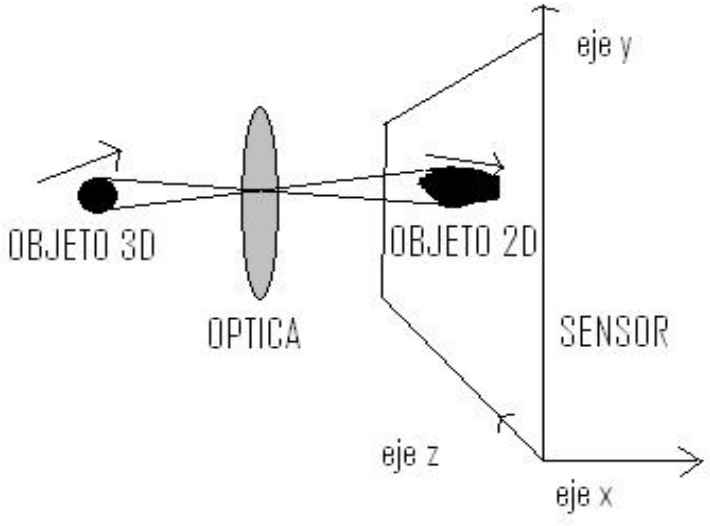


Figura 1.1. La proyección de movimiento desde el mundo 3D a la superficie del detector bidimensional.

Un ejemplo bastante ilustrativo, donde se puede apreciar que el flujo óptico es diferente al campo de velocidades (Figura 1.2), se tiene en una esfera con brillo uniforme rotando, que no produce ningún cambio en la luminancia de la imagen. El flujo óptico será nulo en todo lugar en contradicción con el campo de velocidades. La situación inversa es una escena estática con una luz en movimiento, en la que el campo de velocidades será cero en todo lugar, incluso aunque el cambio de luminancia induzca flujo óptico no nulo. Otro ejemplo se tiene en los postes rotatorios que anunciaban las barberías de antaño, siendo el campo de velocidades perpendicular al flujo.

En general, estas situaciones son atípicas y se verá cómo normalmente se puede hacer una estimación de movimiento, dentro de ciertos límites. Los algoritmos de movimiento descritos en este trabajo de investigación buscan recuperar el flujo como una aproximación del campo de velocidades proyectado, siendo la información del flujo una matriz bidimensional de vectores que pueden estar sujetos a varias interpretaciones de alto nivel. En la literatura actual existen varios usos al respecto [NAK85], [MIT96], [CAM92], siendo algunos de ellos:

- Seguimiento: la persecución de un objetivo en movimiento requiere información de su comportamiento dinámico [SMI95], [BUX96], [DAN98].
- Segmentación de la imagen: en general, los puntos dentro de un objeto se deberían mover a la misma velocidad. Los movimientos discontinuos entre objetos proporcionan

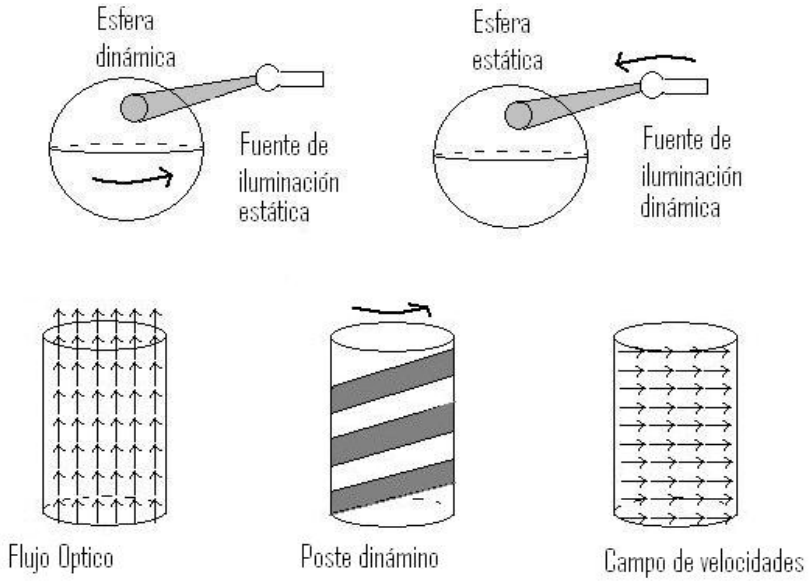


Figura 1.2. Ejemplos de diferencia entre campo de velocidades y flujo óptico.

información para la segmentación [CAM92], [SMI95].

- Detección de movimiento de la escena: como medida fundamental, el movimiento de la imagen es indicativo dentro del entorno dinámico que la rodea.
- Forma proveniente del movimiento: el hecho de que el movimiento de la imagen proporciona pistas interesantes ya fue observado hace más de un siglo por Helmholtz. [HEL62]. Existen numerosos experimentos que resaltan la capacidad del sistema visual humano de ganar información proveniente del movimiento, como los llevados a cabo por Koenderink [KOE86], Mitchie [MIT94] y Gupta [GUP95].
- Predicción: la información del movimiento se puede usar para predecir el futuro basado en el comportamiento dinámico de objetos y escena. Una aplicación particularmente interesante en robótica es el tiempo para un impacto [SCA99] y la detección de obstáculos [ENK91].
- Compresión de video: la redundancia temporal en la secuencia de imágenes puede ser aprovechada usando compensación de movimiento [ACC98], [DEF95].

La interpretación y uso del flujo óptico es un problema mal condicionado ya que, como se ha visto, está basado en proyección de imágenes en tres dimensiones sobre un detector bidimensional. Independientemente de si este detector es una cámara CCD o una retina biológica, el proceso elimina información, siendo su recuperación no trivial. Por consiguiente, la medida de flujo es un problema mal planteado en si mismo, ya que hay infinitos campos de velocidades que puedan causar los cambios observados en la distribución de luminancia, además de que, por supuesto, hay infinitos movimientos tridimensionales que

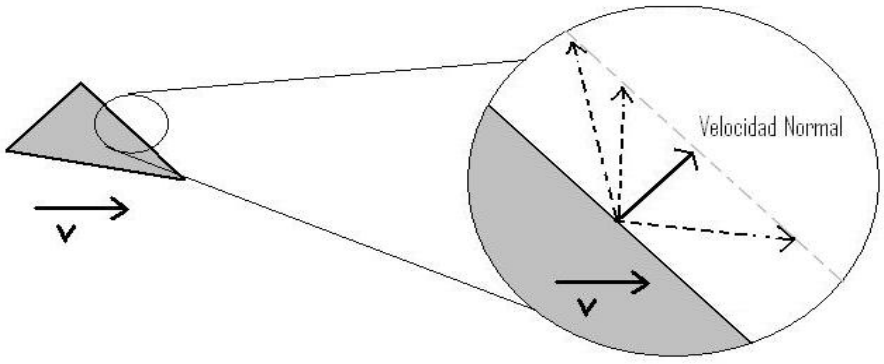


Figura 1.3. El Problema de la Apertura. Existen infinitas soluciones para este problema.

pueden generar un campo determinado de velocidades.

Es necesario entonces, tener en cuenta un número importante de consideraciones para recuperar el flujo. El llamado problema de apertura [WAL76], [HOR81], [ADE85] aparece cuando se miden las dos componentes de la velocidad bidimensional usando sólo medidas locales (Figura 1.3). Sólo se puede recuperar la componente de la velocidad normal al gradiente del borde, forzando añadir condiciones externas que usualmente requieren obtener la información de una vecindad finita espacial. Para resolver este problema, la región debe ser suficientemente extensa para conseguir una solución, como la búsqueda de una esquina, por ejemplo. Sin embargo la recolección de información a lo largo de una región incrementa la probabilidad de tomarla también sobre contornos de movimiento diferentes y consecuentemente de truncar los resultados; esto ha sido bautizado como el problema de apertura general [ONG99].

Las secuencias de imágenes usadas en visión artificial normalmente se obtienen gracias a un dispositivo CCD que expone sus sensores a una frecuencia de unos 25 Hz, esta serie de fotogramas producen una percepción de movimiento continuo. Un problema que se encuentra cuando se mide flujo óptico en imágenes reales es que si se tiene una tasa de muestreo baja los desplazamientos entre fotogramas suelen ser altos, (como se explicará convenientemente en el capítulo II), esto origina dificultades cuando se miden derivadas temporales en modelos de gradiente que se solventan gracias a una búsqueda extensa con algoritmos de emparejamiento. En la práctica, la recuperación del movimiento llega a ser incluso más difícil todavía, puesto que se fuerza a usar imágenes muestreadas temporal y espacialmente con ruido aditivo de varias fuentes y otras dificultades ambientales como sombras, reflejos, límite de rango dinámico de sensores, etc.. En la figura 1.4, se tiene una representación de 3 fotogramas tomados en un intervalo de 1 segundo, obsérvese la dificultad que existe en la recuperación de movimiento si no hay más información.

Figura 1.4. Tres fotogramas de una secuencia de imágenes. Se aprecia un movimiento diagonal hacia la izquierda.

1.3. Ingeniería Neuromórfica

La naturaleza ha diseñado sistemas para procesamiento de visión, cada uno de los cuales se adapta soberbiamente a las necesidades del organismo específico. Los insectos por ejemplo, se dirigen hacía un blanco, interceptan, navegan, detectan obstáculos y reconocen patrones de una manera muy superior a la de los robots actuales, e incluso más, realizan todo esto usando menos de un millón de neuronas [IND00]. Se puede decir que estos algoritmos visuales forman el *software* y las arquitecturas neuronales, el *hardware* del organismo y han tenido que evolucionar para ser altamente eficientes en términos de velocidad, espacio, uso de energía y robustez. Estas propiedades, deseables por ejemplo en un robot, son una fuente de motivación en varias líneas de investigación, que tratan de emular las soluciones de la naturaleza [HIG01], [LIU00], [IND99], [HAR98], [FRA92].

Cada vez más se pone de manifiesto la importancia de mejorar las arquitecturas *hardware* frente al *software*. Por ejemplo, los transistores son cada vez más pequeños siendo la principal dificultad el diseñar una adecuada red de interconexión, mientras que la sincronización de circuitos integrados es un problema creciente, siendo además la actual eficiencia energética varios órdenes de magnitud por debajo de la del cerebro.

La naturaleza pues, ha dotado a los seres vivos de una serie de complejos sistemas para que puedan sobrevivir en el entorno natural que los rodea. Los sistemas denominados bioinspirados tratan de emular artificialmente el comportamiento de los sistemas biológicos naturales; concretamente, los denominados sistemas neuromórficos [MEA90] se basan en la consecución de buenas primitivas para la construcción de sistemas más complicados ya que la salida de cada subsistema neuromórfico es más simple que su entrada. Esta reducción de datos, ayuda en la enorme tarea de reintegrar todas las respuestas de todas los canales de información diferentes (por ejemplo, navegación autónoma y satisfactoria de robots gracias a modelos de visión de insectos [COL02], [WEB97], [FRA92]).

A medida que la tecnología avanza, es razonable centrar la atención en sistemas biológicos más avanzados y versátiles que los de los insectos, de forma que se puedan imitar tareas y comportamientos cada vez más complejos. De hecho, existen varios sistemas biológicos en los que inspirarse, particularmente el del humano, fijándose en las habilidades del sistema visual humano, su robustez y versatilidad.

Aunque no se tiene acceso directo a los métodos usados por el cerebro para recuperar movimiento, hay un número ingente de disciplinas para abordar desde diferentes puntos de vista los problemas que surgen:

- **Física.** Explica la naturaleza de la luz, óptica y formación de imágenes y proporciona un modelo del mundo real donde opera la ciencia de la visión.
- **Anatomía.** La Naturaleza ha desarrollado estructuras físicas requeridas para visión que ayudan en la investigación. Varios ejemplos son los detectores de movimientos elementales [REI61], detectores de borde [HUB65] y otras tareas de bajo nivel [MAR82], [BOL97].
- **Fisiología.** Los sensores ópticos de varios tipos imitan a la Naturaleza. La arquitectura de la red neuronal ha intentado simular la lógica usada por la biología resolviendo problemas del mundo real [YOU01], [YOU93], [REI61], [HUB65].
- **Psicofísica visual.** Permite probar el rendimiento de un sistema visual completo presentando un estímulo y midiendo la respuesta o percepción.
- **Modelos computacionales.** Permiten testear teorías desarrollando modelos de forma que se pueden comparar rendimientos y hacer predicciones.
- **Diseño electrónico.** Igual que en los modelos computacionales, pero diseñando circuitos electrónicos.
- **Neuropsicología.** Las investigaciones de lo que ocurre en el cerebro después de un daño pueden ser muy reveladoras. Por ejemplo, una lesión que provoque ceguera frente al movimiento específicamente, puede sugerir un sistema neuronal especializado [ZIH83].
- **Imagen Cerebral.** Técnicas como la Tomografía por Emisión de Positrones (PET) y las Imágenes de Resonancia Magnética Funcional (fMRI) permiten medir la actividad directamente, en áreas del cerebro como respuesta a la estimulación visual. Esto puede revelar la localización e interacción de subsistemas visuales en el cerebro [WAN99].

El principal algoritmo tratado en esta memoria, ha sido desarrollado como parte de un esfuerzo de investigación teniendo como punto de mira el sistema visual humano, por lo que es un modelo bioinspirado. El modelo también permite hacer predicciones que pueden ser testeadas a través de experimentación psicofísica, como por ejemplo la predicción de varias ilusiones ópticas de movimiento observadas en experimentos reales [JOH94]. Sin embargo, hay importantes diferencias en cuanto a los objetivos de construir un modelo de percepción

humana de movimiento para investigación en psicofísica y construir un modelo para ingeniería sobre *hardware* reconfigurable suficientemente eficiente para que funcione a modo de co-diseño o que aporte datos en tiempo real. Los objetivos del primero, desarrollado por Johnston [JOH99], fueron emular un proceso de visión que permita la formulación de experimentos y predicciones para tener un conocimiento avanzado del sistema visual humano.

El objetivo del presente trabajo de investigación, es diseñar una versión de co-diseño en *hardware* de forma que siente un precedente para que poder ser ampliada en diversas placas de prototipado y usada en aplicaciones en el mundo real.

Es interesante para este estudio, comparar los sistemas de visión del ser humano y el de una máquina usando el lenguaje de la visión artificial:

El sistema visual humano comienza con un par de lentes de 10 Megapuntos de resolución en color, acopladas en tiempo real con auto-iris y foco variable, desarrollando una imagen limitada por difracción. El sensor consigue una resolución de 1’ en la región central con resoluciones más bajas en la periferia, permitiendo alta resolución en la dirección de la mirada fija mientras se mantiene un largo campo visual sin un gasto excesivo de ancho de banda. La distribución de puntos no ordenados también elimina los problemas de *aliasing* o submuestreo espacial, mientras que la captura continua del sensor, elimina el temporal. El auto iris en tiempo real y control de autoganancia permite un rango de trabajo de intensidad de ocho órdenes de magnitud (aunque la autoganancia es lenta y su ajuste puede tardar varios minutos). Los rangos del *autofocus* son desde 3 cm al infinito. Existe *hardware* de pre-procesamiento de imágenes en el sensor capaz de realce de bordes y compresión de datos en tiempo real, orientable a través de tres ejes con tasas superiores a 900º/seg. El principal procesador trata cada punto en paralelo empleando una estrategia de agrupamiento de información multiescala y las rutinas de procesamiento de imágenes tienen altos órdenes de filtrado espacio-temporal con filtros ajustados para orientación, velocidad, tamaño y otras estructuras locales. La construcción completa tridimensional de la escena es estándar, como es la segmentación avanzada y los sistemas de reconocimiento de patrones; además, el sistema trabaja en tiempo real con una latencia variable no más larga de 100 mseg.

El sistema típico de visión por computador, sin embargo, comienza con una cámara CCD de 3 Megapuntos de resolución colocada en una red cartesiana regular. Las medidas de luminancia son cuantizadas a ocho bits de precisión (donde el ruido usualmente domina el bit menos significativo), los parámetros para el foco y el iris son ajustados de forma manual y el rango de trabajo de la intensidad suele ser pequeño. El procesamiento de la imagen se realiza

raramente en el chip, excepto para sistemas experimentales que implementan funcionalidades básicas a baja resolución; el *hardware* de procesamiento de imágenes es capaz de llevar a cabo una sola tarea de visión como el filtrado o la segmentación en una arquitectura típica de procesamiento serie. Por lo tanto, no es sorprendente que la mayoría de los sistemas de visión artificial no sean robustos o suficientemente versátiles para operar en el mundo real, al contrario que el sistema visual humano.

Existen varias líneas de trabajo para los sistemas de visión neuromórficos, con el objeto de mejorar los algoritmos de visión y arquitecturas tradicionales [MEA90]. Dentro de éstos se encuentran los pequeños sensores denominados “retinas de silicio”, donde en el mismo sensor está implementado el procesamiento especializado [BOL97], [ARI96]. Alternativamente, los chips de visión implementan modelos de baja visión biológica sobre *hardware* en la última etapa del procesamiento [COL02], [IND00], [LIU00], [HAR98].

Los sensores neuromórficos, se corresponden con implementaciones *hardware* de algoritmos simples de visión, pues procesan los valores de luminancia directamente usando circuitos analógicos acoplados a su imagen de entrada; al ser, además, masivamente paralelos trabajan en tiempo real para dar imágenes de salida a través de tensión o corriente, señales que podrían ser usadas como entradas de otros sistemas más complejos en cascada.

Estos sensores proporcionan velocidad [HAR98], [DEU98], [ARI96], [FRA92], rastreo [IND99], conducen la atención visual [IND00] y muestreo en el espacio no lineal [BOL97]. Son extremadamente útiles para simplificación del tamaño de los datos y se comportan como excelentes primitivas capaces de emular el comportamiento de los insectos, usando bloques de construcción [LIU00], [MEA90]. Sin embargo, al igual que los insectos, los sensores se confunden fácilmente debido a sus algoritmos simples; para mejorar estos algoritmos se necesita más densidad de transistores en el chip y esto lleva (ya que los componentes son de un tamaño finito) a un compromiso entre la resolución y la complejidad del sensor. A modo de ejemplo se pueden citar los sensores 20×10 [ARI96] y 16×1 [IND99] bastante probados aunque limitados en cuanto a resolución. Harrison [HAR98] describe un sensor que implementa el detector de movimiento elemental de la mosca directamente en un sensor CMOS a través de un proceso VLSI. La arquitectura paralela del sensor permite operar en tiempo real y proporciona un movimiento cualitativo de señales que podrían ser usados como entrada de un nivel de procesamiento más alto. Un sensor unidimensional que use fotorreceptores adaptativos, derivadas espaciales y detección de movimiento es capaz de seguir un estimulo de alto contraste, el cual es descrito por Indiveri [IND99].

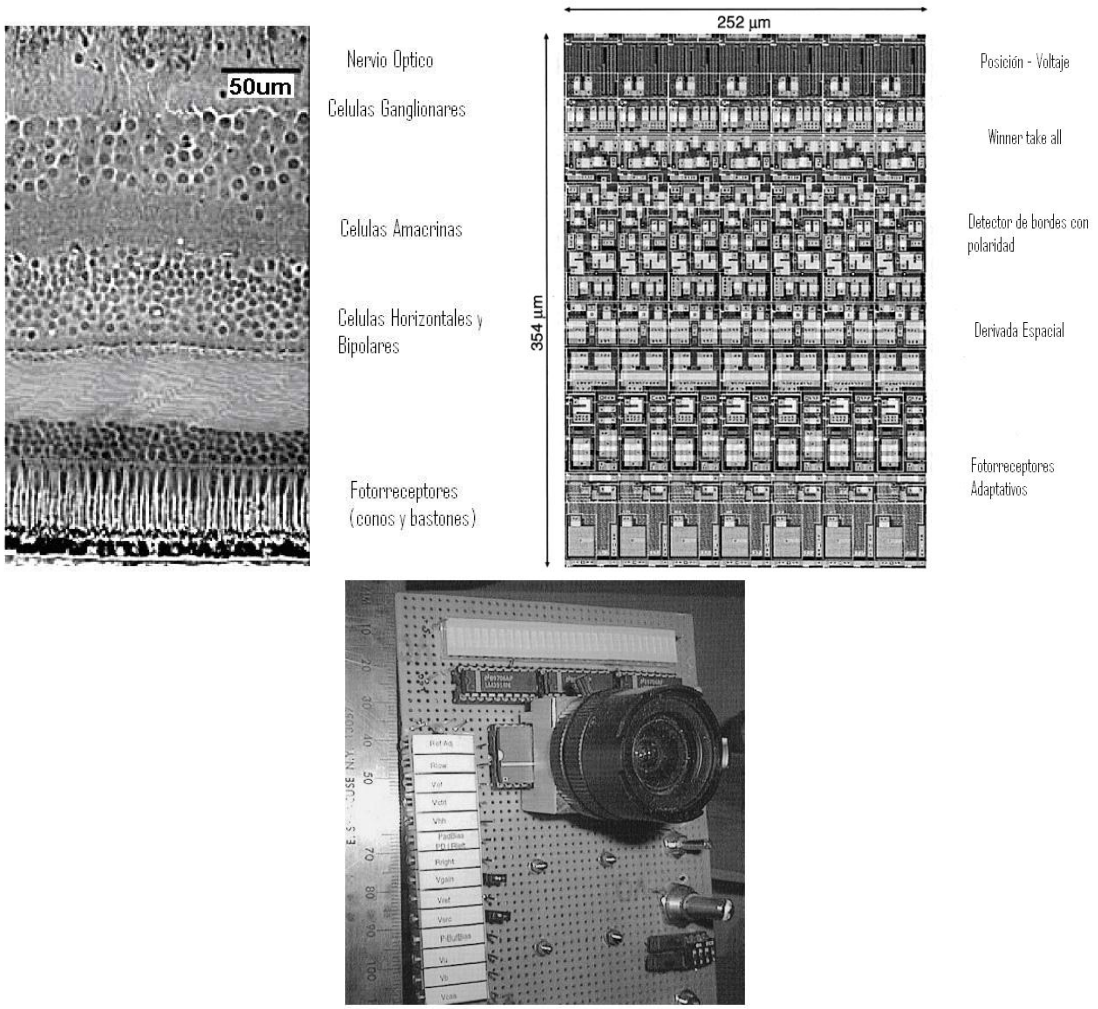


Figura 1.5. Sensor neuromórfico con circuitería analógica (superior derecha) e inspirado en la retina humana (superior izquierda). Fotografía del chip neuromórfico (abajo).

Este sensor de silicio (Figura 1.5) está basado en la estructura de una retina humana [DOW87] que inspira un sensor neuromórfico que usa circuitería analógica y, aunque la computación es muy diferente a la realizada en biología, lleva a cabo un procesamiento elemental en una estructura de capas para construir un detector de movimiento en tiempo real.

Otra ventaja importante de los sistemas visuales biológicos sobre la mayoría de los de visión por computador, es que aquéllos son activos, en el sentido de que son capaces de cambiar sus parámetros de adquisición de datos, basados en la interpretación de la escena (un sistema pasivo, no podría nunca hacer esto). Por ejemplo, un sistema activo puede seguir objetos moviendo sus sensores, navegar a lo largo del entorno y fijar regiones de interés [SON99], [SCA99]. La habilidad de un sistema activo para intentar adquirir más y mejores datos en situaciones difíciles lleva a resolver muchos problemas de visión (como extracción de la forma a través de rayos X, estructura de movimiento, etc.) que un sistema pasivo tiene

vetado; sin embargo, para que un sistema activo interaccione con el entorno, debe operar en tiempo real, si no su interacción es limitada.

Uno de los principales problemas de los sistemas activos es que la tasa de procesamiento cae rápidamente a límites insospechados, teniendo que fusionar las salidas de profundidad, movimiento y reconocimiento, introduciendo unidades neuromórficas diferentes que tienen salidas sencillas, especialmente aquéllas que trabajan a nivel del sensor.

1.4. El Modelo Multicanal de Gradiente (McGM)

El algoritmo neuromórfico bioinspirado abordado en esta tesis doctoral es el Modelo Multicanal de Gradiente (McGM) de percepción humana de movimiento desarrollado por Johnston [JOH94], [JOH99]. Aunque se analizará con todo detalle en el capítulo III y apéndice II, es útil tener una visión muy general del mismo en este punto.

La primera etapa del algoritmo consiste en medir las respuestas de varios filtros espacio-temporales que serán diseñados al efecto. Los filtros espaciales corresponden a las llamadas celdas simples de la corteza visual estriada, modelándose como derivadas de funciones Gaussianas [YOU01], hasta séptimo orden en una dirección, denominada dirección primaria y segundo orden en una dirección ortogonal, tal y como se puede apreciar en la estructura rectangular de celdillas, correspondiente a la figura siguiente.

Los filtros temporales representan cada uno de los tres canales temporales encontrados en los humanos según los experimentos de Hess y Snowden [HES93] [SNO94] (*v*. apéndice

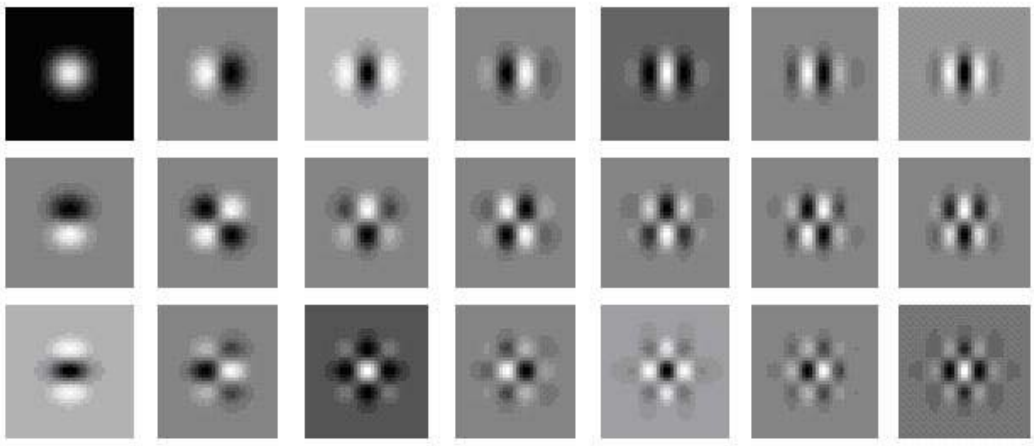


Figura 1.6. Filtros normalizados de derivadas de funciones Gaussianas de hasta orden 7 en la dirección primaria y hasta orden 2 en la dirección ortogonal.

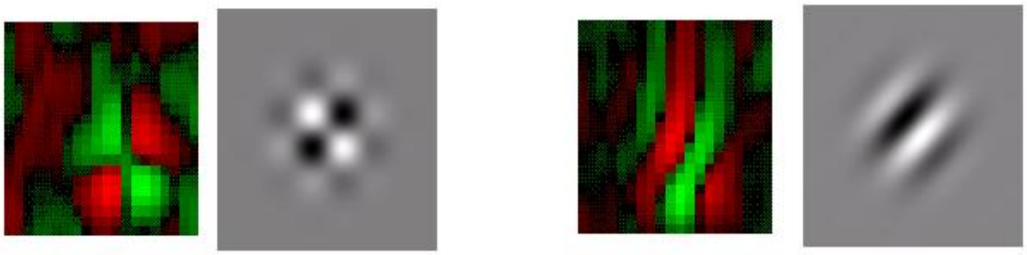


Figura 1.7. Filtros encontrados en el sistema visual del mamífero (coloreados para tener más detalle) y su implementación sintética.

III), modelándose estos filtros sin embargo, como derivadas de Gaussianas en el logaritmo del tiempo [JOH94]. Normalmente estos filtros se orientan a través del rango completo de 360 grados cada intervalo de 15 (24 orientaciones), simulando las columnas encontradas en la región V1 de la corteza visual (Figuras 1.7 y 1.8, también apéndice III), aunque también pueden tomar otros valores.

La estrategia de hacer múltiples medidas mejora la robustez del algoritmo, conformando las respuestas de los filtros una serie de primitivas, que combinadas adecuadamente proporcionan medidas de velocidad inmunes al ruido estático.

La dirección se extrae midiendo como el cambio de la magnitud de la velocidad varía con la orientación, a través de un mecanismo que permite separar la componente asociada a translación de movimiento de sus componentes diferenciales [KOE86]. Cada una de las versiones funcionales del modelo completo implementadas, poseen un gran número de parámetros y grados de libertad correspondientes a ámbitos independientes (tecnológicos, bioinspirados y arquitecturales). La versión completa en *software* requiere una cantidad ingente de tiempo y recursos de computación (del orden de minutos por fotograma en un Pentium IV). A pesar de todo esto, se han desarrollado los diseños, sobre todo el *hardware* pensando en simplificación y eficiencia, guardando un compromiso entre exactitud y velocidad.

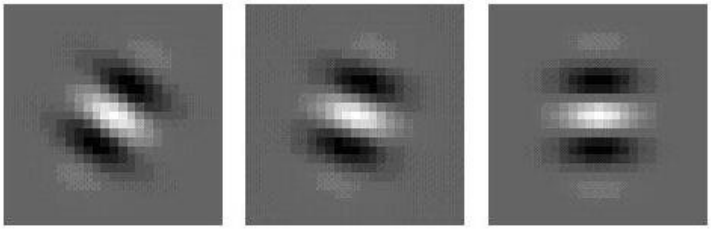


Figura 1.8. Ejemplo de filtros rotados, simulando las columnas encontradas en la corteza visual.

1.5. Implementación en *hardware*

El flujo óptico añade una dimensión temporal a los paradigmas típicos de visión por computador, lo que eleva a la categoría de extrema dificultad este tipo de problemas y retos con respecto a los encontrados en visión espacial. Los algoritmos en tiempo real tienden a ser optimizados hasta el punto donde los resultados son mediocres y ruidosos comparados con sus homólogos en *software* [MAR01], [ACC98], [SMI95]. Aunque este tipo de problemas se solventa con algún tipo de realimentación de corrección, como por ejemplo filtros de Kalman, especialmente en sistemas activos [DAN98], [SCA99], es necesario efectuar las medidas suficientemente rápidas.

Como se ha comentado previamente, existe un compromiso entre exactitud cuando se procesa *software* y velocidad cuando se procesa flujo en tiempo real realizando múltiples medidas. Se pretende que el algoritmo original sea preservado en tanto sea posible, desarrollando este compromiso entre exactitud y velocidad que, como se verá posteriormente, no es sencillo de implementar ni de cuantificar.

Es de esperar que con la implementación del algoritmo en tiempo real, aparezcan nuevos usos e interacciones no esperadas antes. La implementación en tiempo real de un modelo tan complejo como el McGM es un objetivo planteado desde el principio de este trabajo. Teniendo en cuenta una visión general del algoritmo se enumeran algunos de los principales problemas para su implementación:

- Adquisición de imagen en tiempo real.
- Filtrado temporal usando 3 derivadas del logaritmo de la Gaussiana como filtros temporales.
- Filtrado espacial usando filtros de derivadas de Gaussianas para múltiples orientaciones.
- Operaciones aritméticas de imágenes extensas (para formar el desarrollo de Taylor de cada punto de las mismas).
- Múltiples extracciones de medidas de velocidad.

El principal cuello de botella del proceso son las convoluciones. El modelo completo usa más de 100, con núcleos de convolución muy grandes (23×23) de hasta séptimo orden en las derivadas, proyectando el resultado cada 15 grados (a lo largo de 24 orientaciones). Las últimas etapas de procesamiento involucran operaciones aritméticas con determinantes de

matrices, que son los encargados de eliminar los efectos de ruido estático y componentes no asociadas a translación (*v.* apéndice II). Se estudiará en los siguientes capítulos el compromiso entre exactitud y eficiencia, según el número de bits en de cada filtro a lo largo de las diferentes etapas y el error final que se desee obtener.

1.5.1. *Hardware* para visión en tiempo real

Un gran número de algoritmos han sido desarrollados para estimar flujo óptico, algunos de ellos se han diseñado para resolver una tarea específica de visión, otros, como un intento de modelar la biología y otros, como soluciones noveles a un problema difícil.

Sin embargo, existen unos cuantos algoritmos que han sido diseñados para una implementación en tiempo real. Hasta nuestros días, el costo del tiempo real ha sido prohibitivo en términos de coste *hardware*, requiriendo tiempo de desarrollo y experiencia. Afortunadamente, la tecnología de visión artificial está en un momento donde se puede adquirir *hardware* relativamente barato junto con herramientas para su desarrollo. De hecho, existen varias opciones *hardware* para implementar los algoritmos y la elección del *hardware* será determinante en la metodología de implementación usada en esta trabajo de investigación, desarrollando a continuación las principales opciones disponibles.

- Elementos dedicados.
- Chips VLSI.
- Procesadores Digitales de señal (DSP) para procesamiento de imágenes.
- Procesadores generales para procesamiento de imágenes.
- Computadores personales. (PCs).
- FPGAs (en sus diferentes arquitecturas: grano grueso y fino (*v.* apéndice I)).

1.5.1.1. Elementos dedicados

Son circuitos electrónicos que realizan algunas de las funciones de procesamiento de imágenes de forma eficiente. Entre sus características destacan:

- Son económicos, ya que consisten en varios circuitos integrados sin un programa almacenado.
- Tienen alta velocidad y pueden procesar datos sin retardo.

- No son modificables, excepto algunos datos grabados y la capacidad de incluir o no (mediante control de *software*), la función de la imagen.

Algunos ejemplos de funciones ejecutadas por este tipo de elementos son LUTs, conversión de Bayer, FFC (*Flat Field Correction*), FLC (*Flat Line Correction*). Este tipo de circuitos no se suelen encontrar aislados sino que forman parte de otros módulos como cámaras, *frame grabbers* o elementos de procesamiento empotrados. También se encuentran en el manejo de la entrada/salida en tiempo real para el mantenimiento del *trigger* de la cámara.

1.5.1.2. Desarrollo de chips VLSI

La ingeniería neuromórfica se ha centrado en la explotación de los circuitos analógicos VLSI desde hace algún tiempo [MAR01], [IND99], [ARI96], [MEA90]. Son varias las ventajas de dedicar un procesador específico para un algoritmo determinado, entre ellas que el *hardware* es lo más rápido posible. Se puede aplicar la física de los dispositivos analógicos para conseguir robustez y bajo consumo de energía [MEA90]. Sin embargo, el diseño VLSI sólo es apropiado para algoritmos simples que puedan ser implementados sin muchas etapas de estimación; en caso contrario, el algoritmo llega a ser muy complicado para ajustarlo en un chip. Desafortunadamente, el rutado VLSI es caro en términos de diseño y costes de fabricación, además de que el *hardware* final es casi imposible de modificar.

1.5.1.3. Procesadores digitales de Señal (DSP) comerciales

Existen varios productos en el mercado que usan los procesadores digitales de señal (DSP), como los modelos de Texas Instruments TMS320DM64x [TEX06]. Los sistemas tienen un rendimiento muy alto para ciertas operaciones como las MACs (multiplicación y suma) que son claves en la convolución, la correlación y la transformada rápida de Fourier (FFT). Los DSPs modernos tienen funciones de propósito general de forma que cualquier procesamiento de imagen que requiera una relativa velocidad se puede implementar en ellos. La desventaja reside en que para escribir algoritmos eficientes a medida, se requiere experiencia en la programación de DSPs. No suelen tener mucho soporte y documentación relativa a funciones de entrada/salida, se suelen programar en C/C++, cargándose luego para su ejecución, dependiendo el éxito de la tarea del compilador de DSP que se encarga de

optimizar el código para la arquitectura específica, además de que hay un límite en el tamaño y la complejidad del programa que puede ser ejecutado en DSPs de propósito general.

1.5.1.4. Procesadores de imágenes embebidos de propósito general

Este tipo de plataformas están disponibles con un rango amplio de capacidades. Aunque normalmente no son tan rápidos como DSPs dedicados, su arquitectura de propósito general permite la implementación de algoritmos más complicados.

La ventaja de estas plataformas proviene de un bus configurable con un gran ancho de banda hacia la memoria de imagen, de forma que se transfieren las imágenes adquiridas al procesador más rápidamente que con un computador convencional. Entre ellas, unas de las más destacadas son la placa *Genesis* de Matrox [MAT02] y la placa *Mamba* de Coreco [COR06].

La plataforma *Genesis* consiste en un DSP de propósito general (TI de la serie C80 o un G4PowerPC) junto con chips de aceleración para varias funciones de procesamiento de imagen. Esta plataforma puede ser programada fácilmente usando un conjunto de bibliotecas, es muy flexible y dispone de un bus con ancho de banda de 400MB/seg.

La plataforma *Mamba* usa un PentiumIII embebido con Windows NT, similar a un PC convencional. En ella el software se escribe para Windows y se copia dentro de la plataforma para ejecutarse como un programa convencional. La contrapartida es que el procesador de *Mamba* no es tan rápido como un DSP dedicado, ya que no hay un bus de datos dedicado a la CPU.

Mamba ha evolucionado y, actualmente, se dispone de *Anaconda* [COR06] formada por una solución mixta que combina adquisición de imagen de alta velocidad con una FPGA y un PowerPC. Las FPGAs aportan velocidad, consumo moderado, flexibilidad y tiempos de respuesta predecibles, aunque no son particularmente eficientes en tareas que contienen toma de decisiones o bifurcaciones. Muchas funciones de procesamiento de imágenes requieren búsqueda de patrones, decisiones iterativas, ramificación inteligente, etc., conformando bloques muy importantes en problemas de visión artificial .Con el PowerPC se tiene un procesador con baja disipación de energía y un conjunto de instrucciones apropiadas para estas tareas; como resultado, *Anaconda* contiene los dos tipos de tecnologías en una misma plataforma.

1.5.1.5. Procesadores de computador personal (PC)

Un procesador de propósito general como Intel Pentium [INT06], IBM PowerPC [POW06] o AMD [AMD06] es capaz de ejecutar algoritmos de procesamiento de imágenes. Si el procesador se inserta como parte del elemento de procesamiento dedicado y no actúa como *host*, puede liberarse de manejar otras tareas, trabajos de entrada/salida, etc., de forma que se incrementa la velocidad y si se elige un sistema operativo adecuado, se puede obtener rendimiento en tiempo real.

Por consiguiente, el tiempo real es alcanzable con estos procesadores. Hasta ahora el mayor inconveniente de las arquitecturas PC es que no se podía acceder a la imagen tan rápidamente como se requería si se usa el bus de la memoria principal, siendo éste un factor muy importante de limitación. La aparición de buses con gran ancho de banda y procesadores con instrucciones dedicadas para operaciones de tratamiento de señal (MMX, SSE, etc.) han subsanado este problema, de forma que se puede pasar la imagen directamente a la memoria principal sin prácticamente carga de la CPU. Dado que la frecuencia de reloj de estos microprocesadores es como mínimo, un orden de magnitud mayor que la de un DSP, pueden llevar a cabo tareas diseñadas para *hardware* especializado. Su mayor desventaja es su tamaño y consumo de energía que los hacen inadecuados en muchas situaciones, como en el marco de los objetivos de este trabajo de investigación, donde se pretende llegar a un prototipo final para exportarlo a un dispositivo embebido, por ejemplo, dentro de un automóvil.

1.5.1.6. *Field-Programmable Gate Arrays* (FPGAs)

Las FPGAs (*v.* apéndice I) son relativamente recientes. Xilinx [XIL06] fabricó la primera FPGA hace unos 15 años y es uno de los fabricantes dominantes actualmente. Una FPGA consiste en un bloque de elementos lógicos (secuenciales y combinacionales), que pueden ser dinámicamente reconfigurables, con conexiones programables y entre cuyas características destacan:

- Tienen una gran flexibilidad en la configuración, lo que permite ajustar la tarea específica de procesamiento.
- Operan con un reloj bastante más lento (MHz) que un microprocesador convencional (GHz), pero pueden ser más rápidas que éste, debido a que ejecutan varios bloques en paralelo, además el número de ciclos/operación puede ser menor que en un procesador.

- Con respecto a una persona formada en *software*, requieren experiencia y conocimiento *hardware* si se quieren realizar diseños eficientes.

Las FPGAs son excelentes elementos de preprocesamiento, operaciones de vecindad (convoluciones) y segmentación. Son menos efectivas para extracción de características y algoritmos de interpretación que usen bifurcaciones condicionales; además, son muy adecuadas para implementar tareas que se quieran desarrollar en tiempo real. No suelen utilizarse como elemento de procesamiento aislado, ya que, usualmente, suelen estar acopladas con otro módulo como, por ejemplo, un *frame grabber o* se comportan como un coprocesador.

En conclusión, las FPGAs son circuitos VLSI reprogramables en tiempo de ejecución, capaces de ofrecer rendimiento comparables a chips VLSI pero con mayor flexibilidad. El número de unidades lógicas (puertas) hace unos años era muy pequeño (aproximadamente unas 300.000) de modo que sólo se podían implementar en ellas algoritmos muy simples, como por ejemplo sincronización de video. Actualmente con dispositivos de entre 10 y 12 millones de puertas equivalentes la situación ha cambiado completamente, siendo posible realizar satisfactoriamente convoluciones, FFTs o algoritmos básicos de flujo óptico en tiempo real [ZUL98], [MAR05], [DIA06].

Las FPGAs están sujetas a diversas clasificaciones según su granularidad, profundidad, reconfigurabilidad, interfaz y modelo de computación. En el apéndice I se profundiza más en este tipo de plataformas dado que han sido elegidas para el desarrollo e investigación de los sistemas de estimación de flujo óptico tratados en este trabajo.

1.5.2. Consideraciones respecto al tiempo real para visión artificial

En al apartado anterior se ha hecho una breve revisión general de los elementos de procesamiento que se suelen usar para tareas de visión por computador. Si los datos que se han de procesar necesitan unos requerimientos notables (por ejemplo, un sistema que necesitara de unos 20 Mpuntos/segundo) o si el procesador debe realizar otras tareas, es necesario usar recursos adicionales organizando la arquitectura de todos estos elementos de procesamiento.

Para hacer una clasificación formal y académica, se podría usar la taxonomía de Flynn [STA06]; sin embargo, en procesamiento de imágenes esta clasificación no es adecuada, ya

que se usan elementos dedicados no programables, aunque se puede realizar una clasificación con las salvedades que se comentan a continuación:

- Un solo elemento de procesamiento. SISD (Simple Instrucción, Simple Dato).
- Elementos multiplexados. Tienen la ventaja de replicar el diseño de un solo elemento. La desventaja es la complejidad y la necesidad de diseñar un demultiplexor para enviar las imágenes a los elementos de procesamiento. El multiplexor combina la entrada/salida de otros procesadores y el control para todos esos elementos adicionales. Por ejemplo, en una aplicación que requiere aceptar una imagen cada 0.1 segundos y en la que la latencia individual es 0.5 segundos, se pueden emplear 5 elementos de procesamiento ejecutando la misma función y la misma imagen en paralelo. Esta clasificación sería MIMD (Múltiples Instrucciones, Múltiples Datos).
- Elementos paralelos. Es una configuración muy práctica para preprocesamiento, pero es menos óptima para segmentación o extracción de características de imágenes. Se basa en que cada elemento procesa una porción de la imagen de forma paralela, donde estos núcleos de procesamiento comparten la memoria. Como ejemplo se tiene el conjunto de instrucciones Altivez del PowerPC o MMX, SSE para el Pentium. Se asemejaría a una clasificación SIMD (Simple Instrucción, Múltiples Datos).
- Elementos en serie. Es la técnica más usada cuando el ancho de banda del procesamiento de imagen tiene que ser aumentado. Cada elemento procesa la imagen y luego pasa la imagen procesada al siguiente módulo. La ventaja de este tipo de esquemas es que cada elemento se optimiza para una función particular (por ejemplo, el primer procesador podría ser un elemento dedicado, el segundo una FPGA realizando preprocesamiento y el tercero un PowerPC realizando segmentación y extracción de características). Esta filosofía trabaja con diferentes latencias y se necesitan añadir capacidad de almacenamiento entre estos elementos para almacenar los resultados.

Además de esta clasificación, existen otras técnicas para procesamiento de imágenes de alta velocidad:

- Evitar la limitación de ancho de banda en la ruta de datos (examinando cuidadosamente la forma en que el diseño afecta a la ruta del bus y a la memoria).

- Alinear los datos en memoria (buscar grupos de 32 ó 64 bits para evitar ciclos de acceso extras).
- Optimizar el uso de la memoria caché (asegurando que es suficiente y escribiendo *software* para minimizar los fallos de la misma).
- Usar procesamiento vectorial (empleando instrucciones específicas del tipo Altivez o MMX/SSE2 siempre que sea posible).
- Usar regiones de interés (ROIs) en los programas de procesamiento de imágenes y los de adquisición de la cámara.
- Usar una cámara asíncrona (con esto se evita la incertidumbre en la latencia de adquisición de imágenes).
- Usar un sistema operativo en tiempo real o cercano a serlo (eliminando procesos y *drivers* innecesarios, archivos de intercambio, etc.).
- Compensar las variaciones de latencia con una resincronización, usando una cola de salida donde el dato se inserta dependiendo de la diferencia entre el tiempo actual y el tiempo en el que se capturó este dato.

1.5.3. *Hardware* y herramientas de diseño usadas en este trabajo

Teniendo en cuenta las características de cada elemento de procesamiento y los recursos disponibles, para llevar a cabo este trabajo de investigación se ha elegido una plataforma *hardware* constituida por una FPGA embebida junto con RAM externa, que se puede conectar a un computador personal (con un microprocesador Pentium IV con arquitectura RAMBUS, 1 GByte RAM) a través del bus PCI.

La versión completa del algoritmo multicanal de gradiente (McGM) supera con creces las características de la plataforma *hardware* disponible, por lo que se ha tenido que diseñar una versión reducida del algoritmo McGM en co-diseño, de forma que no comprometa los resultados del modelo puramente secuencial. Esta nueva versión se ha implementado manteniendo la arquitectura neuromórfica y bioinspirada del algoritmo original, aunque incluso con esta versión final se hace difícil implementar el algoritmo con las características propias de la FPGA utilizada, una VirtexE BG-560. Dado que la tecnología avanza rápidamente, ya hay en el mercado FPGAs con multiplicadores embebidos, más de 10 millones de puertas o procesadores específicos, entre otras características. Este trabajo de investigación supone el primer intento de migrar el algoritmo del dominio del modelo psicofísico computacional a la implementación en *hardware* reconfigurable general.

Las características más relevantes del sistema de desarrollo empleado en este trabajo se resumen a continuación.

Sobre la placa usada, RC1000 de ALPHADATA [ALP06]:

- Tarjeta PCI (universal, 2.1) de 64 bits, 4 bancos independientes de SRAM con 2 Mbytes/banco, 2 relojes programables (el primero, síncrono para bus local e interfaz PCI 400kHz-40MHz, el segundo, para aplicaciones de usuario, 0-100MHz), configuración por PCI y JTAG y posibilidad de usar DMA.

Sobre el chip, VIRTEX E BG 560 2000 [XIL06]:

- $2.5 \cdot 10^6$ puertas, 655360 Bits de Block Ram, 614000 Bits de memoria distribuida. Matriz de CLBs 80×120.

En la figura 1.9 aparece representada la arquitectura, así como una fotografía de la placa de prototipado utilizada en el presente trabajo de investigación.

Teniendo en cuenta la complejidad del sistema que se va a diseñar, se ha elegido el lenguaje de programación Handel-C [CEL06] para la descripción *hardware*, puesto que acelera la velocidad de prototipado, permitiendo el diseño de arquitecturas a través de diferentes niveles de abstracción.

La contrapartida está en el consumo de recursos y el grado de opacidad frente a la síntesis final cuando se describen este tipo de arquitecturas especializadas haciendo uso de estas herramientas.

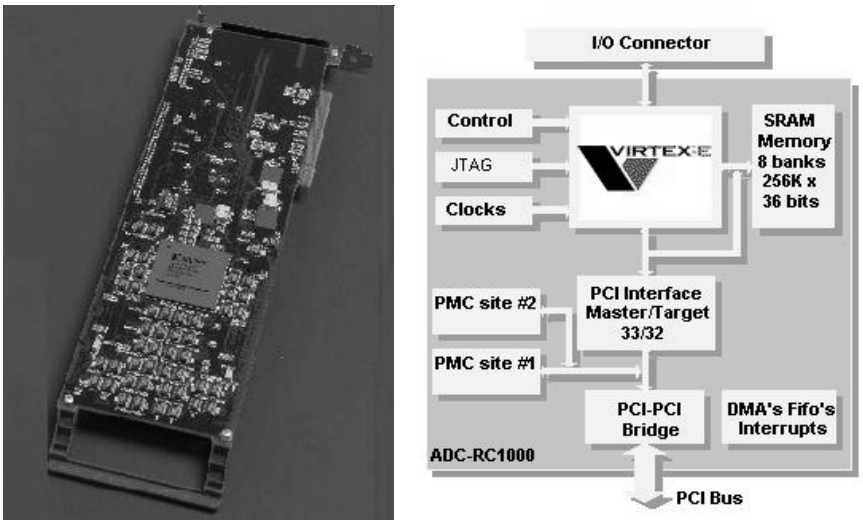


Figura 1.9. Tarjeta usada en este trabajo de investigación junto con el diagrama de la arquitectura de la placa con el chip BG560 2000E.

El kit de diseño DK *(Design Kit)* es un entorno completo de trabajo para entradas de naturaleza algorítmica basadas en ANSI-C a través de Handel-C, simulación y síntesis. Incluye un sistema de co-diseño y co-verificación además de unas pasarelas desde C a RTL y desde C a síntesis de FPGAS, con una serie de propiedades a destacar:

- Aceleración algorítmica completa (de la especificación a la implementación).
- Generación automática de IEEE-RTL, VHDL y Verilog.
- Características de síntesis avanzada para FPGA y SOC.
- Generación automática de *netlist* para FPGAs (incluyendo los últimos modelos como Cyclone II, Spartan 3, Stratix II, Virtex 4, Virtex 5).

estas funcionalidades se traducen en una serie de ventajas, resumiendo a continuación las más relevantes de cara a una implementación práctica:

- Permite realizar mejores diseños y de forma más rápida.
  Es posible diseñar y modelar sistemas en C además de explorar, simular y verificar rápidamente diversas arquitecturas *hardware-software*, encontrando particiones óptimas y sintetizando directamente en la FPGA.

- La aceleración del algoritmo es notable.
  Con el entorno DK se pueden acelerar los algoritmos en *software* y los cuellos de botella en *hardware*, migrando incluso desde los algoritmos secuenciales a una implementación paralela en *hardware*, generándose todo el código necesario EDIF/RTL.

- Se maneja mejor la complejidad y se reduce el riesgo.
  El desarrollo de sistemas modulares se realiza mediante modelos en C con abstracción mixta. El hecho de tener un lenguaje y metodología de diseño *hardware-software* comunes mejora la comunicación y reduce el riesgo.
  Además, se usa un entorno que favorece un mantenimiento apropiado, la programación de tests y la compartición de varios recursos, como código, bibliotecas y modelos de sistemas, con todos los diseñadores, desde los estados de la especificación iniciales a la

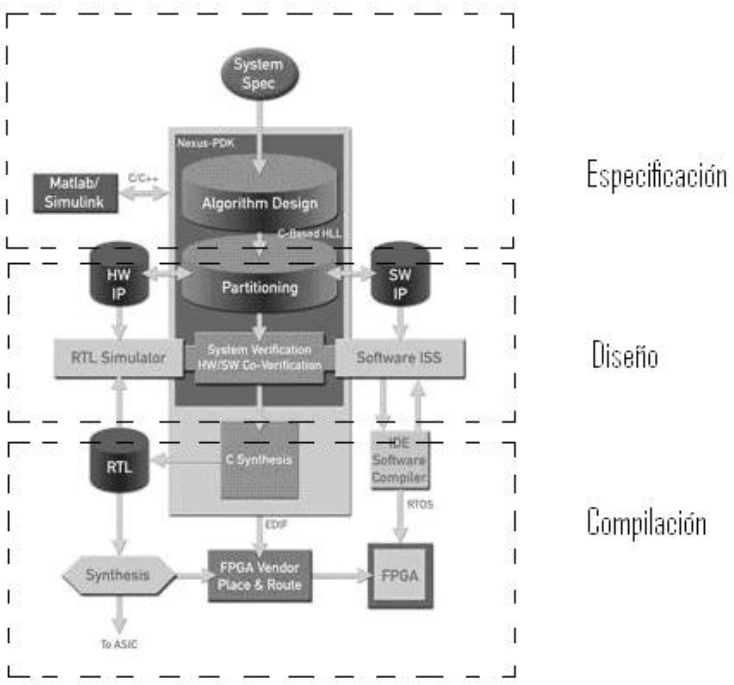


Figura 1.10. Esquema de flujo de diseño completo usando DK.

implementación final. Se representa en la anterior (figura 1.10) el flujo de diseño completo, para la herramienta Design Kit de Celoxica (DK).

- Facilita el diseño en equipo.

  La exploración desde el punto de vista arquitectural llega a ser muy rápida para diseñadores de sistemas, con implementación directa de algoritmos complejos para diseñadores de *hardware*, simulación rápida y co-verificación *hardware-software* para especialistas en verificación, proporcionando un entorno amigable, siendo casi directo el paso para ingenieros de *software* que quieran empezar a trabajar en *hardware*.

El uso de esta plataforma, facilita un flujo rápido desde el algoritmo hasta su implementación, ya que DK proporciona un entorno integrado de desarrollo (IDE) desde la entrada de diseño, simulación y exploración arquitectural a través de su síntesis e implementación.

Esto implica un manejo más funcional del proyecto gracias, entre otras herramientas, al editor integrado. Adicionalmente, se cuenta con un depurador multihebra con exactitud de ciclo, con opción de ejecución de un paso, puntos de ruptura, varias ventanas de inspección de

variables, control de hebras y la posibilidad de escribir el algoritmo en código fuente mixto C/C++ o Handel-C.

Con todo esto, la implementación queda optimizada, ya que el diseño y la síntesis se ajustan mejor para arquitecturas específicas. Por ejemplo, el desarrollo de un código fuente con estimaciones detalladas de tiempo o área permite experimentar con diferentes algoritmos y diseñar estrategias de optimización antes del posicionado, relacionando las salidas con este código fuente, refiriendo los análisis de tiempo a áreas del mismo para su optimización.

Por otra parte, es posible la asignación de restricciones temporales a señales de entrada/salida; también existe una arquitectura optimizada para FPGAs con varias ventajas, como uso de primitivas ALUs embebidas, *pipelining* automático de acceso a RAM, balanceado del *pipeline* (movimiento de *flip-flops*), etc..

La capacidad de simulación comprensiva es más flexible, siendo además su verificación, idónea y potente. Las descripciones *hardware* se pueden simular con la ventaja de tener una verificación funcional con descripciones mixtas, que ayuda al diseñador en la exploración de decisiones basadas en partición de bloques, así mismo, mediante un refinamiento sucesivo y transformación interactiva de algoritmos *software* con funciones del sistema que hacen de pasarela para *software-hardware*, se posibilita una simulación con diferentes niveles de detalle, llevando todo ello a ganar velocidad (en comportamiento, gracias a C/C++, y en exactitud de ciclo, gracias a Handel-C).

También, se pueden escribir los patrones de test en C/C++ desarrollando modelos más elaborados, de forma que tengan entradas/salidas desde y hacia otras herramientas, reutilizando estos mismos patrones a través de etapas del módulo de diseño *hardware*. Es posible y sencillo, además, enlazar con otros simuladores para Handel-C, con posibilidades de co-simulación en VHDL, Verilog, System-C o Simulink.

Por otro lado, DK permite que los programas puedan ser escritos como ejecutables en *software* o DLLs externos para su inclusión en otros modelos del sistema, incluso para verificación, protección y distribución de IPs. El diseño y posterior reutilización de IPs proporciona importantes bloques de código externos (EDIF/VHDL/Verilog), siendo posible también el uso de bibliotecas *wrapper* para simplificar la interfaz con IPs externos y complejos, además de la creación y distribución de IPs mediante bibliotecas compiladas.

En definitiva, DK proporciona un entorno de trabajo en el que se puede realizar el flujo completo de diseño, donde Handel-C se usa para la descripción de *hardware* y C/C++ para la simulación de las especificaciones del sistema, tests y codiseño *hardware-software.*

Todas las construcciones en Handel-C se pueden simular con precisión de ciclo e implementarlas en formato EDIF (*netlist* para FPGASs) y RTL (VHDL o Verilog). Esta última salida preserva la jerarquía de la fuente de Handel-C permitiendo verificar a nivel RTL. El compilador genera código RTL, preparado para ser usado con las últimas herramientas de síntesis, simuladores de tiempo y flujo de diseño ASICs, usándose en este trabajo diversas versiones de ISE Generator de Xilinx, desde 5.2 a la 7.0.

1.6 Conclusiones

Este capítulo ha servido como introducción a varios conceptos como el flujo óptico, la ingeniería neuromórfica, los sistemas bioinspirados y las plataformas y herramientas de diseño empleadas para desarrollar prototipos que permitan llevar a cabo la investigación presentada en esta memoria.

Los problemas relacionados con visión por computador (especialmente los dinámicos, como es el caso del flujo óptico) están cambiando continuamente, interrelacionándose con otras disciplinas (desde las matemáticas hasta la biología, pasando por la tecnología, etc.). Debido a estas características especiales, se requiere enfocar el problema desde todos los puntos de vista disponibles, para saber el alcance de cada aportación y poder resolver las dificultades encontradas mediante soluciones en diversos campos de abstracción, puesto que se está tratando con el mundo real y se tienen que hacer frente a multitud de desafíos, muchos de ellos no explicados todavía ni siquiera de forma teórica, aunque eso sí, resueltos por la naturaleza desde hace mucho tiempo.

CAPÍTULO II

Estimación del flujo óptico

2.1. Introducción

En este capítulo se examinan algunos de los algoritmos utilizados más frecuentemente en visión por computador y estimación del flujo óptico, su inspiración biológica y su eficiencia computacional. Se han propuesto muchos métodos para la estimación del flujo óptico, que emanan de la inteligencia artificial, teoría de la señal, robótica, psicología y biología. Existe una amplia bibliografía al respecto y aunque no es el objetivo de este capítulo explicar todos los algoritmos, se hará una revisión del estado del arte lo más descriptiva posible con objeto de poder justificar el modelo a desarrollar en el siguiente capítulo, dentro de un entorno suficientemente clasificado, describiendo aquí, sólo las cuestiones más importantes y relevantes dentro de esta disciplina.

Cada campo brinda su particular aportación y motivación al problema, resultando un gran abanico de diversos algoritmos. A pesar de toda esta diversidad, se pueden clasificar los modelos de flujo óptico en tres categorías:

- **Métodos basados en correspondencias.** Operan comparando posiciones de la estructura de la imagen entre fotogramas adyacentes e infieren la velocidad a partir del cambio en cada lugar. Son probablemente los métodos más intuitivos.
- **Métodos diferenciales o de gradiente.** Trabajan usando derivadas de la intensidad de la imagen en el espacio y el tiempo. La velocidad se obtiene a partir de cocientes de las medidas anteriores.

- **Métodos de energía.** Se usan filtros orientados espacio-temporales que responden óptimamente a determinadas velocidades. Las estructuras usadas para este tipo de procesamiento son bancos de filtros en paralelo que se activan para un rango concreto de velocidades.

Las diferentes aproximaciones a la estimación de movimiento son apropiadas según la cada aplicación. Según el teorema del muestreo [NIQ06], una señal tiene que ser muestreada a una frecuencia de muestreo que sea como mínimo, el doble de la frecuencia máxima que posea tal señal. Se garantiza de esta manera que, en el campo del procesamiento de imágenes en movimiento, el desplazamiento entre 2 fotogramas es pequeño comparado con la escala del patrón de entrada. Cuando este teorema deja de cumplirse, aparece el fenómeno del sub-muestreo o *aliasing*, en imágenes espacio-temporales este fenómeno produce falsas inclinaciones o estructuras sin ninguna relación entre sí, como ejemplo de *aliasing* temporal, se puede observar rotar las llantas de las ruedas de los coches en dirección contraria a la verdadera. Es definitiva, no se pueden estimar desplazamientos largos provenientes de patrones de entrada con escalas pequeñas.

Aparte de este problema se tiene el denominado problema de la apertura (explicado en el capítulo I). Estos dos problemas (*aliasing* y apertura) conforman el llamado problema global de la correspondencia.

Por tanto, el movimiento de los patrones de entrada no siempre permite corresponder características de fotogramas consecutivos de una manera no ambigua. Esta correspondencia física puede ser indetectable debido al problema de la apertura, ausencia de textura (ejemplo de la esfera de la figura 1.2), largos desplazamientos entre fotogramas, etc., análogamente el movimiento aparente puede dar lugar a una falsa correspondencia. Para este tipo de situaciones, se pueden usar el tipo de algoritmos de emparejamiento (seguimiento y correlación), aunque actualmente existe una gran controversia sobre las ventajas e inconvenientes de usar estas técnicas en lugar de otras basadas en gradiente y energía de movimiento.

Los métodos de correlación son menos sensibles a cambios de iluminación, son capaces de estimar desplazamientos largos que no cumplen el teorema del muestreo [YAC99], sin embargo son extremadamente sensibles a estructuras periódicas proporcionando

múltiples mínimos locales y cuando se presenta el problema de la apertura, se obtienen salidas impredecibles.

Como alternativa, los otros métodos son mejores en eficiencia y exactitud, son capaces de estimar el flujo óptico perpendicular (en presencia del problema de la apertura) y se comportan mejor para desplazamientos cortos relativos al tamaño de los filtros locales usados. En el ámbito de los sensores biológicos, el muestreo temporal no es uniforme como ocurre en una cámara, en su lugar, el sensor (por ejemplo, la retina) se expone sobre el tiempo continuamente, permitiendo muestrear información de forma más efectiva, sugiriendo este efecto que los modelos de gradiente/energía son los más apropiados para sistemas biológicos.

Normalmente en visión artificial, se usan cámaras CCD con un ratio discreto, donde variar éste, implica variar el desplazamiento entre fotogramas, si estos desplazamientos son demasiado grandes, los métodos de gradiente fallarán (de la misma manera que se rompe la continuidad del volumen espacio-temporal). Aunque se puede usar un suavizado espacial *anti-aliasing* para evitar el *aliasing* temporal [CHR98], [ZHA01], todo esto tiene la contrapartida de degradar la información espacial. Por lo tanto, para una resolución espacial dada, se tiene que muestrear a la mínima frecuencia temporal [YAC99].

Por otra parte, es bastante común que los algoritmos de flujo óptico sigan una arquitectura funcional, como la mostrada en la figura anterior, efectuándose un proceso jerárquico.

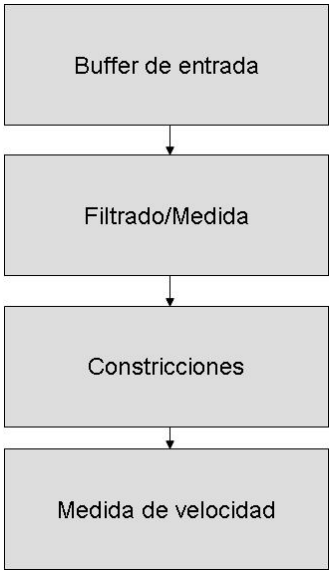


Figura 2.1. Arquitectura funcional básica para el cálculo de flujo óptico, encontrada en muchos algoritmos.

Previamente, se filtra la secuencia de imágenes o el *buffer* temporal para extraer medidas básicas a través de convoluciones, transformada rápida de Fourier (FFT), extracción de patrones, operaciones aritméticas, etc..

Las medidas después se recombinarán a través de varios métodos para llegar a una estimación básica de velocidad (usualmente incompleta y deficiente en estas etapas iniciales). Posteriormente, la estimación final de flujo se realiza imponiendo una serie de restricciones sobre las medidas y resultados. Éstas se generan a través de hipótesis sobre la naturaleza del flujo o movimiento (como las restricciones de un sólido rígido), pero incluso con estas restricciones la información recuperada no suele ser suficientemente buena para obtener una solución única para el campo de flujo óptico.

En el capítulo I se explicó que el flujo óptico es el movimiento estimado desde los cambios observables en el patrón de luminancia sobre el tiempo y que en el caso de una situación de movimiento no observable como, por ejemplo, una esfera rotando con el mismo brillo, el flujo óptico estimado será cero en todo lugar, incluso aunque las velocidades actuales sean no nulas. Otra contrapartida viene dada por la no existencia de un movimiento único de la imagen para explicar un cambio en el brillo observado, por consiguiente la medida de movimiento visual es muchas veces imposible y siempre tiene que tener asociada un numero de interpretaciones físicas.

2.2. Modelos de movimiento elementales

A pesar de las dificultades en la recuperación del flujo, los sistemas biológicos funcionan sorprendentemente bien. De la misma manera que estos sistemas tienen mecanismos especializados para percibir color y estereopsis, también tienen mecanismos dedicados al movimiento visual [ALB93]. Al igual que en otras áreas de investigación, en visión artificial se forman modelos de esos sistemas naturales para formalizar las soluciones bioinspiradas. Gracias a estudios psicofísicos y neurofisiológicos ha sido posible construir modelos que extraen el movimiento de las secuencias de imágenes, caracterizándose estos modelos biológicos por ser complejos y deficientemente diseñados para funcionar a gran velocidad o incluso, en tiempo real.

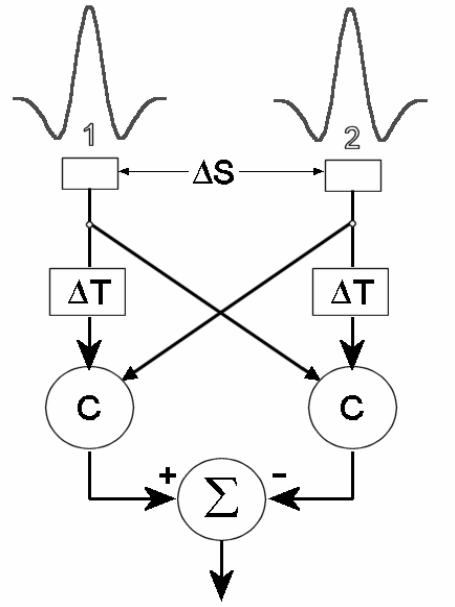


Figura 2.2. El detector elemental de Reichardt.

Uno de los primeros modelos bioinspirados basado en sensores visuales fue propuesto por Reichardt [REI61]. El detector consiste en un par de campos receptivos, (figura 2.2) donde la señal del primero se retrasa con respecto al otro antes de combinarse de forma no lineal con una multiplicación. Los receptores 1 y 2 (mostrados como detectores de borde) son espaciados una distancia ΔS, se impone un retardo sobre cada señal y se combina el resultado en C mediante una multiplicación. Al final, el resultado de la primera mitad del detector se substrae de la segunda y después se estima lo que contribuye cada una a incrementar la selectividad direccional.

Los sensores mostrados en la figura 2.2 son detectores laplacianos, aunque se puede usar cualquier filtro espacial o incluso un detector de rasgos/características.

Uno de los principales inconvenientes de este detector, es que la correlación es dependiente del contraste, además tampoco se puede recuperar velocidad directamente, necesitando bancos de detectores calibrados a varias velocidades y orientaciones, siendo su interpretación cuando menos ambigua. Sin embargo, a pesar de todas estas desventajas, el detector de Reichardt puede ser aplicado fácilmente por sistemas biológicos y se usa satisfactoriamente para explicar el sistema visual de algunos insectos. Este detector continúa siendo empleado como punto de partida de modelos más sofisticados de visión [BEA99], [ZAN96], pudiendo implementar detectores de baja resolución en el sensor CCD con tecnología VLSI [ARI96].

2.2.2. Detección de cambios

Si se considera el caso simple de una región segmentada moviéndose respecto a regiones estáticas, se podría tener una imagen binaria indicando regiones de movimiento o de no movimiento como etapas de partida de otros análisis posteriores. El proceso podría parecer fácil intuitivamente (simplemente se están buscando cambios en la intensidad de la imagen por encima de un umbral, que se supone que causa el movimiento de un objeto en el campo visual). Sin embargo, el número de falsos positivos que parten de las fuentes, como el ruido de los sensores, movimiento de cámaras, sombras, efectos ambientales (lluvia, reflejos, etc.), oclusiones y cambios de iluminación hacen muy difícil la detección robusta de movimiento. De nuevo aquí, se tiene otra oportunidad para aprender de los sistemas biológicos, los cuales a pesar de ser muy sensibles a movimiento son al mismo tiempo muy robustos al ruido y a efectos visuales no interesantes.

En la parte superior de la figura 2.3 se puede observar la escena del taxi de Hamburgo, particularizada para los fotogramas 1 y 8, respectivamente. En dicha figura (inferior izquierda) se observan la diferencia entre ambos fotogramas y la segmentación binaria (inferior derecha) de las diferencias [ROS98]. Esta técnica se emplea en situaciones en las que la detección del movimiento es un evento que debe ser tenido en cuenta para un uso posterior.

Actualmente, los requisitos de estimación para estos algoritmos son mínimos llegándose a un resultado satisfactorio con poco más que un buffer de entrada, aritmética de señales y algunas estadísticas robustas, ya que los sistemas supervisores tienen que ser muy

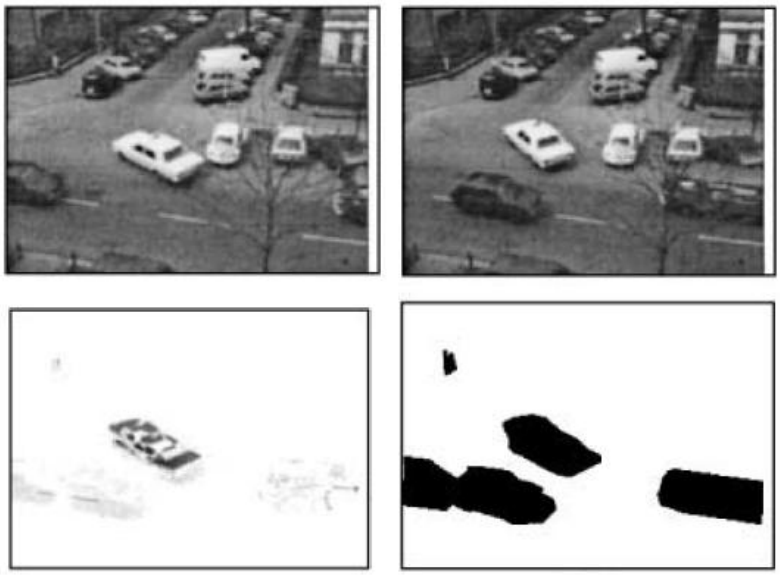


Figura 2.3. Escena del taxi de Hamburgo
(La secuencia de entrada puede ser descargada de la siguiente dirección:
ftp://ftp.csd.uwo.ca/pub/vision/TESTDATA/)

sensitivos y no se dispone normalmente de gran potencia de cálculo [ROS98], [PAJ03].

2.2.2. Métodos de correspondencias

Cuando las aproximaciones diferenciales están sujetas a errores debido al no cumplimiento del teorema del muestreo [NIQ06] o a cambios de iluminación inconvenientes, se hace necesario la aplicación de otro tipo de estrategias. Los métodos de correlación o emparejamiento de patrones son los más intuitivos para recuperar velocidad y dirección del movimiento, trabajan seleccionando características en un fotograma de la secuencia de imágenes y después buscando estas mismas características en el siguiente (figura 2.4). Los cambios en la posición indican movimiento en el tiempo, es decir, velocidad.

Estos algoritmos se caracterizan por una ejecución muy lenta debido a su búsqueda exhaustiva y funcionamiento iterativo, requiriendo normalmente una cantidad de recursos prohibitiva.

Figura 2.4. Estimación de Movimiento usando plantilla de correspondencia. **(Superior)** Una plantilla se selecciona en un fotograma, de forma que se intenta emparejar esta plantilla en el siguiente fotograma **(Inferior)** dentro de su ventana de búsqueda.

Si la imagen es de tamaño M^2, la plantilla de búsqueda es de tamaño N^2, y la ventana de búsqueda es de tamaño L^2 entonces la estimación total de complejidad de cómputo requerida sería del orden de $M^2N^2L^2$. Por ejemplo, para una imagen de 512×512 píxeles, una plantilla y ventana de búsqueda de tamaño 10×10, se requerirían ejecutar unas 2.6 billones de operaciones. La tendencia actual es intentar disminuir el dominio de búsqueda [OH00], [ANA93], [ACC98], aunque todavía se mantiene una necesidad de recursos demasiado alta.

Uno de los más modelos de aplicación más comunes usados es la codificación de video en tiempo real [ACC98], [DEF95]. incrementándose últimamente la cantidad de esfuerzo dedicado a investigación en este tipo de algoritmos. Una de las claves de la compresión de video es la similitud entre imágenes temporalmente adyacentes dentro de una secuencia, necesitándose menos ancho de banda para transmitir las diferencias entre fotogramas que para transmitir la secuencia entera, incluso se puede reducir aún más el volumen de datos transmitidos, si se conoce a priori el movimiento y las deformaciones que se necesitan para pasar de un fotograma al siguiente.

Este tipo de algoritmos se pueden clasificar de forma más profunda, existiendo dos aproximaciones destacadas:

- Correlación, como una función de 4 variables, dependiendo de la posición de la ventana y la del desplazamiento, con salida normalizada entre 0 y 1 e independiente de los cambios de iluminación.
- Minimización de la distancia, cuantificando la no similitud entre regiones. Muchas de sus optimizaciones han ido en la línea de reducir el espacio de búsqueda [OH00] e incrementar su velocidad [ACC98].

Adelson y Bergen [ADE85] argumentan en contra de la evidencia biológica de este tipo de modelos, ya que no son capaces de hacer predicciones sobre estímulos complejos (por ejemplo barras verticales posicionadas aleatoriamente), para los cuales, los observadores experimentales perciben diversos movimientos en diferentes posiciones.

A su favor se puede argumentar, que estas técnicas son simples, llevan muchos años investigándose y dominan en entornos industriales para inspección y control de calidad.

Una de sus principales ventajas es su capacidad de trabajo en entornos donde los desplazamientos entre fotogramas son largos (más que unos pocos puntos), aunque esto requiere el procesamiento de largos espacios de búsqueda.

2.2.3. Métodos espacio-temporales

El movimiento puede ser considerado como una orientación en el diagrama espacio-tiempo. Por ejemplo, en la figura 2.5 se presenta el ejemplo de una barra vertical moviéndose continuamente desde izquierda a la derecha, muestreada 4 veces en el tiempo.

Si se examina el volumen espacio temporal, se puede observar el movimiento de la barra respecto al eje temporal y una barra estacionaria paralela a dicho eje orientada un ángulo determinado, que será el que indique la medida de movimiento.

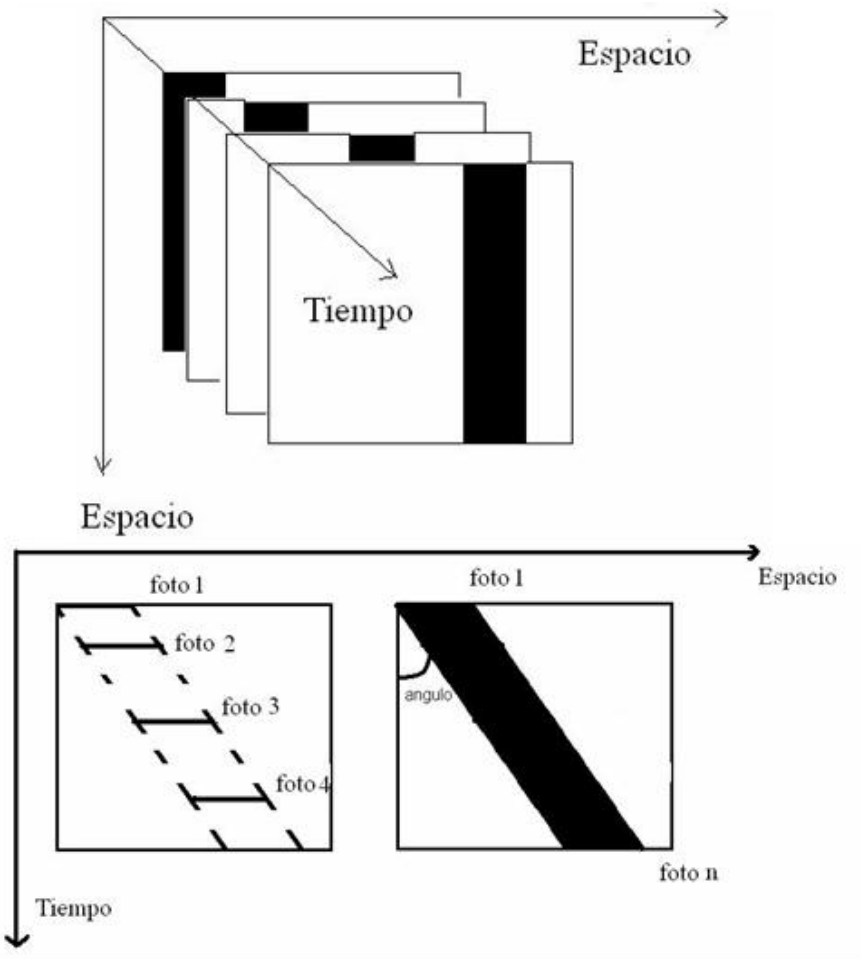


Figura 2.5. (Parte superior) Una secuencia de 4 fotogramas de una barra negra moviéndose hacía la derecha en el espacio-tiempo.**(Parte inferior).** Movimiento, como orientación en el espacio-tiempo, (plano x-t), donde el ángulo aumenta con la velocidad. Esta orientación indica movimiento

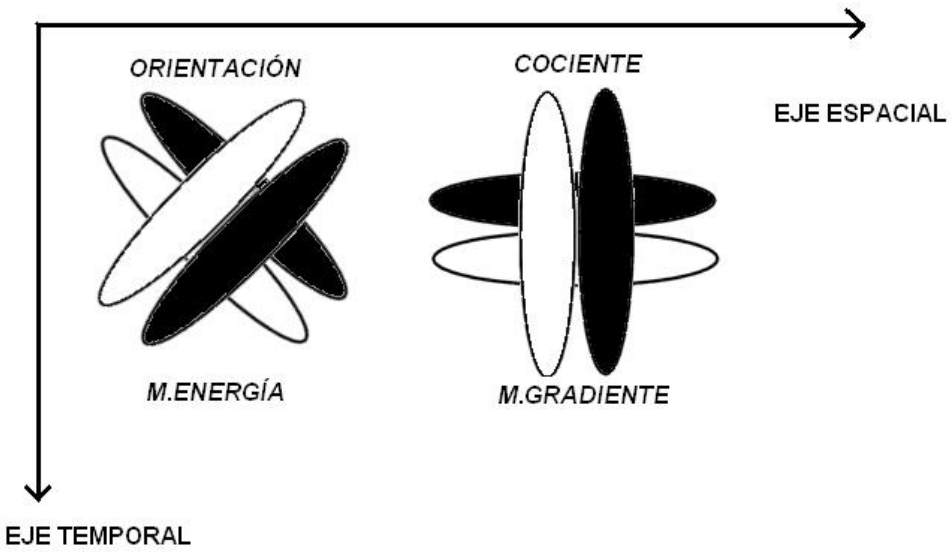


Figura 2.6. Modelos de filtros espacio temporales de bajo nivel. Las elipses representan los lóbulos negativos y positivos del filtro espacio-temporal.

La orientación de la estructura espacio temporal puede ser recuperada a través de filtros de bajo nivel, existiendo actualmente dos estrategias dominantes, el modelo de gradiente y el modelo de energía (figura 2.6). En el modelo de gradiente se usa el cociente de un filtro espacial y uno temporal como medida de velocidad, sin embargo en el modelo de energía se emplean bancos de filtros orientados en el espacio-tiempo. Ambos modelos usan un tipo de filtros bastante comunes en el sistema visual humano [ADE85], [YOU93]. La controversia sobre cual es el esquema adoptado por el sistema visual humano, sigue abierta, existiendo incluso pasarelas para ir de un modelo a otro, debido a que es posible sintetizar los filtros orientados en el modelo de energía través de filtros espacio temporales separables [FLE89], [HUA95].

También es interesante resaltar que un análisis de componentes independientes (ICA) señala que este tipo de filtros espaciales son los que abarcan la gran mayoría de componentes de la estructura de la imagen [HAT98].

En el modelo de gradiente, la principal hipótesis de trabajo es la conservación de la intensidad a lo largo del tiempo [HOR81], consistiendo ésta en que sobre periodos cortos de tiempo, los cambios de intensidad se deben sólo a translación y no a cambios en iluminación, reflectancia, etc.. La derivada total de la intensidad de la imagen con respecto al tiempo es cero en cada punto del espacio-tiempo. Por consiguiente si se define la intensidad de la imagen como $I(x,y,t)$, se tiene que:

$$dI(x,y,t)/dt=0 \tag{2.1}$$

Diferenciando por partes se obtiene la llamada ecuación de constricción de movimiento (MCE, usando notación anglosajona) fundamental en los modelos de gradiente.

$$\frac{\partial I}{\partial x}\frac{dx}{dt}+\frac{\partial I}{\partial y}\frac{dy}{dt}+\frac{\partial I}{\partial t}\frac{dt}{dt}=0 \tag{2.2}$$

$$\frac{\partial I}{\partial x}u+\frac{\partial I}{\partial y}v+\frac{\partial I}{\partial t}=0 \tag{2.3}$$

Donde $u = dx/dt$ y $v = dy/dt$. Los parámetros (x, y, t) se omiten para claridad de notación. Puesto que hay sólo una ecuación con dos incógnitas (dos componentes de velocidad desconocidas), sólo se puede recuperar la componente de velocidad v_n, que está en la dirección del gradiente de luminancia.

$$v_n=\frac{-\partial I/\partial t}{\sqrt{\left(\partial I/\partial x\right)^2+\left(\partial I/\partial y\right)^2}} \tag{2.4}$$

Existen varios problemas asociados a la ecuación de constricción del movimiento (MCE), puesto que es una ecuación con dos incógnitas (u y v), y por consiguiente está incompleta para poder estimar el flujo óptico. Usando la ecuación (2.3) de forma aislada sólo se puede aspirar a obtener una combinación lineal de componentes de velocidad, este efecto, por otra parte, coincide plenamente con el problema de apertura, mencionado en el capítulo I.

Un segundo problema se manifiesta si los gradientes espaciales I_x o I_y llegan a ser muy pequeños o cero, en cuyo caso la ecuación llega a estar mal condicionada y la velocidad estimada tiende asintóticamente a infinito. Además, de que la realización estable de las derivadas espaciales, ya es algo de por sí problemático, aplicando una convolución de filtros diferenciales, como operadores de Sobel, Prewitt o diferencia de Gaussianas. Como se están usando derivadas numéricas de una función muestreada, éstas se ajustarán mejor para intervalos espacio-temporales pequeños, estando siempre presente el problema del *aliasing* al no muestrear suficientemente bien el espacio-tiempo, especialmente en el dominio del tiempo, como se ha comentado de forma previa.

Existen varias técnicas de filtrado para solventar este problema, como por ejemplo, un filtro espacio-temporal de paso baja como apunta Zhang [ZHA03].

Idealmente, la tasa de muestreo debería ser suficientemente alta para reducir todos los movimientos a menos de un punto/fotograma, para que la derivada temporal esté bien condicionada [NIQ06]. Los filtros diferenciales espacio temporales que se usan para implementar algoritmos de gradiente se parecen razonablemente a los encontrados en el córtex visual, aunque no existe un consenso sobre la forma óptima desde el punto de vista de la funcionalidad [YOU01]. Una ventaja de las ventajas de los modelos de gradiente sobre los de energía, es que éstos proporcionan una velocidad explicita de la combinación de respuestas de filtros y los otros, sin embargo, dan una población de soluciones.

2.3. Mejora de la ecuación de constricción de movimiento (MCE)

Se ha visto que la ecuación de constricción de movimiento (MCE) tiene varias anomalías que hay que solventar para poder estimar adecuadamente flujo óptico. En la literatura existe un amplio abanico de métodos para mejorarla. Muchos de ellos aplican restricciones adicionales para resolver las dos componentes de velocidades *u* y *v*, recogiendo más información (con más imágenes o extrayendo más información de cada imagen) o aplicando restricciones físicas para generar MCEs adicionales:

- Aplicando filtros múltiples [MIT87], [SOB91], [ARN93], [GHO97].
- Usando integración en una vecindad [LUC81], [URA88], [SIM91].
- Usando imágenes multiespectrales [GOL97].

Aparte de usar la intensidad de la imagen en la ecuación MCE, se pueden usar otras funciones. Estas funciones se pueden generar fácilmente a partir del patrón de entrada a través de algún tipo de filtro, siempre que se disponga más de una función que satisfaga la MCE (sea invariante a la MCE), el flujo óptico se puede estimar.

Con dos funciones, la ecuación de gradiente puede ser resuelta analíticamente y con más de dos, se puede construir un conjunto de ecuaciones sobrecondicionadas, según indica la expresión (2.5).

$$\begin{aligned} &\frac{\partial I_1}{\partial x}u+\frac{\partial I_1}{\partial y}v+\frac{\partial I_1}{\partial t}=0\\ &\frac{\partial I_2}{\partial x}u+\frac{\partial I_2}{\partial y}v+\frac{\partial I_2}{\partial t}=0\\ &\vdots \qquad\quad \vdots \qquad\quad \vdots\\ &\frac{\partial I_n}{\partial x}u+\frac{\partial I_n}{\partial y}v+\frac{\partial I_n}{\partial t}=0 \end{aligned} \tag{2.5}$$

El sistema de ecuaciones anterior puede ser reescribirse como una matriz a resolver a través de un mecanismo de mínimos cuadrados lineales [SIM91] o mínimos cuadrados totales [WEB91].

$$A \bullet \overline{v} = b \qquad \overline{v} = A^{-1} \bullet b \tag{2.6}$$

donde *A* y *b* resultan:

$$A=\begin{bmatrix} \frac{\partial I_1}{\partial x} & \frac{\partial I_1}{\partial y}\\ \frac{\partial I_2}{\partial x} & \frac{\partial I_2}{\partial y}\\ \vdots & \vdots\\ \frac{\partial I_n}{\partial x} & \frac{\partial I_n}{\partial y} \end{bmatrix} \qquad b=-\begin{bmatrix} \frac{\partial I_1}{\partial t}\\ \frac{\partial I_2}{\partial t}\\ \vdots\\ \frac{\partial I_n}{\partial t} \end{bmatrix} \tag{2.7}$$

Este método requiere que la matriz *A* sea invertible (no singular), lo cual no ocurrirá en muchas regiones de la imagen donde haya falta de contraste, como, por ejemplo, una retícula unidimensional.

Existen por tanto, diferentes métodos generales para restringir la MCE y mejorar las medidas de flujo óptico.

- Llevando a cabo optimizaciones globales, por ejemplo, suavizado [HOR81], [NAG83], [HEI93].
- Restringiendo el flujo óptico a un modelo especifico conocido, por ejemplo el modelo afín [LIU97], [ONG99], [FLE00].
- Usando métodos multiescala de forma espacial y temporal [ANA89], [WEB94], [YAC99].
- Explotando la consistencia temporal [GIA97], [GIA98].

2.3.1. Filtros múltiples

Una ecuación de constricción de movimiento aislada, no es suficiente para estimar el flujo óptico, como se ha indicado de forma previa. Se propone una mejora de la misma teniendo en cuenta que sus derivadas parciales $\partial I / \partial xi$ (donde $x_1 = x$; $x_2 = y$) proporcionan soluciones adicionales a los vectores de flujo $\overline{v}$.

$$\frac{\partial}{\partial x_i}\left(\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t}\right) = 0 \tag{2.8}$$

Nagel [NAG83] fue el pionero en usar este método, usando derivadas de segundo orden, de hecho, el operador diferencial es uno de tantos que podría usarse para generar múltiples MCEs. Normalmente estos operadores se aplican numéricamente mediante convoluciones y dado que la convolución es un operador lineal, se puede escribir:

$$\begin{gathered} O_i \otimes \left(\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t}\right) = 0 \\ \left(\frac{\partial O_i}{\partial x} \otimes I\right)u + \left(\frac{\partial O_i}{\partial y} \otimes I\right)v + \left(\frac{\partial O_i}{\partial t} \otimes I\right) = 0 \end{gathered} \tag{2.9}$$

Donde, O_i es un filtro arbitrario de convolución (como una Gaussiana, su diferencia, media, Laplaciana, etc.). Este proceso funciona debido a que la convolución no cambia la orientación de la estructura espacio-temporal. Por otra parte, es importante usar filtros que sean linealmente independientes, en otro caso, las MCEs producidas estarán degeneradas y no se habrá ganado nada. Los filtros y sus diferenciales pueden ser estimados previamente para ganar eficiencia, y debido a la localidad de los operadores, implementar de forma masivamente paralela estas estructuras.

Si se usan dos filtros tales que generen dos ecuaciones, se tendrá suficiente información para obtener las componentes de velocidad a condición de que las medidas no sean degeneradas, como se ha dicho y que la imagen sea también no degenerada (por ejemplo, una retícula unidimensional).

Sobey y Srinivasan, [SOB91] aportan un modelo de gradiente basado en esta arquitectura, aunque matizan que el operador O_i puede ser diferente para cada término de la ecuación 2.9. Resuelven la MCE usando dos filtros espaciales que conforman a su vez dos

ecuaciones a resolver analíticamente. Sin embargo el problema de usar dos MCEs es que la solución analítica es un cociente, pudiéndose presentar el problema de que el denominador se haga nulo en regiones con baja estructura espacial, ellos estiman de forma previa si la solución estará bien condicionada o no, pero esta estimación no soluciona el problema.

Muchos especialistas, generan más de dos ecuaciones de constricción del movimiento en el marco de un sistema lineal, como Weber y Nagel [WEB94], [NAG83], que convolucionan la secuencia de entrada con un conjunto de derivadas de Gaussianas de primer y segundo orden en diversas orientaciones. Cada imagen prefiltrada I_i se diferencia con respecto a *x, y, t* para dar un conjunto de MCEs. Se usa el método total de mínimos cuadrados para determinar las componentes de la velocidad, rechazando los resultados de las escalas de filtros para las que falla el modelo.

Este modelo usa una etapa de filtrado similar a la del Modelo Multicanal de Gradiente (McGM) abordado en este trabajo de investigación. El tiempo de estimación para una estación de trabajo Sparc, es 6 minutos/fotograma [SPA03], que se reduce a 20 seg/fotograma usando una implementación masivamente paralela con un "CM5 Thinking Machine" [CM502] con 128 procesadores.

La mayoría de las técnicas existentes de flujo óptico dependen de la conservación de intensidad o de valores derivados de ésta. Ghosal y Mehrotra [GHO97] presentan un algoritmo basado en momentos ortogonales de Zernike invariantes frente a rotación y escalado (normalmente utilizados para describir aberraciones ópticas) de intensidad de la imagen. Estos momentos M^i basados en integrales, son robustos para ruido no correlacionado, cumpliendo la siguiente propiedad

$$\frac{\partial M^i}{\partial x}u + \frac{\partial M^i}{\partial y}v + \frac{\partial M^i}{\partial t} = 0 \qquad (2.10)$$

En este algoritmo, se escoge un número de momentos *m* con respecto a una vecindad sobre un área n^2, produciendo un sistema de ecuaciones lineales sobrecondicionado (mn^2), posteriormente se asume un modelo de flujo afín y se resuelve el sistema usando una descomposición de valores singulares (DVS), para las componentes de la velocidad. El algoritmo es relativamente lento y debido a la etapa de integración que tiene que extraer los momentos de Zernike, la ejecución total de la rutina DVS tarda 71 segundos por fotograma con una resolución de 233×256 puntos.

Esta filosofía de uso de filtros múltiples tiene una consistencia biológica, puesto que los sistemas biológicos son masivamente paralelos y apropiados para este tipo de operaciones, similares muchas de ellas a las encontradas en el sistema visual humano. Las múltiples proyecciones desde el núcleo geniculado lateral (LGN) y las capas de células en el córtex visual del cerebro humano pueden actuar como filtros locales. Con estos modelos, se intenta replicar una arquitectura eficiente encontrada en la conexión del córtex visual, aunque la manera en que estas respuestas están recombinadas para proporcionar una percepción de movimiento tan robusta, es todavía un área de investigación muy activa.

Cabe destacar, por otra parte, que la tarea de inversión de matrices para solucionar ecuaciones múltiples es biológicamente menos plausible, considerándose otras alternativas que se estudiarán en el capítulo III de esta memoria.

2.3.2. Integración de vecindad

Es posible usar información almacenada sobre una región local Ω para generar ecuaciones de restricción de movimiento extras [LUC81], [SIM91]. Se asume pues, que el movimiento consiste en una traslación pura dentro de una región local, donde estas restricciones se modelan con una matriz de pesos *W(x,y)* de forma que se sitúen los resultados centrados dentro de una región local como, por ejemplo, siguiendo una distribución Gaussiana. Se reescribe entonces, la MCE como un problema de minimización.

$$\varepsilon = \sum_{x,y\in\Omega} W^2(x,y)\left[I_x u + I_y v + I_t\right]^2 \tag{2.11}$$

El término de error que aparece en la expresión (2.11) se minimiza o bien se resuelve el conjunto de ecuaciones generadas mediante métodos numéricos.

2.3.3. Métodos multiespectrales

Si se trabaja con imágenes policromadas, se pueden generar diferentes funciones de brillo. Por ejemplo, los planos de color rojo, verde y azul de una cámara estándar se pueden tratar como tres imágenes independientes, produciendo tres MCEs para resolver. Una contrapartida de este método es que los planos de color normalmente están correlacionados, (un hecho, por otra parte, que es explotado por la mayoría de algoritmos de compresión).

En estas situaciones, el sistema lineal de ecuaciones se puede degenerar, de manera que no se garantiza finalmente que el coste extra en computación lleve a una mejora en la calidad del flujo.

Una variante de este tipo de métodos, es que aplican adicionalmente invarianza respecto a pequeños desplazamientos y cambios de iluminación, fundamentándose estas medidas en el cociente de diferentes planos de color (funciones de sensibilidad espectral) tales como las comúnmente usadas RGB o HSV. Usando esta última variante se obtienen apreciables mejoras con respecto a usar sólo un plano RGB [GOL97].

2.3.4. Optimización global

Debido a la ausencia de información y estructura espacial de la imagen, no es fácil la estimación de un campo de velocidades suficientemente denso. Para corregir este problema, se aplican varias restricciones, como por ejemplo, que los puntos cercanos entre sí, se mueven de una manera similar. La filosofía general es que el campo de flujo original, una vez estimado, se regularice de forma iterativa con respecto a una restricción de suavizado.

La primera restricción fue propuesta por Horn y Schunk [HOR81], donde el término suavizado de la velocidad viene dado por la ecuación (2.12):

$$\varepsilon^2_{Suavizado} = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \tag{2.12}$$

El error ε en la MCE después se minimiza añadiendo la siguiente restricción de suavizado:

$$\varepsilon^2 = \iint \left[\left(I_x u + I_y v + I_t \right)^2 + \lambda \varepsilon^2_{suavizado} \right] dx\, dy \tag{2.13}$$

en la ecuación 2.13, λ expresa la importancia relativa del termino de suavizado, I_x es la derivada espacial en la dirección de x, I_y en la dirección de y, I_t es la derivada temporal. El algoritmo es capaz de rellenar regiones con pequeños gradientes y proporciona resultados libres de ruido, sin embargo, tiene el inconveniente del grado de emborronamiento sobre las discontinuidades de movimiento. Nagel [NAG83] introduce las derivadas de segundo orden y una restricción que atenúe el suavizado en la dirección donde el gradiente trabaja bien, de

manera que este último mecanismo permite activar el suavizado sólo donde sea necesario, en la dirección tangencial al gradiente, evitándolo en los bordes de intensidad.

Se ha desarrollado activamente la teoría psicofísica de la detección de bordes en imágenes estáticas y se puede incorporar todo el conocimiento anterior sobre el dominio del tiempo, a las medidas del flujo. El método de Hildreth [HIL84] se basa en la Laplaciana de la Gaussiana, para determinar la localización de los bordes y contornos de cambios significativos de intensidad. Se estiman vectores de movimiento sobre los bordes producidos por un filtrado Laplaciano y después se minimiza la variación en velocidad a lo largo de los contornos para resolver el problema de la apertura.

El flujo óptico resultante de las restricciones globales, es bastante robusto debido a la combinación de resultados, siendo también favorecedor para el ojo humano. Dos de los mayores inconvenientes son: su naturaleza iterativa, donde a menudo se requiere cantidades muy grandes de tiempos de computación y que las discontinuidades de movimiento no se manejan correctamente, de modo que se producen resultados erróneos en las regiones que rodean los bordes de movimiento. Para subsanar estas últimas deficiencias se proponen otras técnicas que usan estadísticas globales, como las cadenas de Markov aleatorias [HEI93], que sí que son capaces de preservar el movimiento en los contornos.

2.3.5. Modelos de movimiento

En todas las técnicas de estimación se imponen grandes restricciones sobre una vecindad donde se asume flujo constante. Para cumplir con este requisito, esta vecindad tiene que ser lo más pequeña posible, pero al mismo tiempo tiene que ser suficientemente grande para obtener información y evitar el problema de la apertura. Por todo ello, se necesita una solución de compromiso.

Una gran variedad de modelos usan estimaciones relacionadas con esta vecindad, tales como mínimos cuadrados. Si se usa una función objetivo cuadrática, se asumen inherentemente errores residuales de tipo Gaussiano. Pero si se tienen múltiples movimientos en la vecindad, estos errores ya no pueden considerarse Gaussianos, incluso más, si estos errores fueran independientes (situación bastante probable) la distribución del error puede ser modelada como una bimodal.

Estas consideraciones muestran, que de hecho ya se ha aplicado un modelo de flujo de forma particular, a saber, el más simple posible, de flujo constante y error tipo Gaussiano dentro de la vecindad.

Se procede a continuación a comentar diferentes modelos aproximados de flujo que puedan incorporarse al abanico de técnicas que se están exponiendo. Estas aproximaciones modelan variaciones espaciales además de movimientos múltiples. Las técnicas de integración de vecindad, como se ha dicho, asumen que el movimiento de la imagen en regiones locales es puramente translacional, por lo tanto con modelos más elaborados (como el modelo afín) se puede ampliar el rango de movimientos además de proporcionar restricciones adicionales.

Estos métodos reformulan la MCE con una función de error ε que se resolverá o minimizará por mínimos cuadrados [CAM92], [BER93], [GUP97], [GIA97]:

$$\varepsilon = \sum_{x,y \in \Omega} \left(I_x u + I_y v + I_t \right)^2 \tag{2.14}$$

En estos casos, la ecuación MCE se asume que no se anula y es una función que se tiene que minimizar sobre alguna vecindad local Ω. Esta localidad puede tener más componentes que las de translación, construyendo las componentes de velocidad con diferente complejidad según el modelo a usar, por ejemplo, mediante una translación uniforme de dos parámetros, o una transformación afín de seis. Bergen [BER92] ha trabajado en modelos de movimiento que tengan en cuenta rotación, escalado y translación.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.15}$$

El error de la MCE en ecuación (2.14) se minimiza para los parámetros *a*, *b* y *c* en la ecuación 2.15. Estos parámetros expresan el flujo local en términos de translación, rotación y divergencia. Giaccone [GIA97] ha desarrollado un algoritmo donde se combina la segmentación en primer y segundo planos (por ejemplo, los actores dentro de un escenario) y la segmentación de regiones con un modelo de movimiento rígido para restringir el sistema de ecuaciones. El desplazamiento en una región pequeña se modela por una transformación de ocho parámetros según indica la ecuación (2.16).

$$u = (a_0 x^2 + a_1 xy + a_2 x + a_3 y + a_4)$$
$$v = (a_0 xy + a_1 y^2 + a_5 x + a_6 y + a_7) \quad (2.16)$$

Esta transformación a su vez, emana de un modelo de movimiento global, con la asunción de que los cambios en la posición de las coordenadas entre fotogramas sean pequeños. Gupta y Kanal [GUP97] primero integran la MCE sobre el volumen espacio-temporal *V*:

$$\int_V \left(uI_x + vI_y + I_t \right) dV = 0 \quad (2.17)$$

a partir de la ecuación (2.17) se utiliza el teorema de la divergencia de Gauss para transformar esta expresión en una integral de superficie:

$$\int_S uIdydt + \int_S vIdxdt - \int_V I(u_x + v_y)dxdydt + \int_S Idxdt = 0 \quad (2.18)$$

añadiéndose además, una restricción por la cual el flujo es lineal en una vecindad espacio-temporal pequeña:

$$u(x, y, t) = a_1 x + a_2 y + a_3 t + a_4$$
$$v(x, y, t) = b_1 x + b_2 y + b_3 t + b_4 \quad (2.19)$$

Tras las operaciones realizadas en las expresiones (2.17) a (2.19), se ha reducido el problema a determinar los coeficientes de flujo a_n y b_n. Estas ecuaciones se sustituyen por *u* y *v* dentro de la integral de la ecuación (2.17) y se resuelve por mínimos cuadrados. La robustez de este algoritmo se fundamenta en que la etapa de integración evita la necesidad de usar derivadas espaciales y sus consiguientes problemas asociados. Sin embargo a pesar de la ausencia de filtrado, la integrales consumen ingentes cantidades de computación.

El modelo afín de flujo óptico puede ser ampliado empleando un modelo de movimiento completo, que parametriza un movimiento tridimensional proyectado en perspectiva, para estimar tanto el flujo óptico como otros parámetros interesantes [LIU97].

2.3.6. Métodos multiescala

Los desplazamientos largos de entre fotogramas originan que los métodos de gradiente se comporten de forma inapropiada, dado que las secuencias de imágenes se muestrean

insuficientemente o que las medidas de la derivada temporal son inexactas. Como solución se pueden usar filtros espaciales más grandes [CHR98]. El uso de una pirámide multiescala Gaussiana permite tratar movimientos largos entre fotogramas y rellenar los huecos en regiones extensas donde la textura es uniforme, de modo que las estimaciones de movimiento a escala gruesa se usan como fuentes para una escala más fina [ZHA01], [MAH99].

El uso de la multiescala temporal [YAC99] también permite la estimación exacta de una horquilla de diversos movimientos, aunque este método requiere usar una tasa de muestreo suficientemente alta para reducir los movimientos aproximadamente a un punto/fotograma.

2.3.7. Consistencia temporal

Los esquemas abordados hasta ahora, tratan el cálculo del flujo óptico como un problema independiente para cada fotograma, sin ningún tipo de realimentación (los resultados del movimiento de un fotograma no informan del análisis de los siguientes). Giaccone y Jones [GIA98] han ideado una arquitectura realimentada capaz de tratar movimientos múltiples y segmentación de regiones en movimiento, mediante el uso de un método de mínimos cuadrados para un modelo afín, con la función de error expresada en la ecuación (2.20).

$$\varepsilon = \sum_{x \in \mathbb{R}} e(\overline{x}, u_t(\overline{x}))^2$$
$$e(\overline{x}, u_t(\overline{x})) = I_t(\overline{x}) - I_{t-1}(\overline{x} - u_t(\overline{x})) \tag{2.20}$$

$\overline{x} = (x, y)$ y el término $I_{t-1}(\overline{x} - u_t(\overline{x}))$ es la imagen previa en el instante *t*-1 proyectado hacia delante a través de la imagen actual en el instante t usando la información de la velocidad $u_t(x)$. Este algoritmo ha resultado muy robusto para un rango de velocidades determinado, además, funciona bien si se compara con los modelos afines. El costo de cálculo es de alguna manera sobrecargado por la generación de una imagen proyectada, necesitándose para tamaños PAL unos 40 segundos/imagen.

Sin embargo, esta restricción de consistencia temporal sólo se usa de forma muy esporádica actualmente. Los objetos en el mundo real tienen que obedecer leyes físicas de movimiento como la inercia y la gravedad, de forma que exista una capacidad de predicción en cuanto a su comportamiento, siendo cuando menos, sorprendente que la mayor parte de

algoritmos de flujo no implementen una realimentación basada instantes anteriores. (*v.* capítulo VI, referente a líneas futuras).

Para este propósito se usa el hecho de que es posible generar una ecuación de restricción adicional desde el campo de velocidades para utilizarse en la siguiente iteración, tratando el problema como un fenómeno evolutivo.

El uso de modelos probabilísticos o Bayesianos [SIM91] puede ser una alternativa para usar información del mundo real y actualizar resultados de estimaciones previas integrando información temporal.

2.4. Modelos de energía de movimiento

Se ha visto que la percepción del movimiento se puede modelar como una orientación en el espacio-tiempo, donde los métodos de gradiente extraen esta orientación a través del cociente de filtros orientados. Los modelos de energía de movimiento o basados en frecuencia son similares en muchos sentidos a los modelos de gradiente, ya que ambos sistemas usan bancos de filtros para obtener esta orientación espacio-temporal, por consiguiente, el movimiento. La diferencia principal radica en que los filtros usados en modelos de energía, están diseñados para responder a orientaciones específicas espacio-temporales, más que para un cociente de filtros.

El diseño de filtros orientados espacio-temporales, se lleva a cabo normalmente en el espacio de frecuencia, debido a la siguiente relación entre un patrón bidimensional que se traslada y su transformada de Fourier.

$$\begin{aligned} I(\bar{x},t) &= I_0(\bar{x}-\bar{v}t) \\ \hat{I}(\bar{k},\omega) &= \hat{I}_0(\bar{k}) \bullet \delta(\omega+\bar{v}\bullet\bar{k}) \end{aligned} \tag{2.21}$$

Donde $\hat{I}_0(\bar{k})$ es la transformada de Fourier del patrón estático, $\bar{x}=(x,y)$ es la posición de la imagen, $\bar{k}=(k_1,k_2)$ es la frecuencia especial, t es el tiempo, ω es la frecuencia temporal y $\bar{v}=(u,v)$ la velocidad de la imagen. La presencia de la función delta de Dirac $\delta(\omega+\bar{v}\bullet\bar{k})$ fuerza a todos los valores distintos de cero a encontrarse en un plano alrededor del origen. Por consiguiente, se puede diseñar un algoritmo de movimiento en el plano de

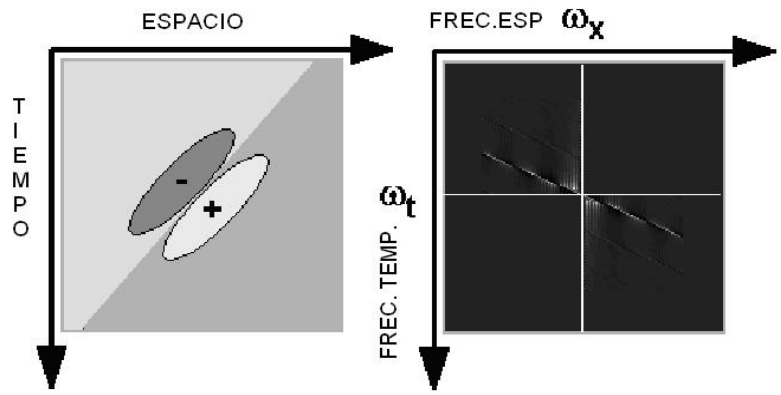


Figura 2.7. Filtro espacio temporal orientado respondiendo a un borde en movimiento.

Fourier mediante bancos de filtros sintonizados que respondan bien a planos específicos en el espacio de frecuencias.

El criterio de diseño para tales bancos de filtros es uno de las principales diferencias entre algoritmos. Un ejemplo simple se puede apreciar en la figura 2.7, donde aparece un borde unidimensional transladándose y su transformada de Fourier asociada, estando restringidas las componentes de Fourier a una línea en el espacio de frecuencias.

Diseñar un filtro con respuesta óptima a una línea en el espacio de frecuencias es equivalente a diseñar un filtro de movimiento sensible a una velocidad particular. Surgen complicaciones adicionales, si se usan filtros dependientes de la fase y la polaridad, de forma que su salida dependa de que los patrones se alineen con el campo receptivo en algún instante específico, además de la polaridad del estímulo. Por ejemplo, una barra moviéndose produciría una respuesta oscilatoria en el tiempo y si se invierte su contraste, la respuesta del filtro también saldría invertida.

Para solucionar estos problemas, se diseñan los filtros en parejas de cuadraturas. (cuadratura se refiere a la propiedad de que 2 filtros forman un par donde la máxima respuesta de un filtro cae en el mismo punto que la mínima respuesta en el otro). Formalmente, los filtros se hacen diferir en su desfase exactamente 90º [ADE85] o se pasan a través de algún mecanismo no lineal [SIM98]. Posteriormente se suman para estimar la llamada, “energía de movimiento”, ahora independiente de la fase.

Un par de filtros en cuadratura, proporciona máxima respuesta a sólo un par puntos en la línea del espacio de frecuencias, por eso se necesitan varios bancos de filtros. En la figura 2.8 se puede ver, a la izquierda, un filtro espacio-temporal orientado responder de forma óptima a un rango pequeño de velocidades (orientaciones espacio-temporales), en el centro, la respuesta parcial a velocidades no óptimas y a la derecha, la reacción del filtro cuando no hay

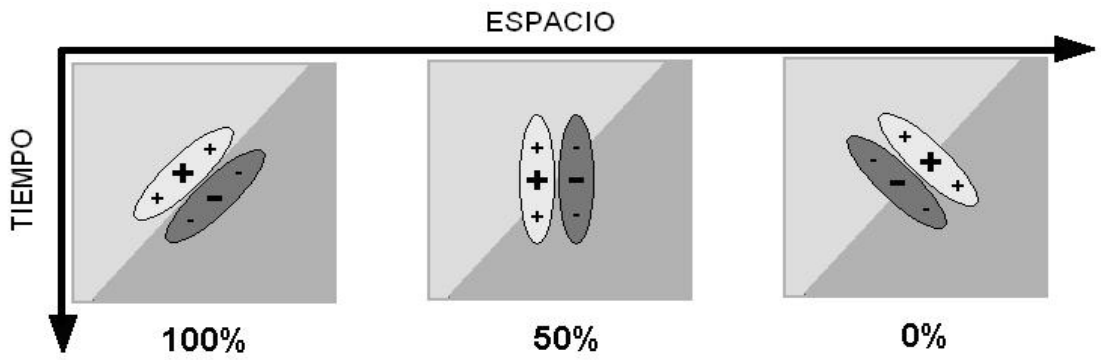


Figura 2.8. Respuestas a bancos de filtros espaciales.

respuesta. Para determinar la velocidad se requiere de forma obligada algún tipo de normalización o comparación.

El hecho de diseñar los filtros para que se ajusten de forma óptima a un rango de velocidades implica, (como se acaba de apuntar), la necesidad de disponer de suficiente cantidad de ellos para poder cubrir todo el espacio, siendo éste, el problema de la no especificidad de la velocidad. Estos filtros deben permanecer sensibles a un rango de orientaciones espaciales y escalas y esto conlleva una demanda computacional alta.

Otra contrapartida de este método es que el contraste del estimulo y la velocidad pueden ocasionar confusión, puesto que el detector responde de forma débil tanto si el contraste es bajo o si la velocidad no está cerca de su pico de sensibilidad máxima. Para tener el sistema invariante a estos problemas, se realiza una comparación de todas las respuestas, lo que supone en la práctica realizar una normalización [SIM98].

Adelson y Bergen [ADE85] presentan un sistema que usa la energía de movimiento para la estimación del flujo. Intuyendo que el futuro de este tipo de disciplinas depende en gran manera de la tecnología disponible, hacen bastante hincapié en incrementar la eficiencia computacional de los algoritmos, demostrando cómo un filtro orientado en el espacio-tiempo se puede sintetizar a partir de unos pocos filtros separables,(resultado que se usará de forma práctica en el capítulo III de la presente memoria).

Para construir una unidad de detección de energía de movimiento, se necesitan 4 filtros separables, de forma que se sinteticen las dos parejas en cuadratura, independientes de la fase. Además, para normalizar las respuestas y quitar la ambigüedad del contraste frente a velocidad, se usa un cociente entre las respuestas de los filtros sensibles a movimiento y las respuesta de los filtros en ausencia de éste (aunque los valores absolutos de salidas cambien

proporcionalmente con el contraste, sus cocientes no lo harán). Esto puede ser problemático, cuando el denominador de los cocientes llega a ser pequeño, en cuyo caso esta medida no es fiable para etapas posteriores del sistema.

Adelson y Bergen [ADEL85] demuestran también que la energía de movimiento puede explicar algunos experimentos psicofísicos e incluso algunas ilusiones ópticas percibidas por humanos gracias al uso de filtros biológicos. En los modelos de gradiente, se evita el hecho de tener que construir filtros espacio temporales orientados (a partir de filtros separables espacio-temporales), ya que se estima el movimiento directamente con dichos filtros, sin necesidad de otras transformaciones.

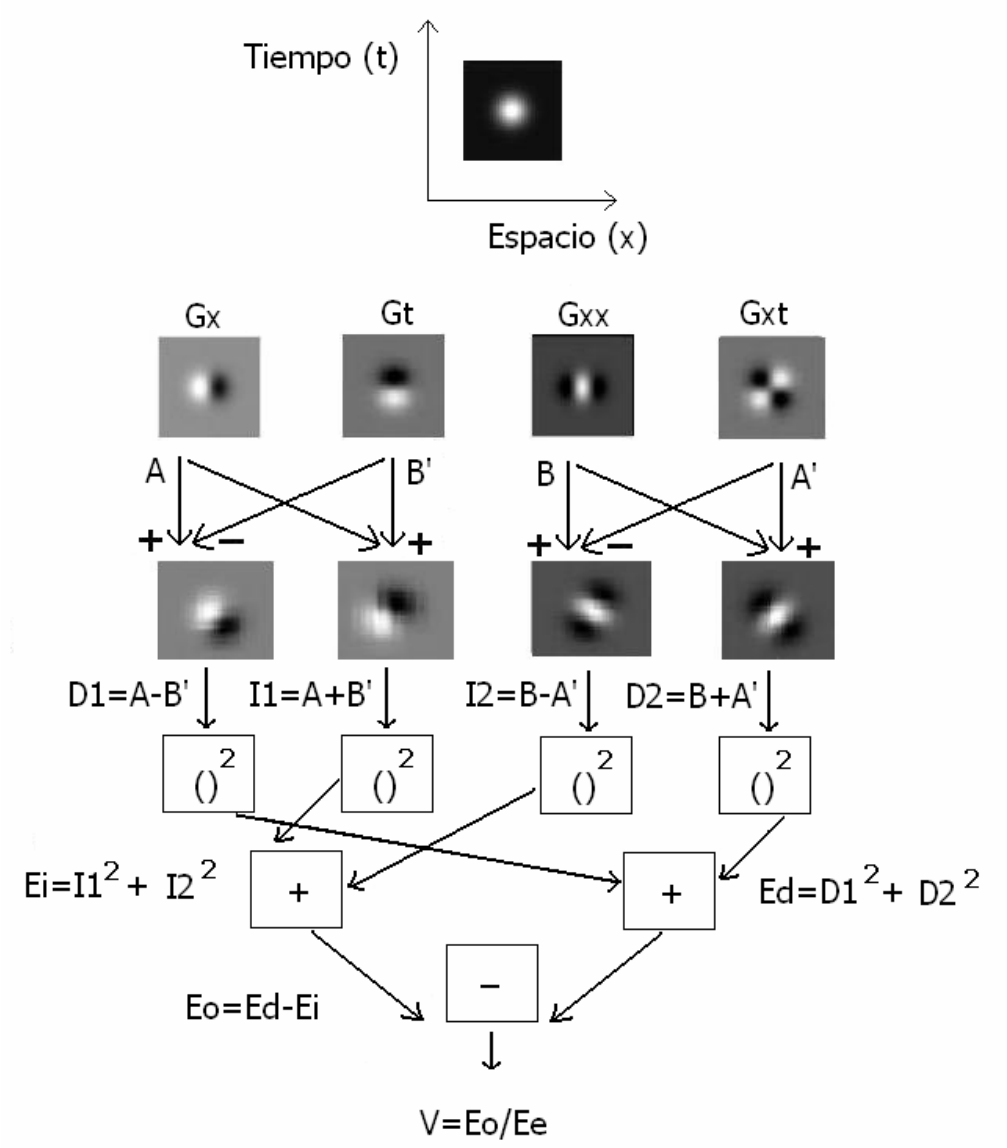


Figura 2.9. El modelo de energía de movimiento. Este esquema será el germen inicial del modelo de partida implementado en *hardware*, en el presente trabajo de investigación.

El diagrama de la figura 2.9 explica como se puede usar una combinación de salidas de filtros para estimar velocidad. Se disponen de cuatro unidades de procesamiento de entrada cuyas respuestas se denominan A, B, A', B'. En la siguiente etapa de procesamiento se suman o se restan siempre en cuadratura (con fase de 90º) para formar filtros direccionales en parejas, sensibles a movimiento hacia la izquierda y la derecha, cuyas salidas se denominan I1, I2, D1, D2. Se calcula posteriormente la energía de movimiento correspondiente a cada dirección sumando los cuadrados de las respuestas de las diferencias de los filtros. Con este paso, se emula el comportamiento de células complejas [DEV88]. La siguiente etapa es la denominada de oponencia, que toma la diferencia entre dos energías definiendo la energía de oponencia como $Eo = Ed\text{-}Ei = 4(A' \cdot B\text{-}A \cdot B')$. Este paso resalta específicamente una dirección y anula la respuesta a imágenes estacionarías. Desafortunadamente la energía de oponencia no estima velocidad, en parte debido a que esta medida varía enormemente con el contraste. Una normalización usando la energía estática *S* podría eliminar la dependencia de contraste, pero incluso este tipo de cálculos no garantizan la recuperación de velocidad a menos que los filtros se escojan cuidadosamente.

En un algoritmo ideal sería deseable que la relación energía de movimiento/energía estática *(Eo/Ee)* fuera igual a la velocidad verdadera (V) para todos los posibles patrones de movimiento y en principio esto se puede hacer. Se ha visto previamente que *A* y *B* son filtros espaciales no direccionales con la misma respuesta temporal, pero desfasados 90º, de forma que la energía estática viene expresada por la ecuación siguiente.

$$E_e = 4(A^2 + B^2) \Rightarrow Eo / E_e = (A'B - AB')/(A^2 + B^2) \qquad (2.22)$$

La ecuación (2.22) implica una analogía muy interesante con los modelos de gradiente y concretamente con el usado en este trabajo (se usará en el capítulo III de esta memoria). Los filtros presentados en la figura 2.9, son derivadas de Gaussianas espaciotemporales, que se usarán ampliamente en el modelo de energía [ADE85] y en el modelo multicanal de gradiente [JOH98].

Fleet y Jepson diseñan otro algoritmo que se ocupa de medir la orientación espacial y espacio-temporal [FLE89]. En este algoritmo los filtros se diseñan en el espacio de frecuencias usando un criterio de diseño basado en, la localización en el espacio-tiempo y la

especificidad de orientación (velocidad), así mismo, los filtros diseñados emplean funciones Gaussianas separadas como mínimo una desviación estándar.

Estas características aportan varias consecuencias interesantes, por ejemplo permiten conocer cómo el número de canales orientados crece linealmente con el espacio de frecuencias que hay que cubrir y cómo los métodos de energía mantienen todavía latente el problema de la apertura siempre que la energía movimiento caiga en un plano dentro del espacio de Fourier.

Fleet y Jepson vuelven a explotar la separabilidad de los bancos de filtros para una implementación eficiente usando diferencia de filtros Gaussianos como base para obtener filtros orientados. Sin embargo, puesto que el problema de la apertura permanece, restringen el flujo a regiones localmente planas, también asumen una constancia de flujo en un volumen espacio-temporal de forma que la FFT (transformada rápida de Fourier) pueda implementarse por bloques.

Simoncelli y Heeger [SIM98] presentan un modelo de dos etapas inspirado en la zona cerebral MT (V5) basado en filtros espacio-temporales sintonizados que se ajustan bien a un conjunto de datos fisiológicos, puesto que las dos etapas que corresponden a áreas corticales V1 y MT, imitan la respuesta de células simples y complejas.

Este algoritmo mide el contraste local para simular la respuesta de células simples y añade un pequeño valor para emular una tasa de disparo neuronal en un segundo plano. Esta medida alimentará a las células simples con campos receptivos orientados (velocidad sintonizada). La respuesta de la célula compleja del área MT es una media ponderada en una región espacial pequeña, produciéndose esta respuesta sobre todas las orientaciones y escalas de las células complejas con la misma velocidad de sintonización.

Los campos receptivos de las células simples se pueden modelar mediante derivadas de tercer orden de filtros Gaussianos o mediante filtros de Gabor (*v.* apéndice III), si se usa la primera opción, se puede aplicar todas las ventajas de la separabilidad de las funciones Gaussianas.

Simoncelli y Heeger sintetizan los filtros orientados espacio-temporales mediante una combinación lineal de filtros separables (espacio-temporales), lo que significa que atraviesan por una etapa donde se podría conformar un método de gradiente. Una neurona aislada MT no puede codificar velocidad ya que su respuesta es dependiente del contraste y de una familia de

velocidades, de ahí, la necesidad de una población de respuestas de células MT, este modelo también es capaz de detectar movimiento transparente producido por una distribución bimodal, siempre que los movimientos de ambas componentes difieran un umbral suficientemente significativo.

Como resumen de todo el apartado 2.4, se puede concluir que los métodos de energía de movimiento son posibles desde el punto de vista biológico aunque las implementaciones asociadas tienen un sobrecoste computacional alto debido al gran número de operaciones de filtrado necesarias. La velocidad resultante de los métodos de energía no se obtiene explícitamente, al contrario que en los métodos de gradiente, sino mediante una población de soluciones.

Una ventaja es que las medidas de velocidades bimodales como ocurre en movimientos invisibles, se pueden tratar mediante estas estructuras [SIM91]. Como contrapartida, la interpretación correcta de los resultados procesados, no resulta una tarea sencilla al tratar con modelos de naturaleza probabilística. También es interesante destacar que se han desarrollado optimizaciones para incrementar la velocidad de estos métodos, combinándolos con detectores de Reichardt [FRA92] a modo de apoyo.

2.5. Algoritmos implementados en *hardware configurable* existentes

Existen varios sistemas *hardware* fundamentados en los algoritmos mencionados en la presente memoria, procediéndose a hacer un repaso breve por el estado del arte a principio de 2007.

- Algunos algoritmos empleados son de emparejamiento o de gradiente, como el algoritmo de Horn y Schunck [HOR81] que ha sido llevado a cabo mediante una FPGA [ZUL98][MAR05]. El modelo empleado es sencillo, no es robusto y no proporciona óptimos resultados generales en *software*, sin embargo la implementación es eficiente, y el modelo es capaz de ir en tiempo real. El diseño usa prefiltrado y una implementación recursiva de la constricción de suavizado que aplica una iteración en cada fotograma.
- También existe una implementación del algoritmo de de Horn y Schunck [HOR81], realizada por Cobos *et al.* [COB98] en una plataforma FPGA, aunque con la misma

contrapartida anotada anteriormente sobre la fiabilidad del modelo usado y rozando el tiempo real.

- El algoritmo ASSET-2 se basa en características y ha sido implementado para ejecución en tiempo real usando *hardware* a medida [SMI95]. El algoritmo es bastante simple y determina la posición de los ejes y esquinas intentando resolver el problema de la correspondencia entre fotogramas vecinos en el tiempo. Para ello se almacenan tres características para cada esquina: el brillo y las dos coordenadas del centro de gravedad. Usando esta información es posible relacionar esquinas entre fotogramas con una tasa de aciertos del 85%. Para reducir el espacio de búsqueda y mejorar la robustez y velocidad se aplica un modelo de movimiento a la posición de cada esquina para predecir la posición en fotogramas siguientes. El sistema fue implementado usando un PowerPC junto con *hardware* a medida para extraer las características en tiempo real. Este sistema no proporciona resultados continuos, siendo éstos escasos, pero es capaz de agrupar grupos de velocidades similares para segmentar objetos según su movimiento.
- Niitsuma [NI04] implementan un modelo basado en correlación, que sí funciona en tiempo real.
- Díaz, Ros *et al* [DIA05] implementan un sistema en tiempo real supersegmentado basado en el algoritmo de Lucas y Kanade [LUC81] con muy buenos resultados en cuanto a rendimiento. Éste es un algoritmo dentro de los denominados de gradiente que se usa como método didáctico de introducción al flujo óptico en la mayoría de escuelas universitarias y se ha demostrado su buena relación resultados/esfuerzo de implementación. El problema de esta implementación viene dado por los cambios de iluminación bruscos y su gran dependencia con el problema de apertura. Su ventaja, es la gran documentación al respecto y experiencia con este algoritmo (más de 25 años) en prácticamente toda la comunidad científica relacionada con la visión por computador.

Es interesante notar que aunque las implementaciones en tiempo real producen peores resultados que sus algoritmos homólogos *software* en el sentido de las métricas usuales de evaluación de algoritmos de flujo óptico, sin embargo esto no es razón para no obtener resultados satisfactorios en sistemas de visión activa y sistemas relacionados con robótica móvil.

Se podría argumentar que incluso aún, cuando los resultados de un sólo fotograma no son de gran calidad, con 25 fotogramas por segundo se puede recuperar esta calidad perdida a través de algún tipo de mecanismo realimentado, permitiendo el desempeño robusto de tareas específicas como navegación y control [WEB94b], [SCA99], [DAN98].

2.6. Conclusiones

En este capítulo se ha realizado una revisión de los principios generales de la estimación de movimiento, estudiándose las diferentes tendencias y el estado del arte actual en la primera mitad de 2007 para implementaciones *software* y *hardware*. Una discusión bastante exhaustiva respecto a los fundamentos del problema a tratar se puede encontrar en el trabajo de Beauchemin y Barron [BEA95] y una introducción bastante buena en cuanto al origen de muchos algoritmos (dinámica de de los medios continuos y su relación con el flujo óptico) se puede encontrar en el trabajo de Pajares [PAJ03].

Después de estudiar las diferentes técnicas de estimación de flujo óptico, se puede concluir que las técnicas de correlación sólo son útiles para largos desplazamientos y que combinadas con una aproximación mutiescala mejoran notablemente. Entre los algoritmos de emparejamiento, los métodos propuestos por Shing [SHI03] parecen ser los más exactos.

Los métodos de gradiente y energía se han mostrado como extractores de movimiento a través de una orientación espacio-temporal, aunque con diferentes estrategias de filtrado. Ambas estrategias están basadas en modelos biológicos, aunque el de gradiente es el que más se presta a implementaciones paralelas, siendo el que se utiliza en el presente trabajo de investigación [JOH95]. Estos métodos diferenciales proporcionan el mejor rendimiento con respecto a exactitud y densidad siempre que se tenga que hacer distinción entre flujo normal y perpendicular y se usen métodos de resolución basados en mínimos cuadrados (locales y totales). Los métodos diferenciales de segundo orden [URA88] proporcionan muy buenos resultados para movimiento translacional sobre cualquier rango de velocidades (hasta el límite del muestreo temporal) independientemente de la dirección de movimiento, sin embargo no deben usarse para otros tipos de movimiento como divergencia o rotación.

Los algoritmos de estimación del flujo óptico que combinan medidas sobre el tiempo son más robustos que aquellos que tratan cada fotograma de forma aislada. Los requerimientos de estos algoritmos se basan en una tasa de muestreo alta que respete una coherencia temporal.

Por otra parte, a medida que la tecnología permita sistemas más avanzados operando en tiempo real, se tendrá una mayor capacidad de implementar algoritmos más complejos y se descubrirán nuevas interacciones entre resultados, hasta ahora desconocidas.

CAPÍTULO III

Modelo multicanal de gradiente
Implementación, desarrollo y justificación

3.1. Motivación

Según se ha explicado en el capítulo II, existen 3 estrategias básicas para el cálculo del flujo óptico, los modelos de correlación, los de energía y los de gradiente. Los dos primeros aplican esencialmente plantillas, siendo el objetivo de estas familias de modelos el obtener mecanismos que respondan bien al movimiento de un estímulo, respecto a una dirección y con una velocidad específica, dando de forma ideal, un ajuste entre el movimiento y la plantilla.

Obviando en este análisis preliminar otras dificultades menores, hay que señalar que la respuesta de estos 2 tipos de modelos depende fuertemente del contraste del estímulo, necesitando etapas de normalización complejas que aunque atenúan efecto, no lo evitan completamente [JOH95]. Adicionalmente, las respuestas a estas plantillas no proporcionan directamente el movimiento, teniendo que calcular la velocidad en etapas posteriores (normalmente poblaciones de filtros, como se vio en el capítulo II).

La tercera estrategia (modelos de gradiente) tiene una filosofía completamente diferente debido a que la velocidad se calcula directamente mediante un cociente de derivadas temporales y espaciales en cada punto. Como el contraste varía igualmente en el denominador y el numerador, el resultado no debe ser afectado por éste, resultando un sistema invariante al contraste sin un sobrecoste adicional.

Este cualidad es importante, aunque deja de cumplirse en zonas de patrones con bajas frecuencias espaciales y temporales, donde no hay información suficiente para disparar un cierto umbral de activación. Johnston [JOH95], realiza un estudio de la velocidad estimada en función de este umbral para una población de filtros derivativos, una contrapartida añadida al problema anterior, aparece con la dependencia de estas tres familias de modelos del patrón estático frente al dinámico.

Consecuentemente, es necesario el desarrollo de otros modelos más elaborados y robustos frente a estos factores, como modelos de gradiente (versiones previas del McGM [JOH92], Lucas [LUC81]), que solventan esta dependencia del patrón estático introduciendo una etapa de filtrado paso banda previa al procesamiento, sin embargo, este uso añade nuevas contrapartidas indeseadas:

- Al eliminar las frecuencias inferiores, se reduce la señal en patrones con un movimiento lento asociado. (se elimina información potencialmente útil).
- Teniendo en cuenta que el umbral inferior de movimiento en la fóvea humana es 0.02º/seg, [JOH94], (correspondiente a una orientación en el espacio-tiempo de 1º, si se escala para que 1º/seg equivalga a 45º de orientación), es difícil justificar un filtro temporal biológicamente plausible que quite la influencia del patrón estático al mismo tiempo que mantiene esta alta sensibilidad a movimientos lentos. Esto sugiere que en sistemas biológicos, se evita la influencia del patrón estático gracias a mecanismos especiales del córtex en etapas posteriores, en lugar de operaciones de filtrado en etapas previas. Existe una consistencia neuronal y fisiológica a nivel de primates [NEW88], que concluye que una lesión en el área cerebral de movimiento (V5-MT) aumenta la sensibilidad al ruido estático y dinámico con sólo un pequeño efecto en el umbral de detección de filtros de bajo nivel [HES89] [PAS94]. Este tipo de modelos proporciona una justificación convincente para la sensibilidad al patrón estático mostrado por pacientes con una lesión del tipo ceguera al movimiento.

Consecuentemente, eliminar el patrón estático utilizando el filtrado temporal no es biológicamente factible, concluyendo que si se desea un sistema biológicamente

asequible, es necesario adoptar una estrategia donde la etapa de independencia al patrón estático esté ubicada al final del mismo. Además, existen otras dificultades:

- Es necesario realizar operaciones de inversión de grandes matrices (Lucas y Kanade [LUC81]), que no son biológicamente justificables. Para este trabajo se ha adoptado un esquema basado en operaciones que utilicen exclusivamente respuestas de filtros orientados (calculadas con operaciones de suma, multiplicación y división).

Se va a construir, entonces, una alternativa a estos 3 modelos previos que cumpla con todos estos requisitos (invarianza al contraste, invarianza al patrón estático sin perder información, consistencia biológica, es decir, el uso de filtros temporales y espaciales encontrados en el ser humano, y el uso de operaciones sólo utilizando estos filtros). El objetivo, del presente capítulo es diseñar un modelo que cumpla todas estas condiciones, implementado en *software* (C++/Matlab) y que se pueda evaluar de forma sencilla.

Este algoritmo atesora una serie de propiedades que no cumplen otros modelos convencionales de gradiente, como: robustez frente a ruido estático y dinámico, consistencia matemática frente a regiones donde no hay apenas contraste, versatilidad para justificar procesamiento de movimiento de segundo orden, capacidad de separar las componentes diferenciales del flujo óptico sin recurrir a métodos iterativos entre otras, algunas de las cuales se justificarán en apéndices posteriores, así mismo se anexa al final de esta memoria (apéndice II) un completo diagrama de flujo desglosado de forma jerárquica, para su correcto seguimiento.

3.2. Introducción al modelo multicanal de gradiente. McGM

3.2.1. Introducción

En esta parte se describe la estructura matemática y computacional del modelo multicanal de gradiente o McGM, [JOH95] [JOH99].

El modelo extiende la ecuación básica de conservación de la intensidad para flujo óptico, expresando la imagen como un desarrollo de Taylor e introduciendo derivadas de órdenes superiores en la descripción. Gracias a esto, se consigue una mayor robustez y se

obtiene un método apropiado para analizar la estructura espacio temporal de la secuencia de imágenes. Mediante el cociente de la derivada temporal y espacial de varios términos del desarrollo de Taylor (calculando este desarrollo para un número de direcciones que se corresponden a la orientación de las columnas encontradas en el córtex visual primario), se obtiene una medida de la velocidad como función de la orientación del filtro.

Esta estrategia de múltiples medidas, ayuda a reducir el ruido y permite separar la componente de translación de la diferencial. Debido a que cada cociente individual está mal condicionado en puntos donde el denominador se hace nulo, se combinan las derivadas de varios términos de este desarrollo y se integran en un volumen espacio-temporal. A partir de esta cantidad de datos se producen las primitivas de velocidad y velocidad inversa que a su vez generan la estimación de velocidad en módulo y fase para cada punto calculado, obteniendo finalmente, un mapa denso de información.

3.2.2. Cálculo del desarrollo de Taylor

La función de luminosidad de la imagen se puede expresar según la expresión siguiente:

$$I(x+p,y+q,t+r)=\sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n}\frac{p^i q^i r^k}{i!\,j!k!}\frac{\partial^n}{\partial x^i \partial y^j \partial t^k}I(x,y,t) \tag{3.1}$$

en particular al tomar l,m,n=1 (despreciando los demás términos) y $p=\delta x$, $q=\delta y$, $r=\delta t$,

$$I(x+\delta x,y+\delta y,t+\delta t)=I(x,y,t)+\frac{\partial I(x,y,t)}{\partial x}\delta x+\frac{\partial I(x,y,t)}{\partial y}\delta y+\frac{\partial I(x,y,t)}{\partial t}\delta t \tag{3.2}$$

si se supone la intensidad constante a lo largo del espacio-tiempo:

$$I(x+p,\ y+q\ ,t+r\ )=I(x,q,t) \tag{3.3}$$

mediante la combinación de (3.2) y (3.3) se obtiene:

$$\frac{\partial I(x,y,t)}{\partial x}\delta x+\frac{\partial I(x,y,t)}{\partial y}\delta y+\frac{\partial I(x,y,t)}{\partial t}\delta t=0 \tag{3.4}$$

Dividiendo ambos miembros por δt, se obtiene la ecuación de restricción del flujo óptico que aparece en la mayoría de literatura especializada, siendo esta ecuación una premisa, en la que se basa el cálculo de estimaciones de movimiento a partir de secuencias de imágenes.

$$\frac{\partial I(x,y,t)}{\partial x}\frac{\delta x}{\delta t}+\frac{\partial I(x,y,t)}{\partial y}\frac{\delta y}{\delta t}+\frac{\partial I(x,y,t)}{\partial t}=0 \tag{3.5}$$

Las pareja $\{\delta x/\delta t,\ \delta y/\delta t\}$ conforma el vector de velocidad local. Se tiene pues, una ecuación con 2 incógnitas $\{\delta x/\delta t,\ \delta y/\delta t\}$, por tanto no se puede determinar la velocidad real sin imponer más restricciones. Esto da lugar al llamado problema de apertura, siempre que no exista textura suficiente para extraer información [PAJ02]. Como se ha visto en el capítulo II, cualquier término del desarrollo de Taylor puede ser usado para formar la ecuación de restricción, eligiendo por ejemplo, $\frac{\partial I(x,y,t)}{\partial x}$, se llega a la ecuación 3.6.

$$\frac{\partial^2 I\ (x,y,t)}{\partial x^2}\frac{\delta x}{\delta t}+\frac{\partial^2 I\ (x,y,t)}{\partial y\partial x}\frac{\delta y}{\delta t}+\frac{\partial^2 I\ (x,y,t)}{\partial t\partial x}=0 \tag{3.6}$$

De esta manera, se asume también la conservación de la derivada con respecto a *x* a lo largo del tiempo. Cuando todas las derivadas sean iguales a cero, la medida estará mal condicionada, sin embargo, si se interpreta el desarrollo de Taylor como una manera de almacenar las distintas derivadas ponderadas, se puede solventar este problema.

Partiendo de una dirección primaria *x*, correspondiente a una columna orientada en V1 o V5-MT [NEW88], se construye un vector, cuya *n*-ésima componente contiene el valor del *n*-ésimo término de la aproximación de Taylor. Cada componente del vector se trunca eliminando términos superiores en el primer orden temporal y en el primer orden en la dirección ortogonal a la dirección primaria. La intención de este mecanismo es aproximar la intensidad de un punto *P(x,y,t)* a través del desarrollo espacio-temporal truncado para asegurarse que el modelo no tiene más de 3 filtros temporales y no hay un orden espacial en la dirección ortogonal superior a 2. Esto se hace así, porque llegar a más órdenes, llevaría a un algoritmo no bioinspirado y a tener información redundante, debido a no tener un conjunto de

filtros completamente ortogonal, estando sus derivadas de órdenes superiores cada vez más correlacionadas entre sí.

H(p,q,r) se va a definir como un vector, desde *P(x,y,t),* al punto en el campo visual que requiere la aproximación. Cada una de las derivadas parciales, se representa por la salida de un filtro cortical espacio-temporal lineal con un campo receptivo centrado en la posición *P* en la imagen. Las variables *p, q* y *r* son pesos de las salidas de los filtros, que permiten la aproximación de la intensidad de la imagen en algún punto *P+H* en la vecindad de *P*. Con todo ello se va a construir el desarrollo de Taylor, mediante derivadas en *x* hasta orden 3 y en *y*, *t* hasta orden 1.

$$
\begin{aligned}
I(x+p,y+q,t+r) &= I(x,y,t) + \{r\frac{\partial(I(x,y,t))}{\partial t} + q\frac{\partial(I(x,y,t))}{\partial y} + p\frac{\partial(I(x,y,t))}{\partial x}\} + \\
&+ \frac{1}{2!}\{p^2\frac{\partial^2(I(x,y,t))}{\partial x^2} + 2qr\frac{\partial^2(I(x,y,t))}{\partial y\partial t} + 2pr\frac{\partial^2(I(x,y,t))}{\partial x\partial t} + 2pq\frac{\partial^2(I(x,y,t))}{\partial x\partial y}\} + \\
&+ \frac{1}{3!}\{p^3\frac{\partial^3(I(x,y,t))}{\partial x^3} + 3p^2r\frac{\partial^3(I(x,y,t))}{\partial x^2\partial t} + 3p^2q\frac{\partial^3(I(x,y,t))}{\partial x^2\partial y} + pqr\frac{\partial^3(I(x,y,t))}{\partial x\partial y\partial t}\} + \\
&+ \frac{1}{4!}\{p^4\frac{\partial^4(I(x,y,t))}{\partial x^4} + 2p^2qr\frac{\partial^4(I(x,y,t))}{\partial x^2\partial y\partial t} + 4p^3r\frac{\partial^4(I(x,y,t))}{\partial x^3\partial t} + 4p^3q\frac{\partial^4(I(x,y,t))}{\partial x^3\partial y}\} + \\
&+ \frac{1}{5!}\{5p^4q\frac{\partial^5(I(x,y,t))}{\partial x^4\partial y} + 5p^4r\frac{\partial^5(I(x,y,t))}{\partial x^4\partial t} + 20p^3qr\frac{\partial^5(I(x,y,t))}{\partial x^3\partial y\partial t}\} + \frac{1}{6!}\{30p^4qr\frac{\partial^6(I(x,y,t))}{\partial x^4\partial y\partial t}\}
\end{aligned}
\tag{3.7}
$$

Para implementar el desarrollo anterior, es necesario crear las derivadas (en formas de filtros) y sus pesos. Se procede a diseñar los diferentes operadores derivativos.

3.2.2.1. Operadores derivativos temporales

La función (3.8) filtrará temporalmente el estímulo de entrada usando derivadas de orden 0,1,2. El presente núcleo es una función dependiente del tiempo, a partir de la cual se obtienen las diferentes componentes derivativas, modelándose como una Gaussiana en el espacio temporal logarítmico con los parámetros ($\alpha = 10$, $\tau = 0.2$, de forma general).

$$
nucleo = \frac{e^{-(\log(t/\alpha)/\tau)^2}}{\pi^{1/2}\alpha e^{(\frac{\tau^2}{4})}} \tag{3.8}
$$

$$\frac{\partial(nucleo_{temporal})}{\partial t} = -2\frac{\ln(t/\alpha)}{\tau^2 t}\frac{e^{-(\log(t/\alpha)/\tau)^2}}{\pi^{1/2}\alpha e^{(\frac{\tau^2}{4})}} \tag{3.9}$$

$$\frac{\partial^2(nucleo_{temporal})}{\partial t^2} = -2\frac{\ln(t/\alpha)}{\tau^2 t}\{-2\frac{\ln(t/\alpha)}{\tau^2 t}\frac{e^{-(\log(t/\alpha)/\tau)^2}}{\pi^{1/2}\alpha e^{(\frac{\tau^2}{4})}}\} - \frac{2(1-\ln(t/\alpha))}{\tau^2 t^2}\{\frac{e^{-(\log(t/\alpha)/\tau)^2}}{\pi^{1/2}\alpha e^{(\frac{\tau^2}{4})}}\} \tag{3.10}$$

En la figura 3.1 se muestran las derivadas de la Gaussiana de orden 0, 1 ,2 así como su respuesta en frecuencia. Se aprecia un filtro paso-baja con una frecuencia de corte en 10 Hz aproximadamente y dos filtros paso-banda a 5-10 Hz y 12-18 Hz.

La forma y naturaleza de estas derivadas temporales viene condicionada por el hecho de inspirarse e intentar reproducir los resultados encontrados por Hess y Snowden respecto a los canales visuales encontrados en el ser humano [HES92a], [HESS92b] (*v*. apéndice III).

Para obtener la derivada temporal de la imagen, es necesario convolucionar cada secuencia con cada uno de estos 3 filtros temporales, expresando en la figura 3.2 y la expresión (3.11) el proceso de filtrado:

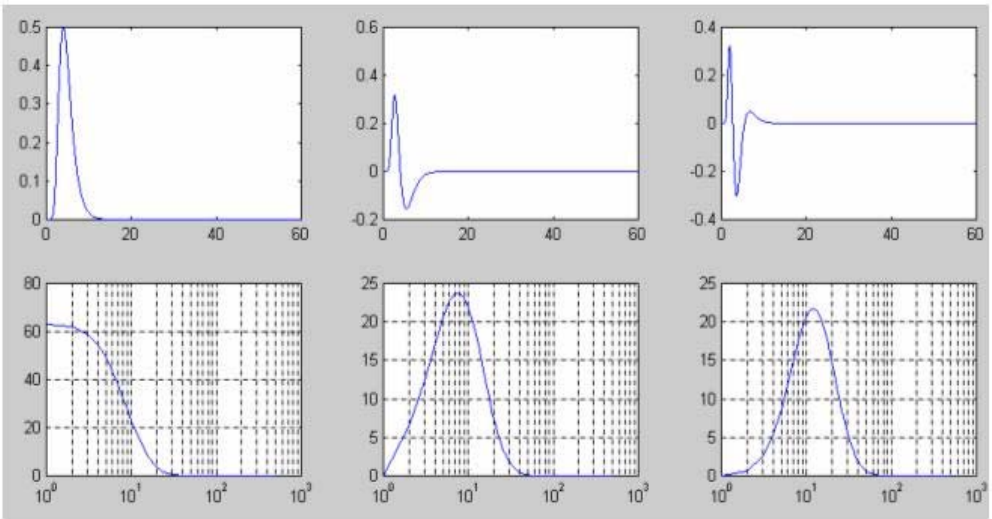


Figura 3.1. Función Gaussiana implementada en el dominio log(t) y sus primeras derivadas (graficas en la parte superior), su respuesta en frecuencia (gráficas en la parte inferior), apreciándose claramente los canales descubiertos por Hess & Snowden.

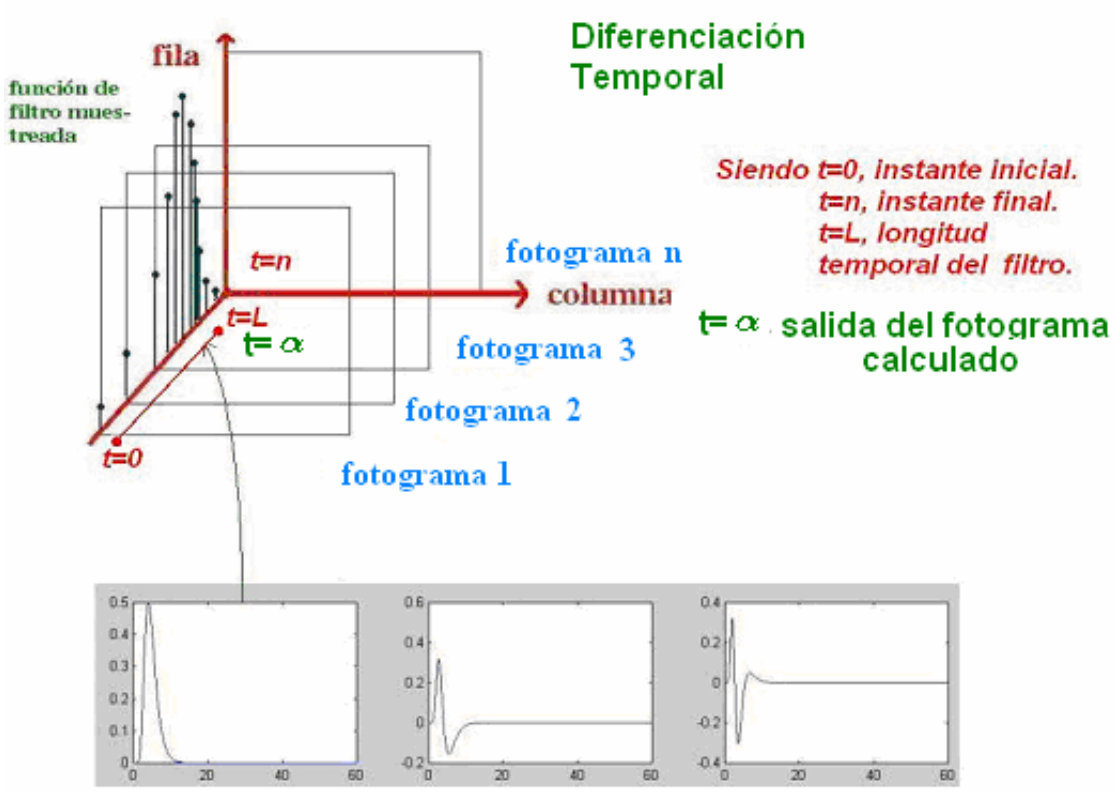


Figura 3.2. Convolución temporal de la secuencia de imágenes con los operadores derivativos diseñados.

$$Convolución_{temporal}(orden) = \sum_{t=0}^{tconv} \sum_{fila=0}^{TAMAÑO} \sum_{columna=0}^{TAMAÑO} filtro_{temporal}(orden,t)imagen(fila,columna) \qquad (3.11)$$

El modelo utiliza un filtro FIR para implementar las derivadas temporales. Por la naturaleza del proceso de filtrado no se obtendrán salidas hasta pasar procesar tantos fotogramas en el filtro como longitud de éste, es decir, si se tiene un filtro temporal de longitud 23, hasta el instante *t=23* (con 23 imágenes) no será posible obtener la respuesta al fotograma dado por la posición α.

Si se realiza este proceso triplemente convolutivo desde el primer fotograma *(t=0)* hasta el que ocupe la última posición menos la longitud del filtro *(t=n-L+1)*, se consigue un conjunto de derivadas temporales de orden 0, 1 y 2 que actuarán como primeros canales de información, extrayéndose a partir de ellos cada una de las variaciones espaciales asociadas a la estructura de la imagen. A continuación se procede a diseñar y justificar todas estas

operaciones espaciales que irán encadenadas a las convoluciones temporales abordadas en este apartado.

3.2.2.2. Operadores derivativos espaciales

También es necesario calcular las derivadas espaciales, de orden 0 hasta 6 en *x* además de orden 0 hasta 2 en *y*. Para ello se usarán operadores diferenciales espaciales, donde será otra función núcleo (en este caso una Gaussiana), la que se derive para obtener estos operadores diferenciales de órdenes superiores.

$$nucleo_{espacial} = \frac{e^{\frac{-x^2}{2\sigma^2}}}{\sigma(2\pi)^{1/2}} \tag{3.12}$$

La derivada unidimensional de la Gaussiana viene dada por:

$$\frac{d^n}{dx^n}\left(\frac{e^{-\frac{x^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}\right) \tag{3.13}$$

donde σ es su desviación típica. (Las funciones están normalizadas a la unidad). Si se generaliza el proceso de la obtención de derivadas, a partir de la fórmula de Rodrígues [SPI88], función generatriz de los polinomios de Hermite.

$$H_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n}\left(e^{-x^2}\right) \tag{3.14}$$

Si se despeja la derivada enésima de *x* y a continuación se procede a realizar un sencillo cambio de escala. $x \rightarrow x/2^{1/2}\sigma$, se llega a la expresión (3.15):

$$\frac{d^n}{dx^n}\left(\frac{e^{-\frac{x^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}\right) = H_n\left(\frac{x}{\sqrt{2}\sigma}\right)\left(\frac{-1}{\sqrt{2}\sigma}\right)^n \left(\frac{e^{-\frac{x^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}\right) \tag{3.15}$$

obteniéndose una fórmula que proporciona las derivadas de la Gaussiana en orden creciente. A partir de estas derivadas en una dimensión, se puede extrapolar a 2D, sin más que multiplicar derivadas 1D en *x* e *y* respectivamente.

$$\frac{d^n}{dx^n}G(x,y,\sigma)=\frac{d^n}{dx^n}\left(\frac{e^{-\frac{x^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}\right)\frac{d^n}{dy^n}\left(\frac{e^{-\frac{y^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}\right) \tag{3.16}$$

Como en el caso anterior, si se sustituyen operadores diferenciales por polinomios de Hermite, se llega a la expresión (3.17):

$$\frac{d^n}{dx^n}\left(\frac{e^{-\frac{(x^2+y^2)}{2\sigma^2}}}{\sigma\sqrt{2\pi}}\right)=H_n(\frac{x}{\sqrt{2}\sigma})H_n(\frac{y}{\sqrt{2}\sigma})(\frac{-1}{\sqrt{2}\sigma})^{2n}\left(\frac{e^{-\frac{(x^2+y^2)}{2\sigma^2}}}{\sigma\sqrt{2\pi}}\right) \tag{3.17}$$

A medida que se usan filtros de mayor tamaño, se constata una mejora de rendimiento al usar filtros separables. Si el tamaño de la imagen es *n* y el tamaño del filtro es *m*, el número de operaciones requeridas para implementar el filtro Gaussiano bidimensional es O($m^2 \cdot n^2$), que se reduce esto a O($2m \cdot n^2 \cdot$) utilizando filtros separables. Así, usando un tamaño de filtros de 23^2, el usar filtros separables implica una mejora de un factor de 11.5.

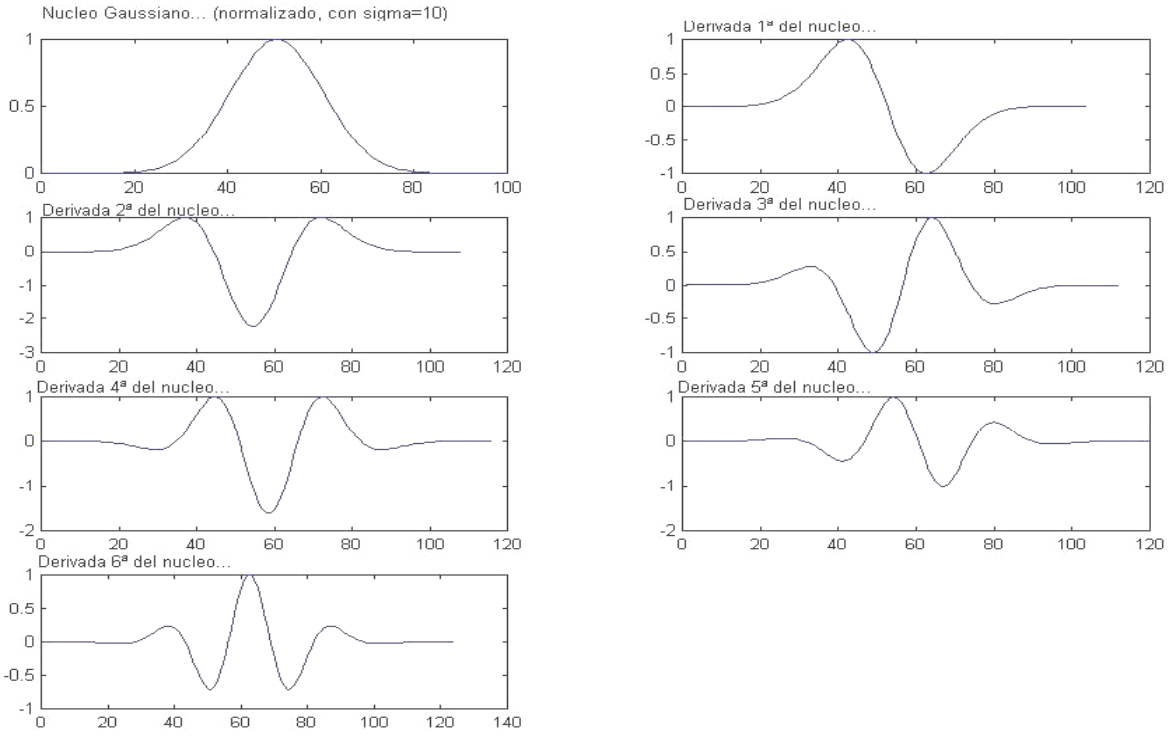


Figura 3.3. Filtros espaciales (1D) usados en el modelo McGM descrito. Derivadas de Gaussianas de distintos órdenes : orden 0 hasta 6. (todas están normalizadas a 1).

3.2.2.3. Creación del banco de filtros

Se tiene un banco de filtros espaciales como se ilustra en la figura 3.3 (donde aparecen filtros espaciales desde orden 0 hasta 6, siguiendo un orden de izquierda a derecha y de arriba abajo). Este banco se ha creado como se ha comentado previamente, a partir de las derivadas espaciales de una función núcleo, modelada por una Gaussiana.

A partir de los filtros de la figura 3.3, se aplica un proceso de muestreo con un número de puntos que definirá el tamaño del filtro espacial, (siendo éste uno de los parámetros importantes del modelo).

$$\begin{bmatrix} orden0_0 & orden0_1 & \dots & orden0_{n-1} & orden0_n \\ orden1_0 & orden1_1 & \dots & orden1_{n-1} & orden1_n \\ orden2_0 & orden2_1 & \dots & orden2_{n-1} & orden2_n \\ orden3_0 & orden3_1 & \dots & orden3_{n-1} & orden3_n \\ orden4_0 & orden4_1 & \dots & orden4_{n-1} & orden4_n \\ orden5_0 & orden5_1 & \dots & orden5_{n-1} & orden5_n \\ orden6_0 & orden6_1 & \dots & orden6_{n-1} & orden6_n \end{bmatrix} \tag{3.18}$$

Después de almacenarlos, estos filtros unidimensionales se combinarán entre sí, para formar filtros bidimensionales. En virtud de la separabilidad de los operadores diferenciales, es posible hacer primero la convolución por filas. Repitiendo para *t=0,1,2*; *ordenc=0,1,2,3,4,5,6*; *fila*, *columnas.*

$$Convolución_{filas}(t, orden_c, fila, columna) = \sum_{c=0}^{(Conv-1)} filtro(orden, c)imagen(t, fila, columna + c) \tag{3.19}$$

Donde se convoluciona cada uno de los filtros espaciales de *ordenc* 0…6 con cada imagen derivada temporalmente de forma previa (orden temporal cero, primero y segundo) para cada fila y cada columna. Este resultado, a su vez, repitiendo el proceso anterior se convoluciona por columnas, consiguiendo un conjunto de información correspondientes a las salidas asociadas a 84 filtros (orden 0 hasta orden 6 para fila y columna).

$$Convolución_{columnas}(t, orden_c, fila, columna) = \sum_{c=1}^{(Conv-1)} filtro(orden, conv)Convolución_{filas}(t, fila + c, columna) \tag{3.20}$$

Figura 3.4. Estructura piramidal diseñada al efecto que servirá como banco de filtros bidimensional. (Nótese que aparecen las derivadas cruzadas de orden 0 hasta 6)

Por lo tanto, automatizando este proceso, es posible generar un banco de filtros bidimensionales desde un orden nulo hasta sexto orden, tal y como se ejemplifica mediante la figura 3.4. En dicha figura se muestra la pirámide formada por la convolución bidimensional separable, donde cada filtro se puede asociar a su expresión matemática dada por la pirámide homóloga de la figura 3.5.

El orden de codificación es muy sencillo, avanzando por columnas en la dirección *x* y por filas en la dirección *y*.

$$
\begin{array}{ccccccc}
nucleo & \frac{\partial nucleo}{\partial x} & \frac{\partial^2 nucleo}{\partial x^2} & \frac{\partial^3 nucleo}{\partial x^3} & \frac{\partial^4 nucleo}{\partial x^4} & \frac{\partial^5 nucleo}{\partial x^5} & \frac{\partial^6 nucleo}{\partial x^6} \\
\frac{\partial nucleo}{\partial y} & \frac{\partial^2 nucleo}{\partial x \partial y} & \frac{\partial^3 nucleo}{\partial x^2 \partial y} & \frac{\partial^3 nucleo}{\partial x^2 \partial y} & \frac{\partial^3 nucleo}{\partial x^3 \partial y} & \frac{\partial^3 nucleo}{\partial x^4 \partial y} & \\
\frac{\partial^2 nucleo}{\partial y^2} & \frac{\partial^3 nucleo}{\partial x \partial y^2} & \frac{\partial^3 nucleo}{\partial x^2 \partial y^2} & \frac{\partial^3 nucleo}{\partial x^2 \partial y^2} & \frac{\partial^3 nucleo}{\partial x^2 \partial y^2} & & \\
\frac{\partial^3 nucleo}{\partial y^3} & \frac{\partial^3 nucleo}{\partial x \partial y^3} & \frac{\partial^3 nucleo}{\partial x^2 \partial y^3} & \frac{\partial^3 nucleo}{\partial x^2 \partial y^3} & & & \\
\frac{\partial^4 nucleo}{\partial y^4} & \frac{\partial^3 nucleo}{\partial x \partial y^4} & \frac{\partial^3 nucleo}{\partial x^2 \partial y^4} & & & & \\
\frac{\partial^5 nucleo}{\partial y^5} & \frac{\partial^3 nucleo}{\partial x \partial y^5} & & & & & \\
\frac{\partial^6 nucleo}{\partial y^6} & & & & & &
\end{array}
$$

Figura 3.5. Expresiones algebraicas de las derivadas mostradas en la Figura 3.4.

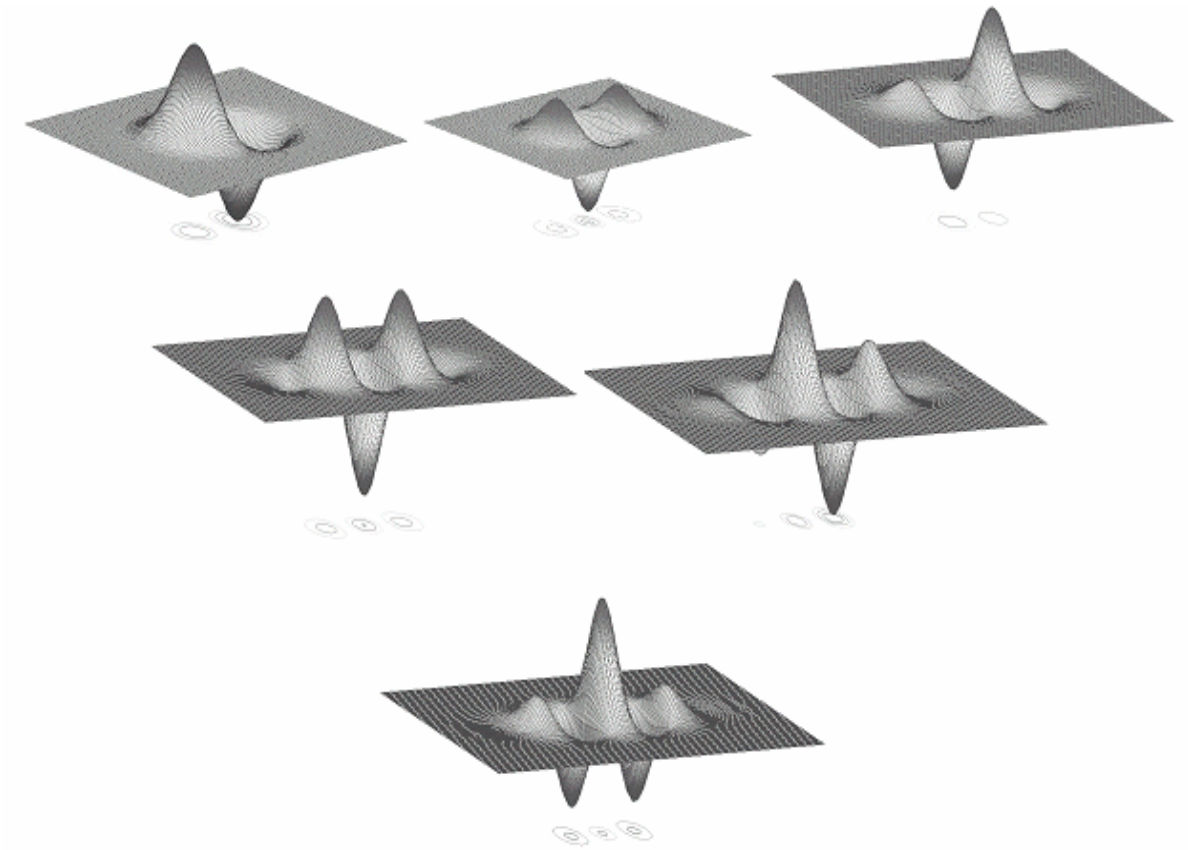


Figura 3.6. Banco de filtros bidimensional de orden 1 a 6 (arriba a abajo e izquierda a derecha).como resultado de convolucionar una Gaussiana unidimensional por columnas con los filtros de la figura 3.3 por filas, haciendo uso de la propiedad de separabilidad de la Gaussiana y sus derivadas.

Esta forma de representación del conjunto de filtros basada en pirámides bidimensionales es altamente atractiva y se usará activamente en el presente capítulo y posteriores, ya que permite organizar la información de manera que con sólo un índice se puede avanzar dentro del mismo orden total de diferenciación, que corresponde a cada frente diagonal. Por otra parte, en la figura 3.6 se muestra una representación tridimensional de las primeras derivadas del núcleo Gaussiano, con órdenes que varían desde el primero hasta el sexto. (Todas las funciones están normalizadas a la unidad y mostradas en perspectiva, omitiendo deliberadamente los ejes para mayor claridad).

Este tipo de filtros son muy útiles en el campo de la visión artificial especialmente en aplicaciones relativas al análisis de orientación local, filtrado angular adaptativo, detección de bordes o formación de imágenes tridimensionales a partir de sombras bidimensionales. Como aplicación práctica se va a diseñar un detector sencillo de bordes convolucionando la imagen de la figura 3.7, con el banco piramidal de filtros de la figura 3.8.

Figura 3.7. Imagen de entrada, con frecuencia espacial variable, y representación en escala de grises (8 bits).

En dicho banco se tienen las 10 primeras derivadas en x, y correspondientes a un orden total de diferenciación comprendido entre 0 y 3, representándose los grupos de derivadas del mismo orden por líneas de diferente color. Se procede pues, a realizar una convolución separable de la imagen correspondiente a la figura 3.7 con el banco de la figura 3.8 representando el resultado de cada una de las convoluciones individuales en la figura 3.9.

Se han señalado dos zonas de interés (de rojo y azul), donde aparecen los bordes orientados de forma vertical y horizontal, pudiéndose comprobar por una sencilla inspección

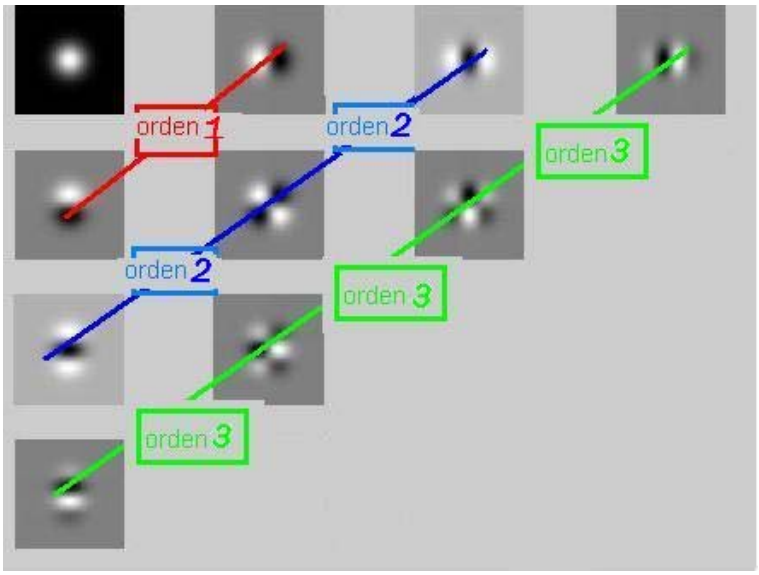


Figura 3.8. Banco de filtros de diferente orden que se usará en este ejemplo.

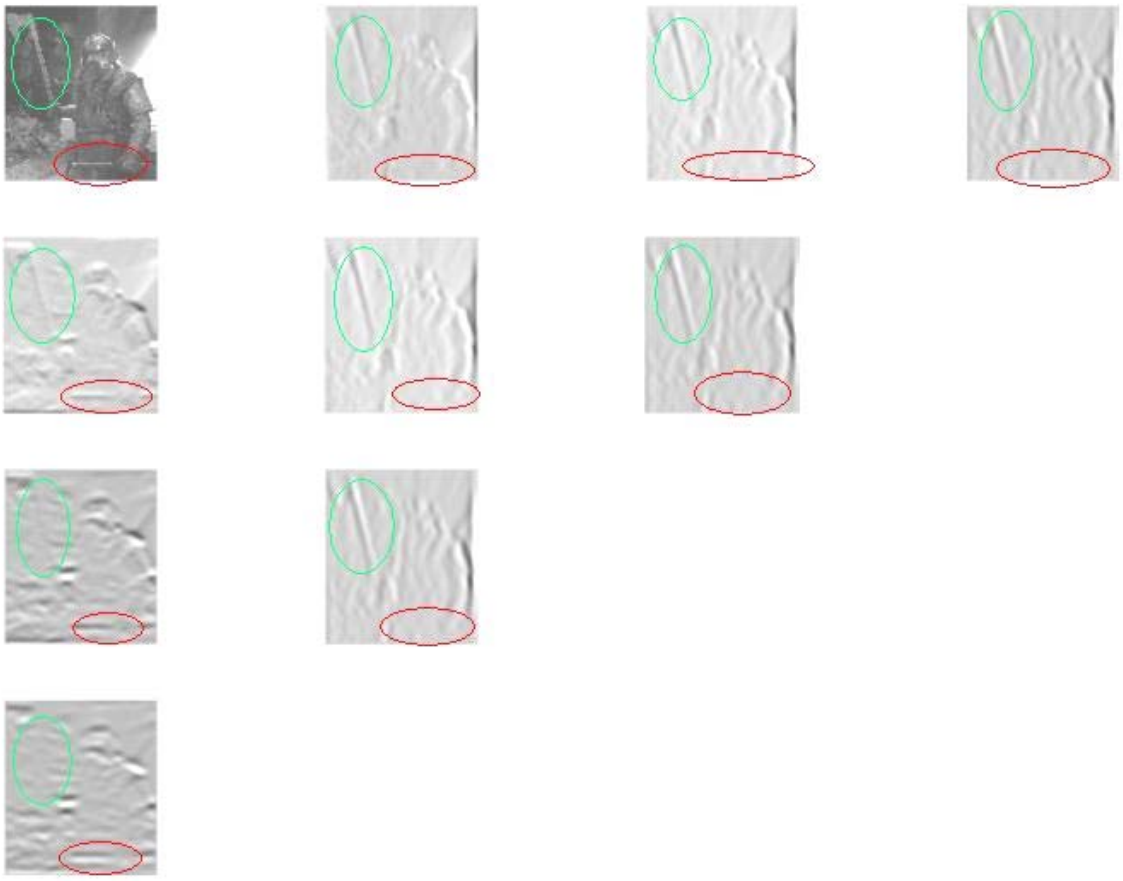


Figura 3.9. Resultado de la detección de bordes robusta, aplicando el banco piramidal de la figura 3.8. Es posible apreciar en rojo **y** azul (donde dominan las frecuencias horizontales, verticales respectivamente) y su detección frente a diferentes tipos de filtros (de orden 0 a 3 en *x* y en *y*, incluyendo todos los órdenes cruzados).

ocular la activación/inhibición de los filtros horizontales para los bordes horizontales/verticales y viceversa.

Por la linealidad de los filtros, este proceso total equivale primeramente a convolucionar la imagen con una Gaussiana (esto se hace porque el siguiente operador derivada, no se comporta bien frente a discontinuidades), suavizando la imagen (quitando componentes de baja frecuencia). Posteriormente se convoluciona este resultado previo con el operador derivada, que se ocupa de detectar bordes con una determinada orientación (el diseño de filtros orientados a cualquier ángulo es el objeto de las secciones siguientes). Este esquema, sin embargo se comporta como un filtro paso alta, que responde bien a partir de determinadas frecuencias espaciales. El producto de los dos es un filtro paso banda, cuya frecuencia de corte inferior viene limitada por la asociada a la Gaussiana y la superior viene limitada por las derivadas.

En este punto, se han construido todas las derivadas temporales y espaciales que utiliza el algoritmo, procediéndose ahora a confeccionar las derivadas direccionales.

3.2.2.4. Filtrado espacial y algoritmos de orientación

3.2.2.4.1. Introducción

El modelo emplea varios filtros orientados que se intentan evitar en sistemas de tiempo real debido a la alta carga de convoluciones. La motivación de éste apartado es estructurar las operaciones de forma adecuada para desarrollar implementaciones rápidas.

Existen estrategias de interpolación de orientaciones, realizando estos métodos la síntesis de un filtro F^{θ} con una orientación q a partir de un banco de filtros básicos. Una vez que se crea el conjunto básico de filtros, se puede sintetizar un filtro de orientación arbitraria a partir de una combinación lineal de filtros básicos, donde la principal ventaja del algoritmo de orientación viene dada por la linealidad de la convolución. Consecuentemente, siendo posible sintetizar el filtro orientado F de su conjunto básico de filtros, también será posible sintetizar la respuesta orientada F^{θ} a partir de un conjunto básico de respuestas $R_1 \ldots R_n$.

$$R^q = F^q * I = (K_1 F_1 + K_2 F_2 + \ldots + K_n F_n) * I = K_1 F_1 * I + K_2 F_2 * I + \ldots + K_n F_n * I = \\ = K_1 R_1 + K_2 R_2 + \ldots + K_n R_n \tag{3.21}$$

Donde $K_1, \ldots, K_n$ son los pesos que multiplican las diferentes funciones básicas para dar lugar a funciones orientadas. I es la imagen de entrada y el superíndice q denota el ángulo de orientación del filtro.

Por consiguiente, es posible evitar el cálculo explícito de las convoluciones con filtros de distintas orientaciones si se efectúa directamente el cálculo de la respuesta del filtro orientado a partir de un conjunto básico de respuestas. Incluso si se construye una suma ponderada de funciones muestreadas espacialmente, sería equivalente a muestrear espacialmente la suma pesada de funciones continuas básicas, por lo tanto, después del muestreo espacial, todas las funciones se orientan perfectamente. El método más eficiente para sintetizar todos los filtros Gaussianos orientados, se diseña a partir de un banco de filtros separables, ya que éstos requieren menos operaciones para ser implementados, como se justifica en la siguiente sección.

3.2.2.4.2. Desarrollo de filtros orientados

En el contexto computacional, los filtros separables son deseables y ventajosos, ya que son necesarios para generar un conjunto básico de respuestas.

Este proceso tiene que ser compatible con que el proceso de orientación mantenga unas cotas de eficiencia adecuadas. Se va a justificar y explicar la construcción de una derivada orientada de un filtro a partir de un banco de filtros separables, en virtud de la ecuación (3.21). Se cumplirá también para las respuestas $R_1 ... R_n$. Para determinar qué filtros básicos son necesarios para construir un filtro orientado derivativo, se procede a reescribir la derivada direccional en su forma más general:

$$Df(\vec{p}) = \nabla f(\vec{p})\vec{u} \tag{3.22}$$

donde *D* es el operador derivativo actuando sobre la función *f(p)* en la dirección dada por el vector unitario *u*, ∇ es el operador nabla, que actúa como gradiente. Es importante notar que *f(p)* es una función simétrica, y se rotarán las derivadas de esta función (no la función en sí). En cuanto al vector unitario *u* puede ser expresado en términos de orientación θ con respecto al eje x de un sistema de coordenadas cartesianas:

$$\overline{u}(\theta) = \hat{i}\cos\theta + \hat{j}\sin\theta \tag{3.23}$$

siendo *i* y *j* los vectores unitarios en los ejes coordenados, se define ahora un vector ortogonal unitario a *u*:

$$\overline{u}_{\perp}(\theta) = -\hat{i}\sin\theta + \hat{j}\cos\theta \tag{3.24}$$

Con estas definiciones preliminares, es posible determinar la derivada de primer orden de la función en alguna dirección, expresada como una suma escalar de derivadas en las direcciones *x, y*. Ejemplo, la derivada de primer orden de la Gaussiana *G(x,y)* en la dirección *u(θ)* vendrá dada por la expresión siguiente:

$$\begin{aligned} &DG(x,y) = \nabla G(x,y)\vec{u}(q) = \\ &= \left[\frac{\partial G(x,y)}{\partial x}\hat{i} + \frac{\partial G(x,y)}{\partial x}\hat{j}\right]\left[\hat{i}\cos q + \hat{j}senq\right] = \\ &= \left[\frac{\partial G(x,y)}{\partial x}\cos q + \frac{\partial G(x,y)}{\partial x}\sin q\right] \end{aligned} \tag{3.25}$$

Es decir, la primera derivada con respecto a *x*, rotada según sentido antihorario un ángulo que puede ser expresado como una suma pesada de las primeras derivadas:

$$DG(x,y)\overline{u} = G_1^{\theta}(x,y) = k_1 G_x(x,y) + k_2 G_y(x,y) \tag{3.26}$$

donde $k_1=cos\theta$, $k_2=sen\theta$. Esto se clarifica en la figura 3.10, donde se ilustra un desarrollo de filtros orientados de primer orden así como las contribuciones propias de cada filtro de la base, sin embargo para órdenes superiores, la situación merece un análisis propio.

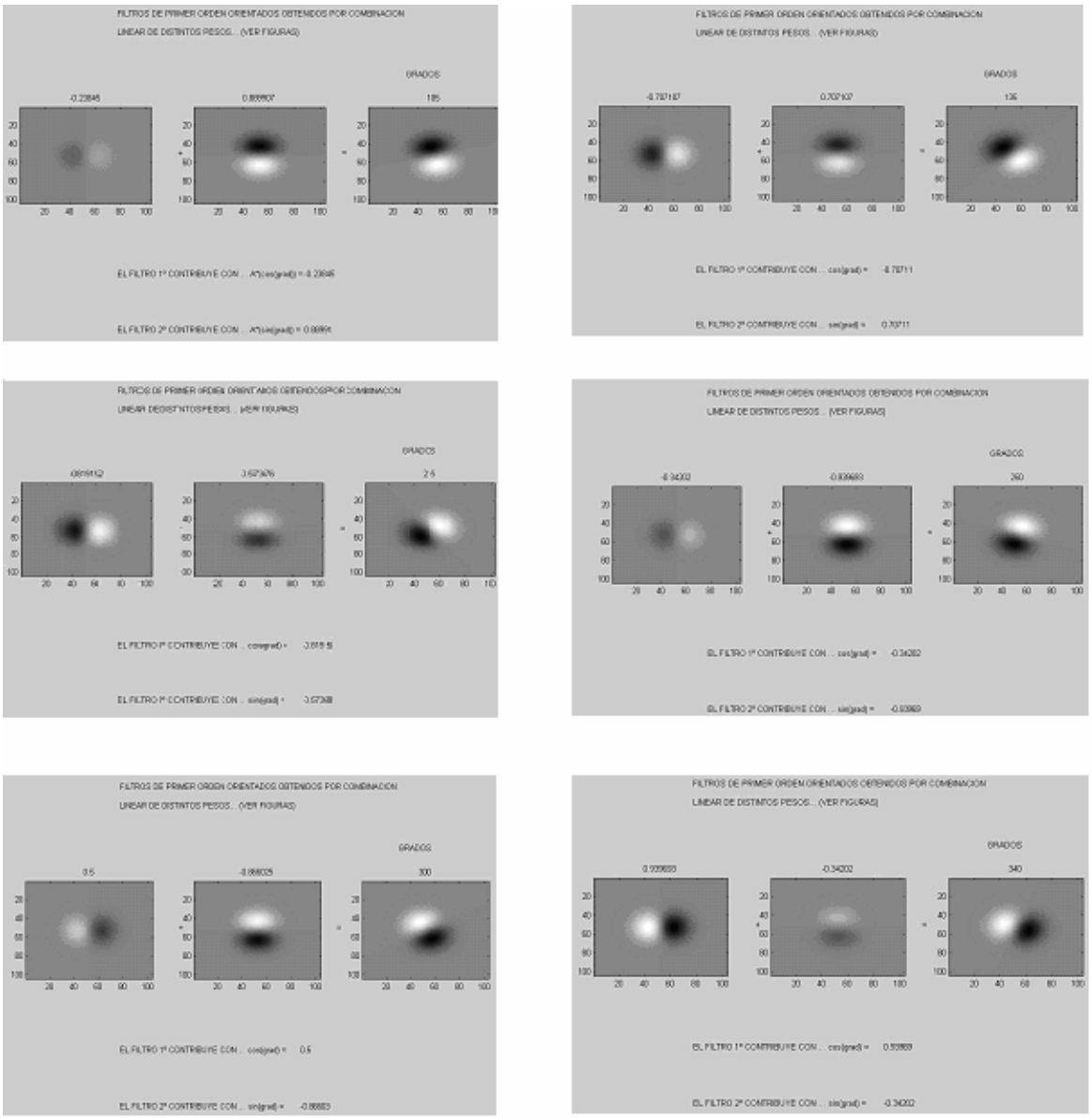


Figura 3.10. Varios ejemplos de rotación *(Steering)* de filtros orientados, para varios ángulos avanzando según el sentido antihorario. Esquematizando de arriba a abajo e izquierda a derecha, la implementación de filtros de 1er orden para 105º, 135º, 215º, 250º, 305º y 340 º respectivamente. (Cada subfigura se expresa mediante las contribuciones individuales de los elementos de su base).

3.2.2.4.3. Orientación de órdenes superiores

Es posible sustituir la derivada direccional anterior para obtener núcleos derivativos orientados de órdenes superiores. Para ello se define una nueva función:

$$f_1(\vec{p}) = \nabla f(\vec{p})\vec{u} \tag{3.27}$$

Aplicando derivadas de órdenes superiores:

$$\begin{aligned} &D_2 f(\vec{p}) = D(Df_1(\vec{p})) = \nabla f_1(\vec{p})\vec{u} = \\ &= \nabla(\nabla(f(\vec{p})\vec{u})\vec{u}) \end{aligned} \tag{3.28}$$

donde es posible continuar inductivamente este proceso a partir de sucesivas sustituciones posibilitando este mecanismo el cálculo de una derivada orientada de cualquier orden. Por ejemplo la derivada de 2º orden de la Gaussiana *G(x,y)* en la dirección *u(θ)* viene dada por la expresión siguiente:

$$D_2 G(x,y) = \nabla(\nabla(G(x,y)\vec{u})\vec{u}) \tag{3.29}$$

sustituyendo la ecuación (3.25) para el término entre paréntesis:

$$\begin{aligned} &D_2 G(x,y) = \nabla\left[\frac{\partial G(x,y)}{\partial x}\cos q + \frac{\partial G(x,y)}{\partial x}\sin q\right]\hat{u} = \\ &= \hat{i}\left[\left[G_{2x}\cos q + G_{yx}\sin q\right] + \hat{j}\left[G_{xy}\cos q + G_{2y}\sin q\right]\right]\left(\hat{i}\sin q + \hat{j}\cos q\right) = \\ &= G_{2x}\cos q^2 + G_{yx}\sin q\cos q + G_{xy}\cos q\sin q + G_{2y}\sin q^2 \end{aligned} \tag{3.30}$$

Existe una invarianza de este proceso frente al orden de derivación cruzado, tal y como se expresa en la igualdad (3.31):

$$G_{xy} = G_{yx} \tag{3.31}$$

a partir de aquí y teniendo en cuenta que $k_1=cos(q)$ y $k_2=sen(q)$:

$$D_2 G(x,y) = G^q{}_{2x} = K_1^2 G_{2x} + 2K_1K_2 G_{yx} + K_2^2 G_{2y} \tag{3.32}$$

se llega a una expresión que produce la segunda derivada de la Gaussiana rotada a través de un ángulo general, mediante una superposición lineal de 3 derivadas separables.

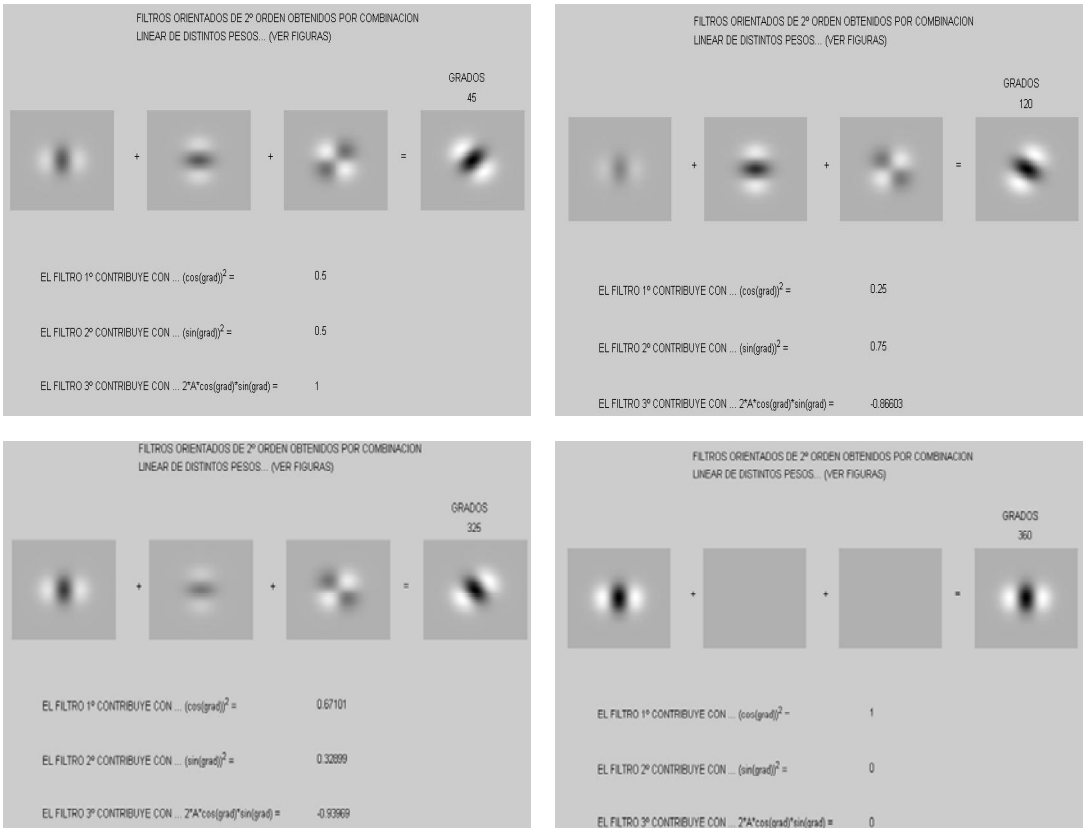


Figura 3.11. Ejemplos de rotación , para filtros de 2º orden, según el sentido horario. Implementación de filtros según 45 (arriba, izquierda), 120 (arriba, derecha), 325(abajo izquierda), 360 º (abajo, derecha) respectivamente.(Para cada subfigura se muestra la contribución individual).

En la figura anterior, se manifiestan varios ejemplos de rotaciones de segundo orden, aunque esta vez en sentido horario, expresándose de nuevo, las contribuciones individuales de cada filtro.

3.2.2.4.4. Derivadas cruzadas

Las derivadas cruzadas como $G_{xy}(x,y)$ requieren el uso de ambos vectores ortogonales $u(\theta)$, $u_{\perp}(\theta)$ y ambas derivadas. El proceso es análogo al realizado en para las derivadas usuales, construyendo la ecuación (3.33), de forma general:

$$D(D(G(x,y))) = \nabla(\nabla(G(x,y)\vec{u}))\vec{u}_{\Lambda} \tag{3.33}$$

De nuevo, es posible sustituir, la ecuación (3.25), para el término entre paréntesis de la ecuación anterior (expresada de forma general), según indica el desarrollo de la expresión (3.34):

$$
\begin{aligned}
&D(D(G(x,y))) = \nabla\left[G_x \cos q + G_x \sin q\right]\vec{u}_{\Lambda} = \\
&= \left(\hat{i}\left(G_{2x}\cos q + G_{yx}\sin q\right) + \hat{j}\left(G_{xy}\cos q + G_{2y}\sin q\right)\right)\left(-\hat{i}\sin q + \hat{j}\cos q\right) = \\
&= -G_{2x}\cos q \sin q - G_{yx}\sin^2 q + G_{xy}\cos^2 q + G_{2y}\cos q \sin q
\end{aligned}
\tag{3.34}
$$

Aplicando la invarianza al orden de diferenciación en la derivada parcial, se llega a la expresión (3.35), donde $k_1=cos(q)$ y $k_2=sin(q)$.

$$
D(D(G(x,y))) = G_{xy}^{q} = -K_1 K_2 G_{2x} + (K_1^2 - K_2^2) G_{yx} + K_1 K_2 G_{2y} \tag{3.35}
$$

Se puede identificar simetrías en todo este proceso de construcción si se avanza cuidadosamente, nótese que se requieren las mismas derivadas de segundo orden para generar respuestas orientadas tanto para G_{2x} como G_{xy} con sólo cambiar los pesos relativos de las funciones básicas. Al generalizar este proceso se encuentra la expresión (3.36), que recoge todas la derivadas que se necesitan para generar la respuesta orientada a partir de un orden total *D*.

$$
G_D^q = \sum_{i=0}^{D} k_i G_{ix(D-i)y} \tag{3.36}
$$

Por ejemplo, si se orienta el tercer orden de la derivada de la Gaussiana $G_{2yx}(x,y)$ es necesario el uso de los siguientes filtros separables G_{3x}, G_{2xy}, G_{x2y}, G_{3y}, ,G_{y2x} mostrándose en la figuras 3.11(a) y 3.11(b) cada una de las componentes individuales (las funciones básicas) y el filtro sintetizado a partir de ellas.

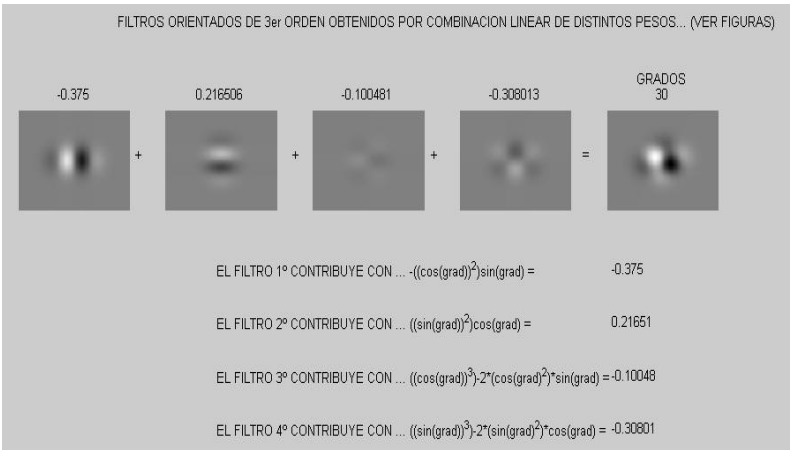


Figura 3.11(a). Ejemplos de rotación, para filtros de orden 2, según el sentido horario. Implementación de filtros según 45 º.(Para cada subfigura se muestra la contribución individual).

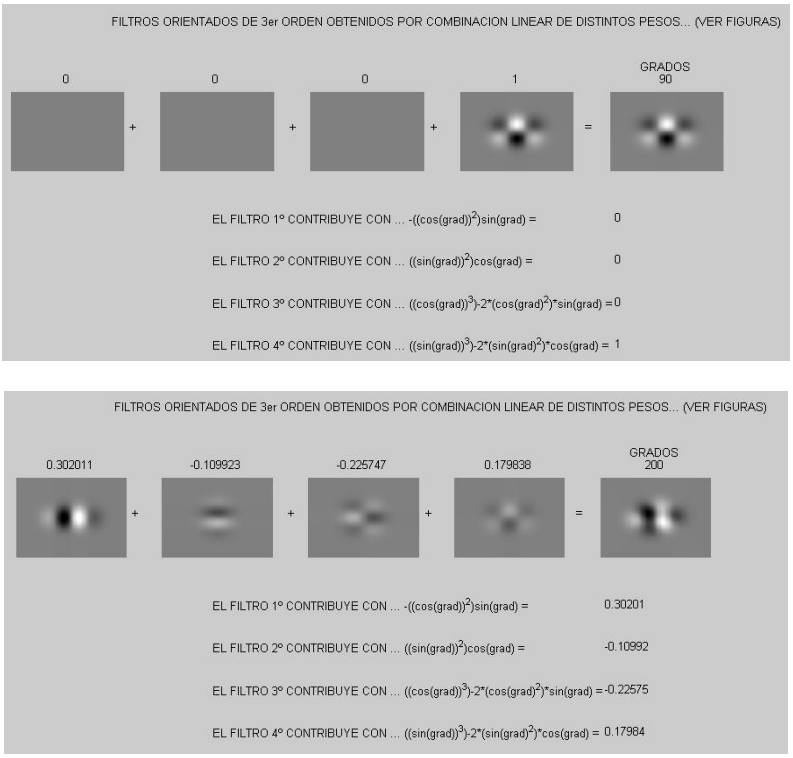


Figura 3.11(b). Ejemplos de rotación , para filtros de orden 2, según el sentido horario. Implementación de filtros según 90º y 200 º.(Para cada subfigura se muestra la contribución individual).

3.2.2.4.5. Expresión general

Después de haber examinado las expresiones anteriores, se observa que es posible extrapolar el tipo de expresiones anteriores, a una del tipo siguiente:

$$G_{n,m}^{\theta}(x,y) = \left(D_x \cos\theta + D_y \sin\theta\right)^n \left(-D_x \sin\theta + D_y \cos\theta\right)^m G_0(x,y) \tag{3.37}$$

siendo $G_{n,m}^{\theta}(x,y)$ el filtro orientado, D_x la derivada en x, D_y la derivada en y, $G_0(x,y)$ el núcleo Gaussiano y θ el ángulo de orientación. Introduciendo la expresión binomial del álgebra elemental:

$$\left(a+b\right)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k} \quad \text{donde} \quad \binom{n}{k} \equiv \frac{n!}{(n-k)!k!} \tag{3.38}$$

se obtiene en la expresión general (3.39) que describe la derivada de la Gaussiana asociada a cualquier rotación.

$$G_{n,m}^{\theta}(x,y)=\left[\sum_{k=0}^{n}\binom{n}{k}\left(D_x\cos\theta\right)^k\left(D_y\sin\theta\right)^{n-k}\right]\cdot\left[\sum_{i=0}^{m}\binom{m}{i}\left(-D_x\sin\theta\right)^i\left(D_y\cos\theta\right)^{m-i}\right]G_0 \qquad (3.39)$$

3.2.2.4.6. Complejidad de las diversas alternativas

Se procede a evaluar la reducción en el número de operaciones al usar este conjunto de filtros básicos (mediante una convolución separable, se pasa de n^2 a *2n multiplicaciones*, siendo *n* el tamaño del filtro). Si se opta por realizar una convolución sin ningún tipo de optimización, (denominada *"normal"*) para una imagen L^2 y una máscara M^2, con *O* orientaciones, se tienen $M^2L^2O(n+1)$ operaciones para cada filtro de orden *n*. Al generar una base de funciones (para orientarlas posteriormente según se ha explicado en 3.2.2.4.2), se necesitarán $L^2M^2(n+1)$ operaciones. Si la base se genera de forma separable, se usan $L^2 2M(n+1)$ operaciones. Por último, la etapa de orientación (*Steering*), consume $L^2(n+1)$ operaciones.

En la Tabla 3.1 se realiza una compararación del proceso de convolución normal y el realizado con la generación de la base y la etapa de orientación, mostrando gráficamente mediante las figuras 3.13(a), 3.13(b) los resultados obtenidos.

TABLA 3.1. Comparativa (imagen de 512×512, un filtro de tamaño 23×23 y orientaciones cada 15º) de los recursos consumidos según las diferentes alternativas de convolución. Haciendo una suma final de todos los recursos globales. Los recursos están medidos en millones de operaciones.

Orden Filtro	*Conv. Normal*	*Generación de la Base*		*Orientación*	*Orientación + Generación Normal*	*Orientación + Generación Separable*
		Normal	*Separable*			
1	6656	277	24	13	290	37
2	9985	416	36	19	435	55
3	13313	555	48	25	580	73
3	16641	693	60	31	734	91
5	19969	832	72	38	870	110
6	23297	971	84	44	1014	128
Total	89861	3744	326	170	3914	496
%	100.00	4.2	0.35	0.1	4.35	0.55

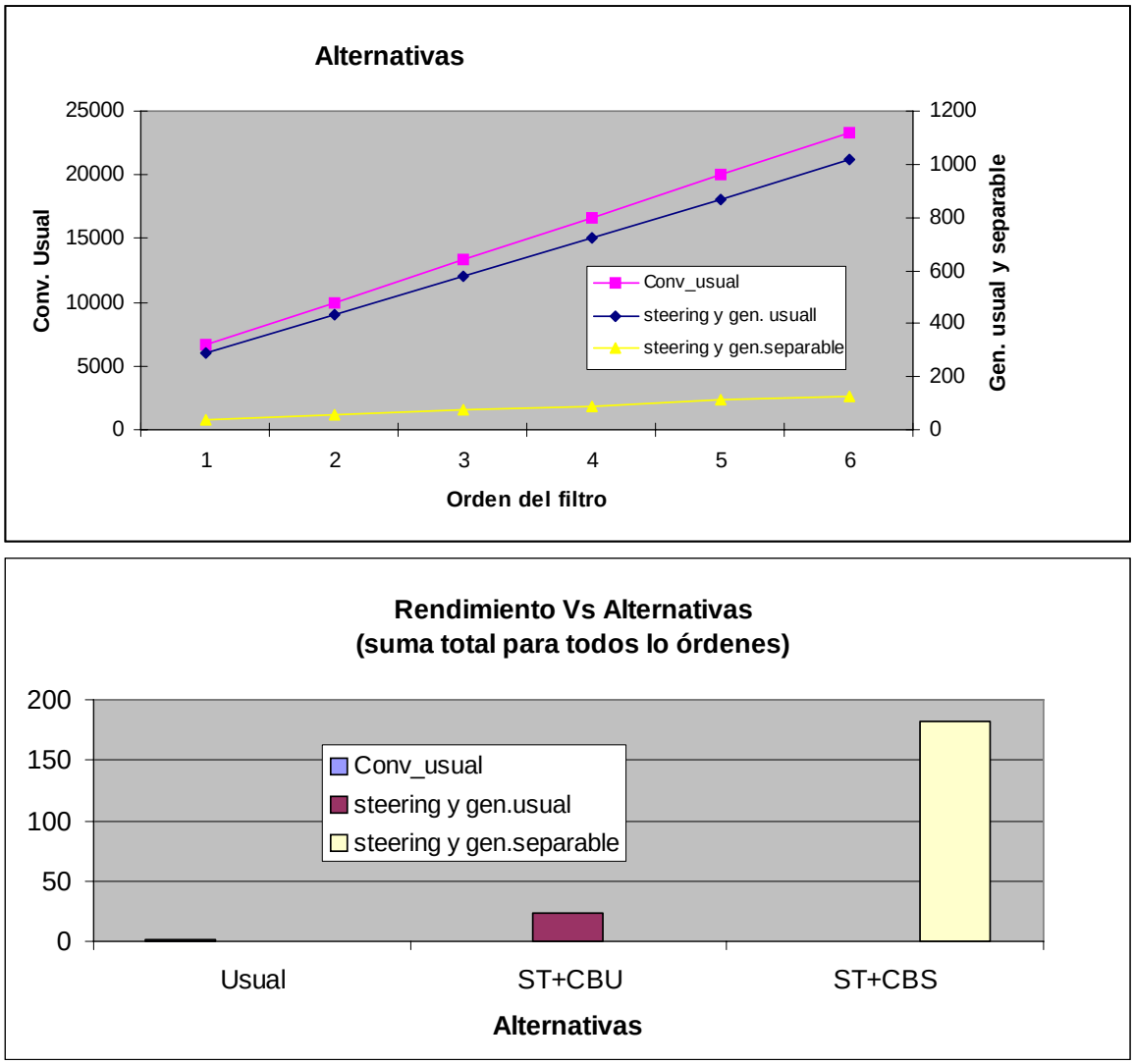


Figura 3.13(a) superior. 3.13(b) inferior. Evolución del número de operaciones (medidas en millones) en función del orden del filtro para diversas alternativas (superior). Incremento de velocidad), para la suma de todas las operaciones en todos los filtros. ST "Steering", CBU "Contribución de la Base Usual", CBS "Contribución de la Base Separable" (inferior).

3.2.2.4.7. Codificación de los pesos de las derivadas orientadas

Es necesario almacenar los pesos de todas las derivadas calculadas para cada una de las orientaciones. Para la codificación de estos pesos, se proporciona la posición dentro de la matriz piramidal según se avanza por líneas diagonales de la parte superior izquierda hacía la parte inferior, tal y como aparece en el esquema de la figura siguiente.

Véase a modo de ejemplo, como encontrar las coordenadas en el esquema de la figura 3.14 para la obtención de una derivada arbitraria (se elije como ejemplo una orientación de 135º (expresión 3.40), respecto a los ejes de referencia, en sentido horario).

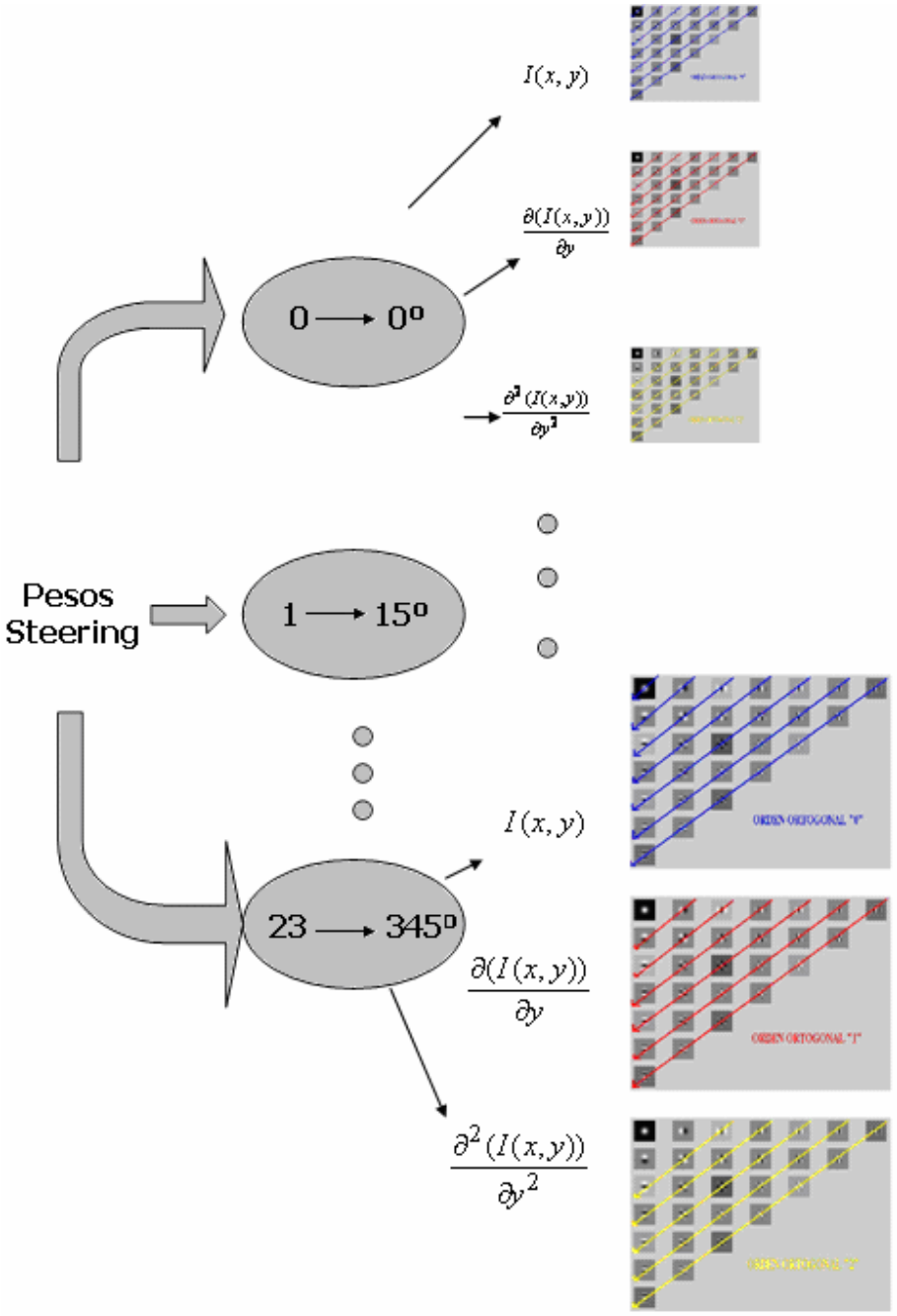


Figura 3.14.. Esquema donde aparece la estructura utilizada en la codificación de pesos de distintos órdenes aplicada en el algoritmo.

$$\left.\frac{\partial^5(I(x,y))}{\partial y^2 \partial x^3}\right|_{135^\circ} \tag{3.40}$$

Según el diagrama anterior 135=15·9, (suponiendo una elección de 24 orientaciones espaciales) consecuentemente hay que situarse en el índice noveno de la tabla donde esté almacenado.

Una vez allí, como se quiere calcular la derivada ortogonal de segundo orden, hay que fijarse en la estructura piramidal formada por líneas amarillas. Localizada ésta, se avanza a partir de la fila segunda (como se empieza a contar en cero, ordinalmente es la tercera), se busca la columna tercera (ordinalmente, la cuarta).

Los pesos que se necesitan, son los almacenados a lo largo de la diagonal amarilla que atraviesa este punto cuyas coordenadas (fila, columna) se han hallado según los órdenes de sus derivadas. Aplicando este proceso se llega a la expresión (3.41).

$$\left[\frac{\partial^5(I(x,y))}{\partial y^2 \partial x^3}\right]_{135^o} = k1\frac{\partial^5(I(x,y))}{\partial x^5} + k2\frac{\partial^5(I(x,y))}{\partial y\ \partial x^4} + k3\frac{\partial^5(I(x,y))}{\partial y^2 \partial x^3} + k4\frac{\partial^5(I(x,y))}{\partial y^3 \partial x^2} + k5\frac{\partial^5(I(x,y))}{\partial y^4 \partial x} + k6\frac{\partial^5(I(x,y))}{\partial y^5} \quad (3.41)$$

A lo largo de esa diagonal se almacenan los pesos $k_1 \cdot\cdot k_6$. Esta tabla tiene la función de guardar todos los pesos para poder expresar todas las derivadas cuyo orden esté comprendido entre 0 y 6 para la x, 0 y 2 para la y. Se tienen pues 24·(28+27+25)=1920 valores para expresar la orientación de 21 funciones (orden espacial 0…6 para x , 0..2 para y) a lo largo de 24 ángulos. A su vez, estos 1920 valores para una imagen con una resolución de 384×128 precisan más de 350 Mbytes de memoria en simple precisión. De esta forma se

$$\sum_{t=0}^{2}\sum_{orden_x=0}^{6}\sum_{columna=0}^{(TAMAÑO-conv)}\sum_{fila=0}^{(TAMAÑO-conv)}\sum_{c=0}^{(orden\_filtro+1)}\{...\}$$

$$\{...\} = peso(angulo, orden\_ortogonal, posición + c)\cdot imagen(t, posicion + c, fila, columna) \quad (3.42)$$

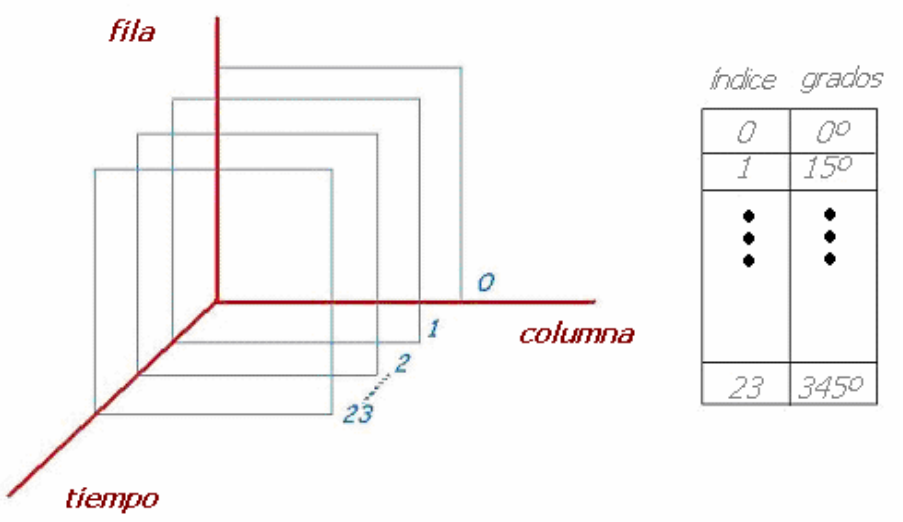


Figura 3.15. Correspondencia entre índice y orientación.

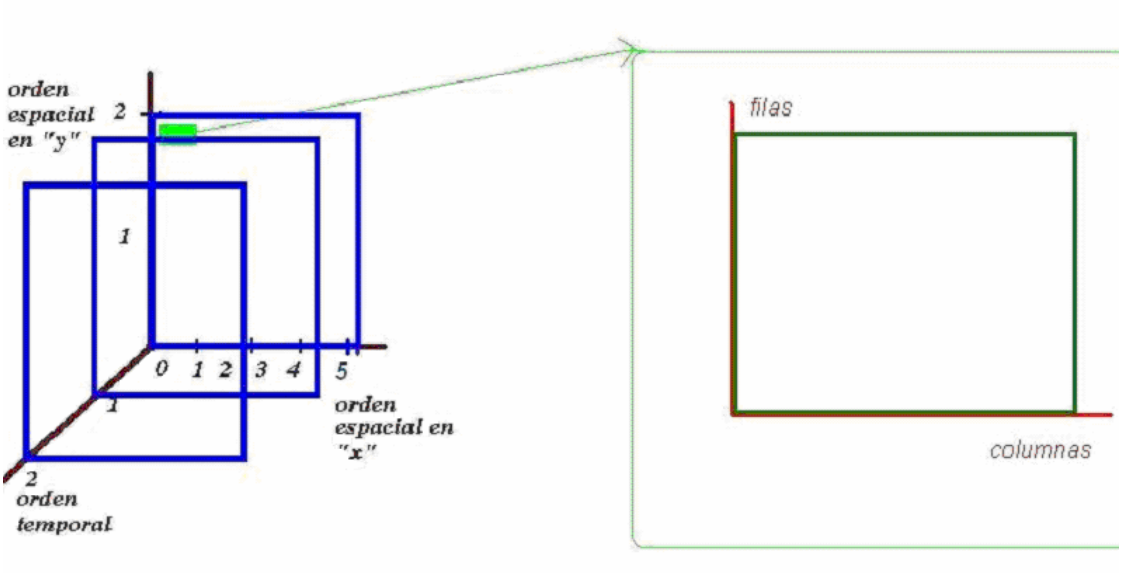


Figura 3.16. Estructura espacio-temporal diseñada al efecto para guardar las diferentes convoluciones orientadas.

consiguen todas las derivadas temporales de orden *0,1,2*, las espaciales en *y* de orden *0,1,2*, las espaciales en x de orden *0…6*, para cualquier orientación (índice entre *0…23,* según la figura 3.15).

Como se ha mostrado de forma previa, gracias a aplicar todos los operadores derivativos de diferente orden a la imagen, se tiene un conjunto de respuestas básicas, que combinadas linealmente, generarán respuestas orientadas (mediante la linealidad de la convolución, son a su vez, respuestas a filtros orientados). Los resultados se guardan en la estructura de la figura 3.16, donde aparecen tres ejes que representan las variables *x*, *y*, *t.* La información se almacena por celdillas individuales, cada una de estas celdillas está bivaluada para la fila y la columna del fotograma a tratar. Como se puede apreciar, el acceso a cada componente espacial o temporal y dentro de éste, a cada fila y columna asociada a cada respuesta orientada se hace de forma totalmente independiente.

3.2.2.4.8. Generación de pesos en el desarrollo de Taylor 3D

Hasta aquí, se ha descrito cómo calcular y justificar todas las derivadas orientadas. Se restringirá el desarrollo de Taylor a un orden 6 total, que implica calcular derivadas en *x* de orden 0 hasta 5, en *y* de orden 0 hasta 2 y en *t* de orden 0 hasta 2. Como el

objetivo es construir un desarrollos de Taylor (3.7) pero múltiplemente orientado, es necesario asignar los pesos asociados (p,q,r,p^2,q^2,r^2 ...) a cada derivada calculada.

Por lo tanto, se almacenarán todos los pesos del desarrollo de Taylor tridimensional correspondiente a un volumen de integración (en este caso esférico), es decir todos los pesos de las derivadas de orden *0...4* en x , *0,1*en y ,*0,1* en t, correspondiente al desarrollo de Taylor 3D de una zona en $x=0...11$, $y=0...11$, $t=0...11$.

3.2.2.5. Desarrollo de Taylor completo. Generación de vectores de filtros

Con este número de derivadas, serán 20 los términos que se obtienen (las derivadas del único término de orden 0, de los tres términos de orden 1, de los cuatro términos de orden 2, de los cuatro términos de orden 3, de los cuatro términos de orden 3, de los tres términos de orden 5 y del único término de orden 6), como aparece en (3.43), (3.44), véase también (3.7).

A su vez, se han generado 24 desarrollos de Taylor, cada 15 º hasta cubrir todo el espacio. Ahora, siguiendo el algoritmo se abordará la agrupación de los términos de cada desarrollo de Taylor, en conjuntos del mismo orden de diferenciación total.

$$I(x+p,y+q,t+r) = Orden_0(1_{tér\min o}) + Orden_1(3_{tér\min os}) + Orden_2(4_{tér\min os}) + \\ + Orden_3(4_{tér\min os}) + Orden_4(4_{tér\min os}) + Orden_5(3_{tér\min os}) + Orden_6(1_{tér\min o}) \tag{3.43}$$

$$\begin{aligned} Vector_{\ nucleo\ \_de\_\ filtros}(x,y,t) = \{ & I(x,y,t), \\ & \frac{\partial(I(x,y,t))}{\partial x}, \frac{\partial(I(x,y,t))}{\partial y}, \frac{\partial(I(x,y,t))}{\partial t}, \\ & \frac{\partial^2(I(x,y,t))}{\partial x^2}, \frac{\partial^2(I(x,y,t))}{\partial y \partial x}, \frac{\partial^2(I(x,y,t))}{\partial t \partial x}, \frac{\partial^2(I(x,y,t))}{\partial y \partial t}, \\ & \frac{\partial^3(I(x,y,t))}{\partial x^3}, \frac{\partial^3(I(x,y,t))}{\partial y \partial x^2}, \frac{\partial^3(I(x,y,t))}{\partial t \partial x^2}, \frac{\partial^3(I(x,y,t))}{\partial t \partial y \partial x}, \\ & \frac{\partial^4(I(x,y,t))}{\partial x^4}, \frac{\partial^4(I(x,y,t))}{\partial y \partial x^3}, \frac{\partial^4(I(x,y,t))}{\partial t \partial x^3}, \frac{\partial^4(I(x,y,t))}{\partial y \partial t \partial x^2}, \\ & \frac{\partial^5(I(x,y,t))}{\partial y \partial x^4}, \frac{\partial^5(I(x,y,t))}{\partial t \partial x^4}, \frac{\partial(I(x,y,t))}{\partial t \partial y \partial x^3}, \\ & \frac{\partial^6(I(x,y,t))}{\partial y \partial t \partial x^4}, \} \end{aligned} \tag{3.44}$$

El vector mostrado en (3.44) se denomina núcleo de filtros .En la expresión anterior se desglosan cada uno de los términos $orden_i$ donde $i=0...6$, (faltan los coeficientes que acompañan a cada derivada, que serán almacenados aparte), a partir de él se generan todas las funciones necesarias para el cálculo del flujo gracias a sus derivadas respecto a *x, y, t.* Las derivadas de este núcleo se pueden clasificar mediante las expresiones (3.45) y (3.46) para conformar los vectores K_x, $K_{y,}$ K_t.

$$J = D\{I(x+p, y+q, t+r)\} = \{Kx, Ky, Kt\}^T \tag{3.45}$$

$$Kx = \frac{\partial\{\sum_{i=0}^{6} Orden_i(x,y,t)\}}{\partial x} \quad Ky = \frac{\partial\{\sum_{i=0}^{6} Orden_i(x,y,t)\}}{\partial y} \quad Kt = \frac{\partial\{\sum_{i=0}^{6} Orden_i(x,y,t)\}}{\partial t} \tag{3.46}$$

Cada columna representa el desarrollo de Taylor derivado respecto *x*,*y* o *t*, y cada fila contiene las derivadas del mismo orden. No es necesario implementar el desarrollo de Taylor primero y luego derivar, pues todos los términos derivativos se crean en la etapa de filtrado, esta es la razón por la cual el orden de derivación realizado, siempre tiene un valor superior en una unidad al del desarrollo de Taylor. Una vez calculado *Kx,Ky,Kt*, es necesario realizar la integral del producto escalar de las 6 combinaciones que resultan al tomar las parejas *X·X, X·Y, X·T, Y·Y, Y·T, T·T*.

3.2.3 Etapa de integración

Con esta información, se procede a procesar la matriz producto, según (3.47):

$$J^T J = \begin{bmatrix} KxKx & KxKy & KxKt \\ KyKx & KyKy & KyKt \\ KtKx & KtKy & KtKt \end{bmatrix} \tag{3.47}$$

Es necesario integrar esta matriz a lo largo del volumen espacio-temporal (11×11×11) que se muestra en la figura 3.17. Esta integral se puede calcular sumando los productos internos sobre la extensión del dominio de la operación (hipervolumen escogido de forma simétrica), tal y como indica la expresión (3.48).

$$M = \int_e^f \int_c^d \int_a^b J^T J dp dq dr = \begin{bmatrix} XX & XY & XT \\ YX & YY & YT \\ TX & TY & TT \end{bmatrix} \tag{3.48}$$

$$\Omega = \{a \le p \le b, c \le q \le d, e \le r \le f\} \ \ \forall a,b,c,d,e,f \in R$$

Como estas combinaciones se realizan multiplicando los valores de los vectores, para ello se hace uso de los coeficientes o pesos asociados a cada derivada, siendo el resultado final un producto escalar de vectores diferenciales. Para realizar este proceso, se construye primeramente una matriz de dimensiones 7×7×7. El primer índice es relativo al orden total de diferenciación en la serie de Taylor (*v.* 3.1.8) y el segundo, tercero, corresponde a una matriz cuadrada, donde el número de componentes diferentes de cero son las variaciones con repetición, según versa (3.49).

$$V_{m,n}^{R} = m^{n} \tag{3.49}$$

Siendo *m* el número de elementos del orden considerado en el desarrollo de Taylor y *n* el número de elementos de que tiene cada término producto. Estos términos, contienen las integrales de los productos de los pesos (es decir, los productos de los pesos, sumados para todos los *p,q,r* en todo el volumen de integración espacio-temporal).

Si se realizan algunas definiciones previas con objeto de simplificar los cálculos, expuestas en las (3.50) teniendo en cuenta que cada un a de las matrices definidas según la expresión *matriz(r,i,j),* cuyas componentes están en la expresión (3.51):

$$p(t,y,x)=pesos(p,q,r,t,y,x);\ \ matriz'(r,i,j)=\sum_{r=-5}^{5}\sum_{q=-5}^{5}\sum_{p=-5}^{5} matriz(r,i,j);\ \ \forall r=1...7 \tag{3.50}$$

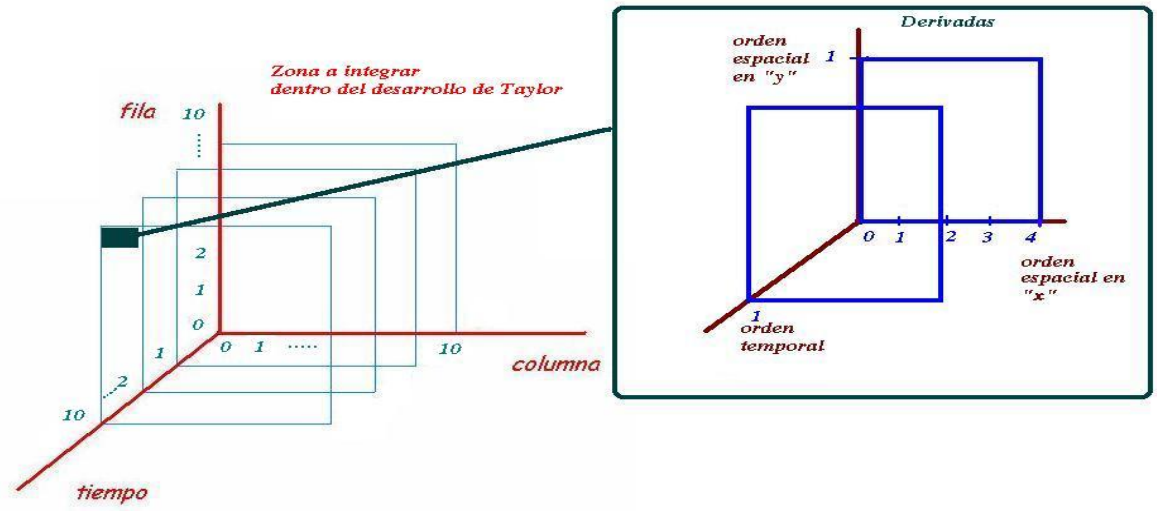


Figura 3.17. Hipervolumen de integración donde se aprecia cómo cada celda, está formada a su vez por un volumen derivativo en (*x,y,t*), habilitando el acceso a la toda la información requerida de forma eficiente . Nota: Se ha elegido un hipervolumen de 11^{3} para calibrar el modelo a 1º=128 puntos y 1 seg=128 puntos .

$$
matriz(0,i,j)=\begin{bmatrix}
p(0,0,0)^2 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

$$
matriz(1,i,j)=\begin{bmatrix}
p(0,0,1)^2 & p(0,0,1)p(0,1,0) & p(0,0,1)p(1,0,0) & 0 & 0 & 0 & 0\\
p(0,1,0)p(0,0,1) & p(0,1,0)^2 & p(0,1,0)p(1,0,0) & 0 & 0 & 0 & 0\\
p(1,0,0)p(0,0,1) & p(1,0,0)p(0,1,0) & p(1,0,0)^2 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

$$
matriz(2,i,j)=\begin{bmatrix}
p(0,0,2)^2 & p(0,0,2)p(0,1,1) & p(0,0,2)p(1,0,1) & p(0,0,2)p(1,1,0) & 0 & 0 & 0\\
p(0,1,1)p(0,0,2) & p(0,1,1)^2 & p(0,1,0)p(1,0,1) & p(1,0,1)p(1,1,0) & 0 & 0 & 0\\
p(1,0,1)p(0,0,2) & p(1,0,0)p(0,1,1) & p(1,0,1)^2 & p(1,0,1)p(1,1,0) & 0 & 0 & 0\\
p(1,1,0)p(0,0,2) & p(1,1,0)p(0,1,1) & p(1,1,0)p(1,0,1) & p(1,1,0)^2 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

$$
matriz(3,i,j)=\begin{bmatrix}
p(0,0,3)^2 & p(0,0,3)p(0,1,2) & p(0,0,3)p(1,0,2) & p(0,0,3)p(1,1,1) & 0 & 0 & 0\\
p(0,1,2)p(0,0,3) & p(0,1,2)^2 & p(0,1,2)p(1,0,2) & p(0,1,2)p(1,1,1) & 0 & 0 & 0\\
p(1,0,2)p(0,0,3) & p(1,0,2)p(0,1,2) & p(1,0,2)^2 & p(1,0,2)p(1,1,1) & 0 & 0 & 0\\
p(1,1,1)p(0,0,3) & p(1,1,1)p(0,1,2) & p(1,1,1)p(1,0,2) & p(1,1,1)^2 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\qquad (3.51)
$$

$$
matriz(4,i,j)=\begin{bmatrix}
p(0,0,4)^2 & p(0,0,4)p(0,1,3) & p(0,0,4)p(1,0,3) & p(0,0,4)p(1,1,2) & 0 & 0 & 0\\
p(0,1,3)p(0,0,4) & p(0,1,3)^2 & p(0,1,3)p(1,0,3) & p(0,1,3)p(1,1,2) & 0 & 0 & 0\\
p(1,0,3)p(0,0,4) & p(1,0,3)p(0,1,3) & p(1,0,3)^2 & p(1,0,3)p(1,1,2) & 0 & 0 & 0\\
p(1,1,2)p(0,0,4) & p(1,1,2)p(0,1,3) & p(1,1,2)p(1,0,3) & p(1,1,2)^2 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

$$
matriz(5,i,j)=\begin{bmatrix}
p(0,1,4)^2 & p(0,1,4)p(1,0,4) & p(0,1,4)p(1,1,3) & 0 & 0 & 0 & 0\\
p(1,0,4)p(0,1,4) & p(1,0,4)^2 & p(1,0,4)p(1,1,3) & 0 & 0 & 0 & 0\\
p(1,1,3)p(1,1,4) & p(1,1,3)p(1,0,4) & p(1,1,3)^2 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

$$
matriz(6,i,j)=\begin{bmatrix}
p(4,1,1)^2 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

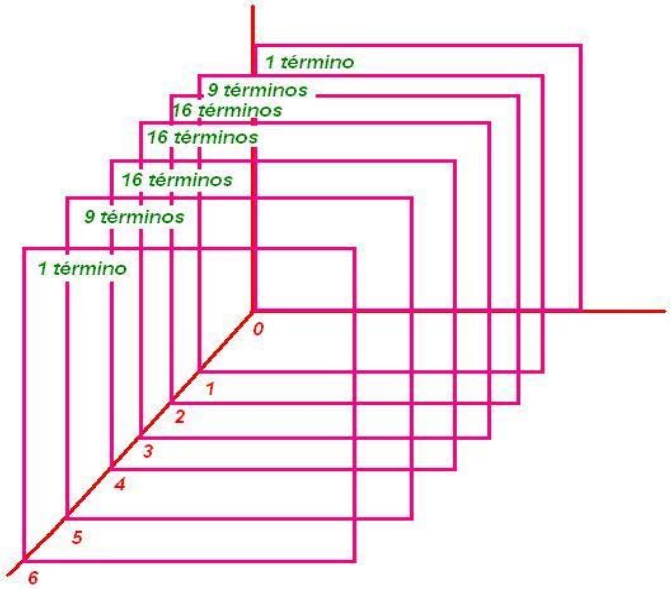


Figura 3.18 Esquema de la matriz construida para almacenar la información en la etapa de integración.

Representándose en la figura 3.18, la estructura de la expresión anterior. A partir del conjunto de matrices anteriores, se obtienen el sexteto de los productos *XX, XY, XT, YY, YT, TT* mediante las operaciones definidas en (3.52).

(3. 52)

$$XT = \sum_{k=o}^{orden} \sum_{i=0}^{m^n} \sum_{j=0}^{m^n} Kx(i)Kt(j)matriz'(k,i,j) \qquad XY = \sum_{k=o}^{orden} \sum_{i=0}^{m^n} \sum_{j=0}^{m^n} Kx(i)Ky(j)matriz'(k,i,j)$$

$$YT = \sum_{k=o}^{orden} \sum_{i=0}^{m^n} \sum_{j=0}^{m^n} Ky(i)Kt(j)matriz'(k,i,j) \qquad YY = \sum_{k=o}^{orden} \sum_{i=0}^{m^n} \sum_{j=0}^{m^n} Ky(i)Ky(j)matriz'(k,i,j)$$

$$TT = \sum_{k=o}^{orden} \sum_{i=0}^{m^n} \sum_{j=0}^{m^n} Kt(i)Kt(j)matriz'(k,i,j) \qquad XX = \sum_{k=o}^{orden} \sum_{i=0}^{m^n} \sum_{j=0}^{m^n} Kx(i)Kx(j)matriz'(k,i,j)$$

Por ejemplo, el producto interno XY indica que los términos en el primer vector son aquellos generados por la diferenciación de la representación básica de la estructura con respecto a *x*, adicionalmente los términos del segundo vector son generados por la diferenciación de la representación básica con respecto a *y*.

3.2.4. Velocidad y velocidad inversa

3.2.4.1. Introducción

Mediante las matrices referidas en (3.52), es posible recuperar las medidas de velocidad de la imagen estimadas en 2 direcciones ortogonales, mediante un cálculo de cocientes según señala la expresión (3.53).

$$velocidad = \frac{X \cdot T}{X \cdot X} \quad velocidad\ ortogonal = \frac{Y \cdot T}{Y \cdot Y} \tag{3.53}$$

Esos cocientes están bien condicionados matemáticamente para una dirección que no sea paralela a los contornos de igual brillo (en ese caso se daría el problema de la apertura), ya que el denominador es una magnitud al cuadrado de un vector (la suma de todas sus componentes al cuadrado). Este producto escalar es cero cuando todos los términos del producto escalar sean cero, (es decir, cuando la imagen es totalmente uniforme). En esta situación se llega a una indeterminación, que será resuelta imponiendo un valor nulo.

Posteriormente, se va a considerar el proceso de obtención de estas medidas de velocidad para un rango de direcciones primarias, (correspondiendo a un rango de direcciones de columnas del sistema visual del primate, *v*. apéndice III), mediante la suma de todas las contribuciones debidas a cada orientación espacial, (24, en este caso), para cada sexteto *X·X, X·Y, X·T, Y·X, Y·Y, T·T*.

3.2.4.2. Origen de estas expresiones

La justificación de las fórmulas anteriores viene de la posibilidad de formación de múltiples estimaciones con los vectores de *X,Y,T* para aumentar la robustez en el cálculo del flujo óptico.

Dados un conjunto de estimadores de un parámetro particular (la velocidad en este caso), se compone un ajuste de mínimos cuadrados para encontrar el valor de *v'* óptimo, que estará basado en las medidas ponderadas del modo en que el brillo de la imagen y sus derivadas están cambiando respecto al espacio y el tiempo.

Se asume que la velocidad es localmente constante, *X* y *T* son linealmente dependientes (paralelos) y por consiguiente *v*' es la longitud relativa entre los dos. La mejor aproximación restringe estos 2 vectores a minimizar la distancia métrica, dada por (3.55).

$$velocidad = \left\| v' X - T \right\| \tag{3.55}$$

La elección de estas ponderaciones, da una idea de la orientación espacio-temporal, que minimiza la medida del cambio del brillo sobre una región a lo largo de un punto de interés. Minimizar la ecuación de (3.55) es lo mismo que minimizar la expresión (3.56).

$$\sum_i (v' x_i - t_i)^2 \tag{3.56}$$

Siendo *i* el índice de los vectores. Para minimizar esta distancia, se tiene que cumplir que *V'X –T* sea perpendicular a *X* , por consiguiente se establece que *(V'·X –T)·X=0* consecuentemente *V'=X·T/X·X*.

Para la velocidad ortogonal el razonamiento es análogo al realizado con la velocidad. (aunque también se puede justificar mediante geometría diferencial la expresión para velocidad ortogonal *Y·T/Y·Y*.).

3.2.4.3. Necesidad de condicionar asintóticamente. Problema de la apertura

Cuando el denominador sea cero (no haya variación en *X* o *Y*) la velocidad toma un valor de 0/0 (estará indeterminada).

Gracias a tener múltiples medidas de la ecuación MCE (expresión 2.5 del capítulo II) y evolucionar desde un núcleo de filtros a un vector de núcleos de filtros, se consiguen condicionar los resultados extremos desde un valor dividido por cero (en el caso de una sola medida) hasta 0/0, (que no tiene un valor concreto). Estos vectores son calculados a lo largo de *m* diferentes orientaciones para cada punto en la imagen. En este caso 24 orientaciones, (que simularán las columnas orientadas en el córtex cerebral estriado).

Llegado a este punto se puede descomponer la velocidad para cada orientación determinada en dos componentes (paralela y perpendicular) según un eje cartesiano, representando esta descomposición, junto con las componentes de translación netas (campos de vectores orientados hacía abajo) en la figura 3.19 de la página siguiente.

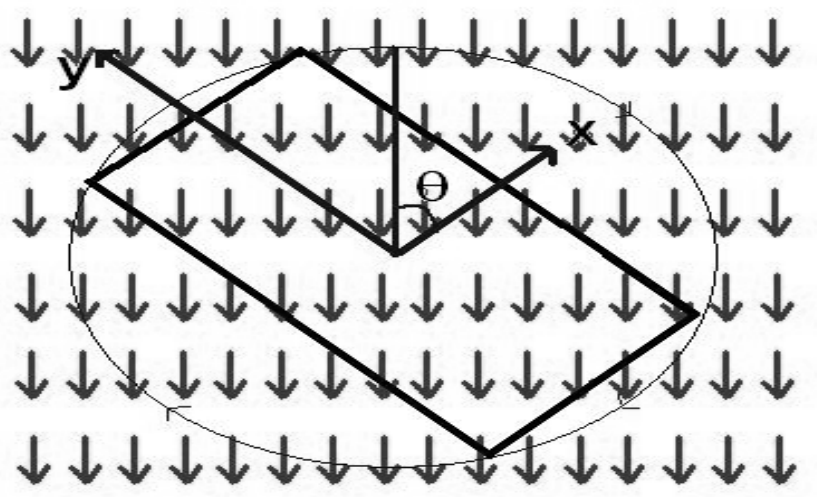


Figura 3.19. Velocidad y velocidad inversa se extraen en un sistema coordenado de orientación determinada relativa a los ejes de referencia. En el modelo se usarán 24 orientaciones. Las medidas de las componentes translacionales puras, (esquematizadas como flechas hacia abajo), varían sinusoidalmente con el ángulo.

A continuación se exponen las expresiones de la velocidad (descompuesta en paralela y ortogonal), normalizada por un factor dependiente del número de orientaciones:

$$\hat{s}(\theta) = (\hat{s}_{\shortparallel}, \hat{s}_{\perp})$$

$$\hat{s}(\theta) = \sqrt{\frac{2}{m}}\left[\frac{X_\theta \cdot T_\theta}{X_\theta \cdot X_\theta}\left(1+\left(\frac{X_\theta \cdot Y_\theta}{X_\theta \cdot X_\theta}\right)^2\right)^{-1}, \frac{Y_\theta \cdot T_\theta}{Y_\theta \cdot Y_\theta}\left(1+\left(\frac{X_\theta \cdot Y_\theta}{Y_\theta \cdot Y_\theta}\right)^2\right)^{-1}\right] \tag{3.57}$$

y su homóloga, la velocidad inversa, (en sus componentes paralela y ortogonal).

$$\breve{s}(\theta) = (\breve{s}_{\shortparallel}, \breve{s}_{\perp})$$

$$\breve{s}(\theta) = \sqrt{\frac{2}{m}}\left[\frac{X_\theta \cdot T_\theta}{T_\theta \cdot T_\theta}, \frac{Y_\theta \cdot T_\theta}{T_\theta \cdot T_\theta}\right] \tag{3.58}$$

Estas cuatro medidas de la velocidad, se calculan a partir de diferentes combinaciones de salidas de filtros que se pueden interpretar como diferentes estimaciones a la velocidad local. (la inclusión de varias estimaciones para añadir robustez es una de las principales aportaciones del presente modelo, se profundiza más sobre este tema en el apéndice II). A pesar de que la velocidad para una sola dirección (no paralela a los contornos de igual brillo), está bien condicionada, el problema de apertura es inevitable cuando se consideran orientaciones [PAJ02], es necesario pues hacer que la velocidad dependa de la estructura orientada de la imagen.

3.2.4.4. Comportamiento asintótico de velocidad y velocidad inversa

Si se piensa en un punto cualquiera perteneciente a un objeto en movimiento, dentro de una ventana espacial (por ejemplo la figura 1.3, del capítulo I) y todos los puntos candidatos a corresponder a éste en el instante temporal siguiente, se pueden aplicar las expresiones (3.57) y (3.58) para estudiar su comportamiento asintótico. En este caso se usa un diagrama polar, (ilustrado en la figura 3.20), considerando las diferentes expresiones de la velocidad, como una funciones de la dirección.

La distancia desde el origen a la línea de puntos, indica velocidades sin condicionar en función de las diferentes componentes direccionales medidas en el diagrama polar, o también la velocidad medida para una sola dirección del estímulo inicial orientado múltiplemente (el problema formalmente es indistinguible). La velocidad bien condicionada se representa por la pareja de círculos concéntricos, velocidad (círculo verde grande) y velocidad inversa (azul pequeño). Se puede ver como en límite, (cuando la línea de puntos llegue hacia 90º -90º), habrá una distancia infinita hasta el origen. En la figuras 3.21, 3.22 se puede representar la velocidad sin condicionar y la velocidad directa e inversa (dependiente de la estructura espacial), pero esta vez en una escala lineal. Las medidas de las diferentes velocidades condicionadas según la estructura de la imagen quedan transformadas en sinusoides según muestra la figura 3.21, en comparación con el conjunto de hipérbolas de cada medida, según muestra la figura 3.22.

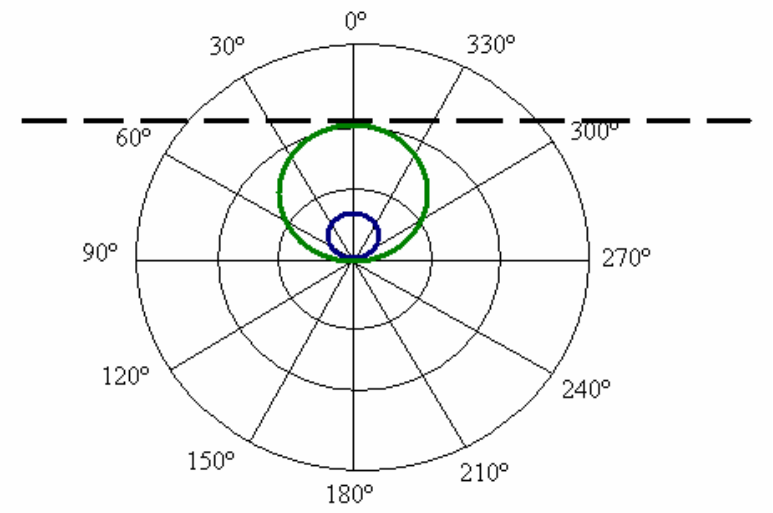


Figura 3.20. Diagrama polar, para apreciar velocidad frente a dirección. La línea punteada indica la velocidad mal condicionada sin ningún añadido (la velocidad es la distancia desde el origen). La velocidad inversa se esquematiza con un circulo pequeño, frente a la velocidad directa (circulo grande).

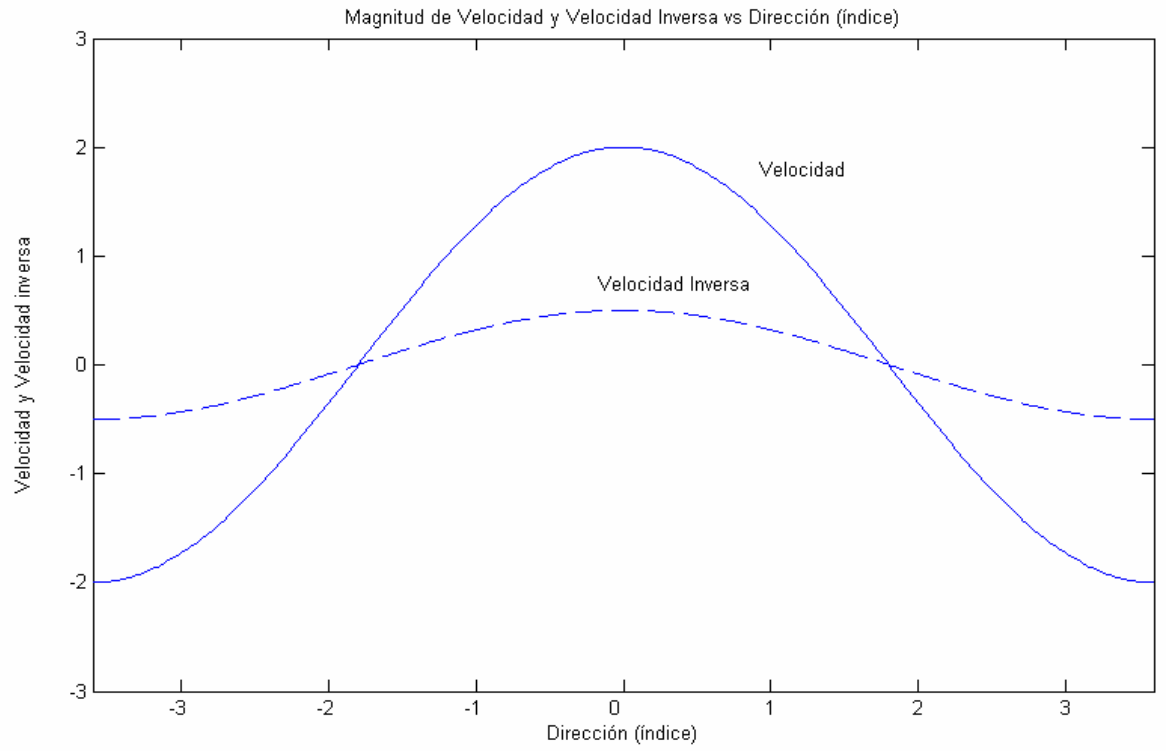


Figura 3.21 Otra forma de representar la información de la figura 3.20. Amplitud grande (pequeña) indica velocidad directa (inversa).

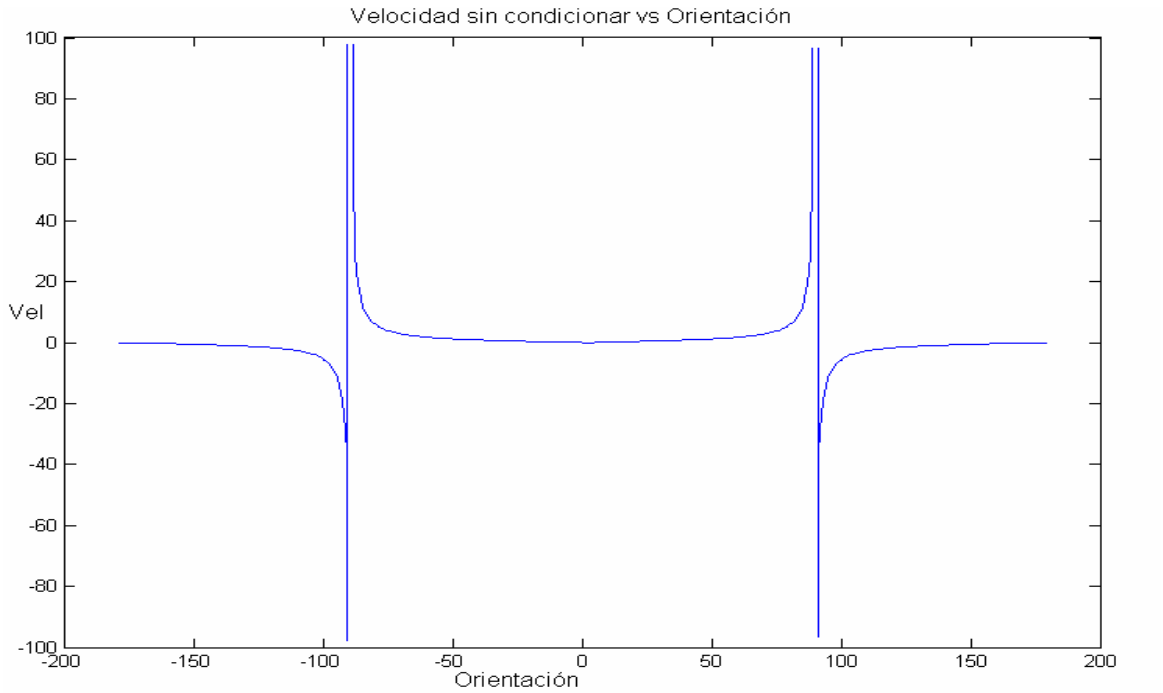


Figura 3.22 Otra forma de representar la información de la figura 3.20, donde se ve que la estimación de la velocidad a lo largo de la línea es infinita.

3.2.4.5. Extracción sólo de componentes de traslación

Koenderink y Van Doorn [KOE76] mostraron que el campo de velocidad puede ser descompuesto principalmente en componentes de translación y componentes diferenciales. Para una translación pura, la suma de las componentes de velocidad directa en el intervalo $[0,\pi]$ será cero, pero para el caso de tener una divergencia, la contribución neta es no nula. Si se quiere recuperar la componente de translación y anular las componentes diferenciales, se tiene que imponer un valor nulo a la integral de las funciones de velocidad, (mediante la extracción de los coeficientes de Fourier), la forma de llevar esto a cabo es gracias a la proyección sobre funciones seno y coseno:

$$\bar{F}(\theta) = ((F_{\parallel}(\theta), F_{\perp}(\theta)) = \sqrt{2/m}\left[\cos(\theta), \sin(\theta)\right] \tag{3.59}$$

donde, es interesante notar que notar que los vectores de la expresión anterior están normalizados por el número de orientaciones a considerar.

3.2.4.6. Cálculo final de la velocidad en módulo y fase

La velocidad final, se calcula como un cociente de determinantes, según la expresión (3.60). Esto tiene su interpretación, el denominador vale 1 en el caso de un movimiento rígido de patrones simples (una retícula sinusoidal trasladándose) y puede variar punto a punto en la imagen para compensar situaciones donde la velocidad y velocidad inversa, no son exactamente inversas.

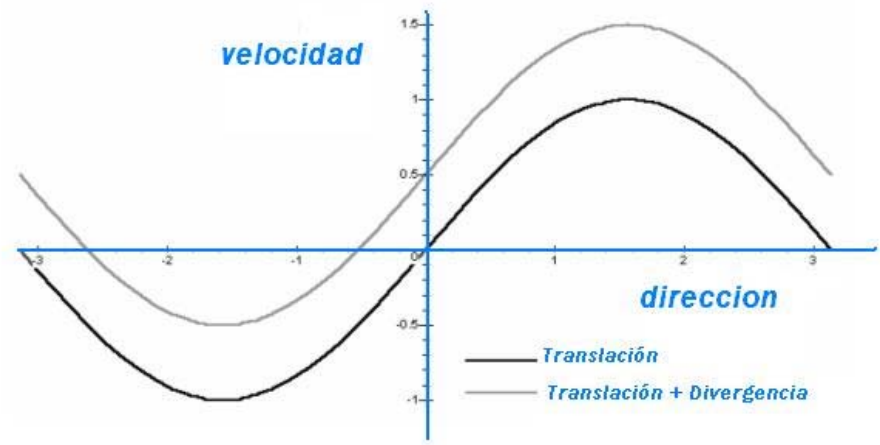


Figura 3.23. En el caso de un movimiento, el caso general tendrá componentes de translación puras además de otras diferenciales. En el caso de una translación pura, las componentes varían sinusoidalmente con la orientación.

$$S^2 = \begin{bmatrix} \widehat{s}_{\text{II}} F_{\text{II}} & \widehat{s}_{\text{II}} F_{\perp} \\ \widehat{s}_{\perp} F_{\text{II}} & \widehat{s}_{\perp} F_{\perp} \\ \hline \widehat{s}_{\text{II}} \breve{s}_{\text{II}} & \widehat{s}_{\text{II}} \breve{s}_{\text{II}} \\ \widehat{s}_{\perp} \breve{s}_{\perp} & \widehat{s}_{\perp} \breve{s}_{\perp} \end{bmatrix} \tag{3.60}$$

Con respecto al denominador, este será nulo para una divergencia pura, aunque en ese caso también sería el numerador, obteniendo una situación de indeterminación. El denominador (y sus medidas de velocidad inversa) se incluyen para estabilizar la estimación final de velocidad, este hecho tiene una especial importancia, cuando el movimiento ocurre en presencia de un entorno donde hay ruido estático.

La dirección de movimiento se extrae calculando una medida de la fase (expresión 3.61) de las funciones de velocidad que se combina con las cuatro medidas relacionadas con la velocidad, mediante las expresiones 3.57 y 3.58.

La orientación del flujo está codificada como la fase de las funciones de velocidad y puede ser extraída explícitamente, proyectando sobre funciones *senos* y *cosenos*, para extraer el componente fundamental de Fourier según indica la expresión 3.59. Debido a que las funciones velocidad y velocidad inversa están en fase (figura 3.21) es posible sumarlas sin problema para mejorar la robustez de la medida.

De forma posterior, se suman también las medidas ortogonales después de que la diferencia relativa de fase haya sido considerada, dando la medida direccional, tal y como apunta la figura 3.24.

$$\textit{Dirección} = \arctan\left(\frac{(\breve{s}_{\text{II}} + \widehat{s}_{\text{II}})F_{\perp} + (\breve{s}_{\perp} + \widehat{s}_{\perp})F_{\text{II}}}{(\breve{s}_{\text{II}} + \widehat{s}_{\text{II}})F_{\text{II}} - (\breve{s}_{\perp} + \widehat{s}_{\perp})F_{\perp}} \right) \tag{3.61}$$

Cuando la velocidad es alta (y la velocidad inversa es baja) la dirección está dominada por las medidas de velocidad, y cuando la velocidad es baja (y la velocidad inversa es alta) entonces es la velocidad inversa la que domina, siendo esta utilización de medidas complementarias y antagónicas, (velocidad y velocidad inversa) realmente útiles para un sistema donde haya elementos externos que afecten al cómputo del flujo como oclusiones o distintos tipos de ruido.

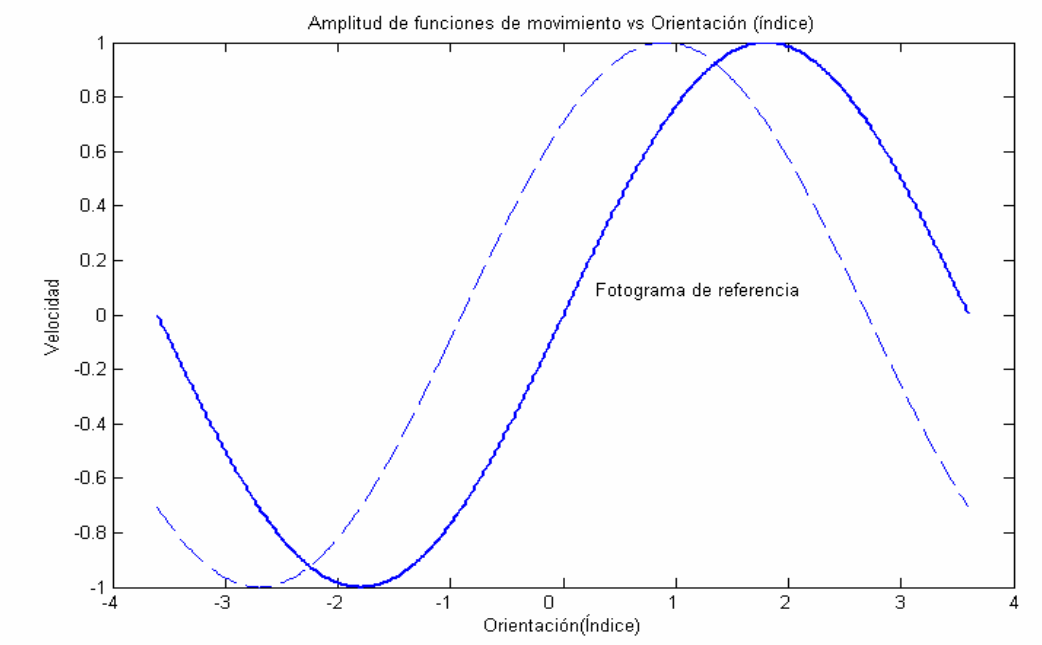


Figura 3.24. Velocidad como amplitud de una función sinusoidal y dirección como fase entre la función velocidad y la referencia punteada.

En resumen, el presente modelo puede considerarse como una extensión estable, robusta y bioinspirada de un algoritmo general de gradiente. El modelo afronta los problemas de incertidumbre matemática realizando múltiples medidas del estímulo (basadas en derivadas de órdenes superiores) en vez de introducir constantes o umbrales arbitrarios. Hay que notar también que el modelo no contiene etapas de pre-proceso de naturaleza no lineal que intenten extraer rasgos o texturas del análisis de movimiento. El cálculo de la velocidad y la dirección se hace directamente a partir de los datos de los filtros aplicados a la imagen considerada.

3.3. Conclusiones

Se ha descrito, justificado e implementado eficientemente en C++ un método de estimación de flujo óptico computacionalmente robusto (McGM [JOH99]).

Si se desea ver algunos ejemplos breves de salidas procesadas (correspondientes a secuencias reales) se puede consultar el apéndice IV, por otra parte, para ver un análisis por el que el modelo completo no se ve afectado por la presencia de un patrón estático, estudiar su

posible degradación a modelos de gradiente usuales y para consultar un completo diagrama de flujo del algoritmo se puede consultar el apéndice II.

El presente modelo procesa velocidad direccional directa e inversa a partir de las derivadas de la estructura espacio-temporal de la imagen (representada por un desarrollo truncado de Taylor) y combina esas medidas para dar un cociente de determinantes.

Todas las divisiones usadas para extraer estas medidas direccionales incluyen la proyección de un vector derivado parcialmente sobre otro, estando estas operaciones auto-normalizadas respecto al contraste y por consiguiente, no se requiere un proceso activo de control de ganancia para estabilizar el sistema. La velocidad final de computación también implica un cociente, pero los factores que afectan a las funciones de velocidad tendrán la misma influencia en el denominador y numerador, de forma que el cociente eliminaría esta dependencia.

¿Por qué se incluyen éstas medidas de funciones de velocidad si no dependen al final de ellas?, esta arquitectura es el secreto de su robustez, (cuando la velocidad directa no aporta información, lo hace la velocidad inversa y viceversa), adicionalmente, usando esta filosofía se evitan situaciones de singularidad matemática que no condicionan adecuadamente el problema.

Los dispositivos de visión artificial ideados por el hombre están lejos de conseguir una eficiencia y exactitud en el análisis de escenas visuales como la verificada en animales vertebrados. El modelo multicanal de gradiente (McGM) [JOH95], es un algoritmo de extracción de flujo óptico con unas serie de cualidades expuestas a continuación.

- Bioinspiración, pues hace uso de propiedades intrínsecas del córtex visual estriado de los mamíferos en general.
- Robustez, es resistente al patón estático, a diversos tipos de ruido [BOT01] e invariable al contraste.
- Fiabilidad, da una representación densa de la velocidad correcta en módulo y dirección en cada punto del campo visual.
- Versatilidad, ya que puede dar cuenta de movimiento de segundo orden, incluso detectar algunas ilusiones ópticas percibidas por los humanos [BOT01][JOH95].
- Autonomía, ya que no necesita más información *a priori* que la propia secuencia de imágenes.

- Directividad, puesto que separa las componentes diferenciales de las translacionales, sin recurrir a métodos iterativos.

Por lo tanto, la implementación de este sistema se hace sumamente atractiva debido a su robustez e imitación del procesamiento natural humano de forma eficiente, altamente paralela y autoadaptable en el mundo real.

El siguiente objetivo será describir una plataforma que optimice el algoritmo y le vaya confiriendo naturaleza de *hardware* (los recursos consumidos por éste son ingentes), mediante diversas transformaciones a efecto y estudios de la precisión necesaria, que lleven a un compromiso entre eficiencia y exactitud.

CAPÍTULO IV

Modelo de precisión controlada

4.1. Introducción a la plataforma de precisión controlada

En este capítulo se presenta una plataforma de precisión controlada que servirá como herramienta de transición entre los modelos *software* y *hardware* a partir de una versión del McGM con una capa de abstracción superior donde es posible definir el *bit cutting* o la precisión decidida en cada una de las diferentes etapas conceptuales del capítulo III.

Además, se explican los parámetros más relevantes y se comparan posteriormente los resultados de diferentes métricas de error, siendo el objetivo principal observar la degradación del modelo frente a la precisión usada y escoger una combinación de parámetros óptima para implementar en *hardware* configurable. La plataforma está realizada en C++ e incluye varias clases al efecto para recorte y control de precisión, siendo también posible elegir qué etapas se pueden simular con una precisión controlada y cuáles permanecen en doble precisión, para construir un sistema de co-diseño funcional.

Otro aspecto importante es el control de la densidad de puntos a calcular, discriminando entre puntos que se pierden debido al efecto de la precisión y puntos perdidos debido a filtros en el camino de datos de datos (como umbrales en etapas tempranas que ayudan a disminuir el error).

En cualquier caso, el objetivo consiste en encontrar una solución de calidad alta manteniendo al mismo tiempo la densidad de cálculo próxima a 100%.

4.2. Descripción de cada una de las etapas y los parámetros usados

4.2.1. Descripción de cada una de las etapas: variables y funciones

Se ha implementado una plataforma experimental para estudiar la profundidad de bit requerida en cada etapa de procesamiento del McGM.

Las variables consideradas son las siguientes:

- $\boldsymbol{I_n}$: profundidad de bit en la entrada de la etapa *n*-ésima.
- $\boldsymbol{O_n}$: profundidad de bit en la salida de la etapa *n*-ésima.
- $\boldsymbol{F_n}$: profundidad de bit correspondiente al filtro.
- $\boldsymbol{M_n}$: tamaño de la ventana espacial y/o temporal (número de elementos) de la máscara para convolución.

El número de bits requeridos para almacenar el elemento *x* viene según la expresión (4.1):

$$f_b(x) = \lfloor \log_2(x) + 1 \rfloor \tag{4.1}$$

En todas las etapas se restringe la profundidad de bit de 2 maneras:

a. se comprime la información linealmente;
b. se comprime la información saturando en un número fijo de bits.

En general $\boldsymbol{S_n}$ es el saturador para la etapa *n*, lo que equivale a una cuantización no uniforme. La saturación se lleva a cabo según la expresión (4.2)

$$\text{Si } x_n > S_n \text{ entonces } \underline{x_n} = S_n \tag{4.2}$$

4.2.2. Etapas funcionales de la plataforma

A lo largo de todo este proceso de cálculo, se han concebido 8 etapas funcionales que se describen a continuación.

I. Obtención de las derivadas temporales. (I_1, F_1, O_1)

Se denomina M_1 al tamaño de la ventana temporal (el número de imágenes consideradas en la convolución temporal) necesitando 3 derivadas temporales en esta etapa.

Operaciones a efectuar: convolución a través del eje temporal (*v*. 3.2.2.1). A partir del número de bits necesarios para evitar la pérdida de información, por ejemplo I_1=8, F_1=4, M_1=23, el número de bits requeridos para la salida en esta etapa es O_{1_nr}=8+4+f_b(23)=17, denotando O_{1_nr} la salida no restringida. Ésta se puede reducir a una salida restringida, O_{1_r}, calculando x_{1_r} a partir de cada valor x_{1_nr}.; posteriormente se aplica el mecanismo de saturación S_1, mostrado en (4.2).

$$x_{1\_r} = \frac{2^{O_{1\_r}}}{2^{O_{1\_nr}}} x_{1\_nr} \tag{4.3}$$

II. Convolución de las derivadas temporales con filtros espaciales (I_{2_I}, F_{2_I}, O_{2_I}, I_{2_II}, F_{2_II}, O_{2_II})

En esta etapa se obtiene una convolución espacial para cada una de las 3 convoluciones temporales previas. Se ha demostrado que la función Gaussiana y sus derivadas forman un conjunto separable, por lo que se aplicará la convolución de esta manera, ahorrando en operaciones y latencia (*v*. 3.2.2.2).

Operaciones a realizar: convolución espacial por columnas, siendo el tamaño de la máscara F_{2y} y su longitud $M_{2y,}$ correspondiendo la entrada a la salida anterior I_2=S_1; por tanto, se tiene que la profundidad de bit sin pérdida de información es $O_{2\_nr\_I}$=I_2+F_{2y}+f_b(M_{2y}).

$$x_{2\_r\_I} = \frac{2^{O_{2\_r\_I}}}{2^{O_{2\_nr\_I}}} x_{2\_nr\_I} \tag{4.4}$$

Una vez finalizado este proceso, se realiza la convolución por filas siendo el tamaño de la máscara de convolución total M_2= M_{2x}×M_{2y}. Para esta última convolución, su tamaño de la máscara es $F_{2x,}$ su longitud M_{2x} y su entrada I_2=$x_{2\_nr\_I}$. De acuerdo con estos datos, la profundidad de bit sin pérdida de información es $O_{2\_nr\_II}$= I_2+F_{2y} f_b(M_{2x}). Después de este procedimiento se usará el mecanismo de saturación, S_2.

$$x_{2\_r\_\mathrm{II}} = \frac{2^{O_{2\_r\_\mathrm{II}}}}{2^{O_{2\_u\_\mathrm{II}}}} x_{2\_nr\_\mathrm{II}} \tag{4.5}$$

III. Orientación de los filtros espacio-temporales "*Steering*" (I_3, F_3, O_3)

En esta etapa se orientan cada uno de los filtros calculados. Las componentes para diferentes ángulos se calculan como sumas ponderadas de las salidas piramidales anteriores. (*v.* 3.2.2.3). Si se codifican los pesos con W_3 bits, el número de bits para esta etapa será $O_{3\_nr}=I_3+W_3+3$ (donde "3" da cuenta del máximo número de adiciones para conservar).

Se comprime otra vez linealmente siguiendo la expresión (4.6) y se procede al mecanismo de saturación según S_3.

$$x_{3\_r} = \frac{2^{O_{3\_r}}}{2^{O_{3\_nr}}} x_{3\_nr} \tag{4.6}$$

IV. Cálculo del desarrollo de Taylor e integración en el volumen espacio-temporal. (I_4, F_4, O_4)

En esta etapa, se construye un desarrollo de Taylor truncado y cada una de sus derivadas, todas estas operaciones requieren sumas ponderadas de los términos y los pesos.

Operaciones a realizar: Cálculo de 6 funciones ($X \cdot X$, $X \cdot Y$, $Y \cdot Y$, $X \cdot T$, $T \cdot T$,$Y \cdot T$) y suma de 20 contribuciones de sumas pesadas (*v.* 3.2.4). En este caso $O_{4\_nr}=2 \cdot I_4+2 \cdot W_4+5$ (donde "5" viene de $f_b(20)$) siendo necesario almacenar 6 estructuras con esta profundidad de bit. Posteriormente a este proceso de compresión (4.7) se usa S_4 para realizar la saturación.

$$x_{4\_r} = \frac{2^{O_{4\_r}}}{2^{O_{4\_nr}}} x_{4\_nr} \tag{4.7}$$

V. Cálculo de las primitivas generales del flujo óptico

En esta etapa se calculan unas funciones a partir de las cuales se obtendrán el módulo y la fase del flujo. Esta es la última etapa conceptual común para módulo y fase conjuntamente. Operaciones : La organización de esta etapa se basa en 6 primitivas básicas.(4.8 a-f):

$$f_{p\,0} = \frac{x_\theta \cdot t_\theta}{t_\theta \cdot t_\theta}$$

$$f_{p1} = \left(\frac{x_\theta \cdot t_\theta}{x_\theta \cdot x_\theta} \right) \Big/ f_{p5} \tag{4.8b}$$

$$f_{p2} = \frac{y_\theta \cdot t_\theta}{t_\theta \cdot t_\theta} \tag{4.8c}$$

$$f_{p3} = \left(\frac{y_\theta \cdot t_\theta}{y_\theta \cdot y_\theta} \right) \Big/ f_{p4} \tag{4.8d}$$

$$f_{p4} = 1 + \left(\frac{y_\theta \cdot x_\theta}{y_\theta \cdot y_\theta} \right)^2 \tag{4.8e}$$

$$f_{p5} = 1 + \left(\frac{y_\theta \cdot x_\theta}{x_\theta \cdot x_\theta} \right)^2 \tag{4.8f}$$

TABLA 4.I Profundidad de bit necesaria para cada primitiva

Primitiva	*Profundidad bit*
f_{p0}	S_4
f_{p1}	S_4
f_{p2}	S_4
f_{p3}	S_4
f_{p4}	$1+2 \cdot S_4$
f_{p5}	$1+2 \cdot S_4$

La profundidad de bit en esta etapa es diferente para cada primitiva, representándose en la tabla 4.I sus valores. Independientemente a este efecto, el proceso de compresión lineal se realiza según (4.9) y el de saturación se efectúa con S_5.

$$x_{5\_r} = \frac{2^{O_{5\_r}}}{2^{(fpi)}} x_{5\_nr} \tag{4.9}$$

VI.I. Primitivas del módulo de la velocidad (I). B_{matrix} y T_{matrix}

En esta etapa aparece una ramificación del tronco común de operaciones con objeto de calcular las primitivas del módulo de la velocidad B_{matrix} y T_{matrix}.

Operaciones: Componentes del denominador y numerador del determinante (*v*. 3.2.5.6), necesarios para calcular el módulo, como muestran las ecuaciones (4.10a-4.10d) y (4.11a-4.11d). Para estas operaciones $O_{6T\_nr}= f_b(A/2)+I_6+ f_b(\theta)$ y $O_{6B\_nr}= f_b(A/2)+ 2I_6$, siendo A el número de ángulos totales y θ cada ángulo a cuantizar . Las profundidades de bit son O_{6B} y O_{6T}.

$$T_{matrix}[0][0] = \sum_{0}^{a/2-1} f_{p1}\cos(\theta) \;;\; T_{matrix}[0][1] = \sum_{0}^{a/2-1} f_{p1} sen(\theta) \qquad (4.10a)\ (4.10b)$$

$$T_{matrix}[1][0] = \sum_{0}^{a/2-1} f_{p3}\cos(\theta) \;;\; T_{matrix}[1][1] = \sum_{0}^{a/2-1} f_{p3} sen(\theta) \qquad (4.10c)\ (4.10d)$$

$$B_{matrix}[0][0] = \sum_{0}^{a/2-1} f_{p1} f_{p0} \;;\; B_{matrix}[0][1] = \sum_{0}^{a/2-1} f_{p1} f_{p2} \qquad (4.11a)\ (4.11b)$$

$$B_{matrix}[1][0] = \sum_{0}^{a/2-1} f_{p3} f_{p0} \;;\; B_{matrix}[1][1] = \sum_{0}^{a/2-1} f_{p3} f_{p2} \qquad (4.11c)\ (4.11d)$$

La profundidad de bit restringida puede ser calculada de acuerdo a las ecuaciones (4.12a) y (4.12b), en esta etapa y las que hacen referencia al módulo no se aplicará proceso de saturación.

$$x_{6B\_r} = \frac{2^{O_{6B\_r}}}{2^{(O_{6B\_nr})}} x_{6B\_nr} \qquad (4.12a)$$

$$x_{6T\_r} = \frac{2^{O_{6T\_r}}}{2^{(O_{6T\_nr})}} x_{6T\_nr} \qquad (4.12b)$$

VI.II. Primitivas de la fase de la velocidad (I)

En la presente etapa se calculan las primitivas necesarias que producirán la fase de la velocidad. Las operaciones necesarias son las multiplicaciones de funciones trigonométricas con las primitivas de la etapa *V* (extractores de los coeficientes de Fourier, *v*. 3.2.5.6). La saturación en el caso de la fase, se vuelve a usar de la manera usual, en este caso se denomina S_{6M}.

$$M = \left[\sum_{0}^{a/2-1} (f_{p0} + f_{p1})\cos(\theta) \;,\; \sum_{0}^{a/2-1} (f_{p0} + f_{p1}) sen(\theta) \right] \qquad (4.13a)$$

$$Mo = \left[\sum_{0}^{a/2-1} (f_{p2} + f_{p3})\cos(\theta) \;,\; \sum_{0}^{a/2-1} (f_{p2} + f_{p3}) sen(\theta) \right] \qquad (4.13b)$$

Los valores restringidos de profundidad de bit pueden ser calculados mediante la expresión (4.14), donde O_f es la función que modela la exactitud de las funciones trigonométricas *sen y cos.*

$$x_{6M\_r} = \frac{2^{O_{6\_r}}}{2^{(O_5+O_f+1)}} x_{6M\_nr} \tag{4.14}$$

VII.I. Primitivas del módulo de la velocidad (II). det_B_m y det_T_m

Se procede a continuar el cálculo de las primitivas desglosadas en módulo, para ello se efectúan las operaciones de cálculo de sumas y productos que conforman los determinantes en el denominador y numerador del cociente final. Adicionalmente, hay que notar que la profundidad de bit es O_{B7} y O_{T7}.

$$\det\_t_m = \begin{bmatrix} T_{matrix}[0][0] & T_{matrix}[0][1] \\ T_{matrix}[1][0] & T_{matrix}[1][1] \end{bmatrix} \tag{4.15a}$$

$$\det\_b_m = \begin{bmatrix} B_{matrix}[0][0] & B_{matrix}[0][1] \\ B_{matrix}[1][0] & B_{matrix}[1][1] \end{bmatrix} \tag{4.15b}$$

La profundidad de bit restringida puede ser calculada mediante las ecuaciones (4.16a) y (4.16b), definidas a continuación, en las que $O_{7T\_nr}=2I_7+1-f_b(P(O_{7T\_nr}))$ y $O_{7B\_nr}=2I_7+1-f_b(P(O_{7B\_nr}))$ siendo $P(i)$ una función que cuantifica la probabilidad de que ocurra el suceso en el caso más desfavorable de forma que haya que tomar el valor extremo de número de bits, $f_b(Prob(O_{7T\_nr}))=2$, (*idem* para $f_b(Prob(O_{7B\_nr}))$).

$$x_{7B\_r} = \frac{2^{O_{7B\_r}}}{2^{(O_{7B\_nr})}} x_{7B\_nr} \tag{4.16a}$$

$$x_{7T\_r} = \frac{2^{O_{7T\_r}}}{2^{(O_{7T\_nr})}} x_{7T\_nr} \tag{4.16b}$$

VII.II. Primitivas de la fase de la velocidad (II)

En este apartado se calcula la fase mediante operaciones de sumas y cocientes, donde las expresiones (4.17) y (4.18) definen la fase, así como el número de bits restringido:

$$\left[\frac{M(1) + M_o(0)}{M(0) - M_o(1)} \right] \tag{4.17}$$

$$x_{7M\_r} = \frac{2^{O_{7MB\_r}}}{2^{(O_{7MB\_nr})}} x_{7B\_nr} \tag{4.18}$$

VIII.I. Módulo de la velocidad (III). Cálculo final

Por último se computa el módulo de la velocidad, donde el tipo de operación a efectuar viene determinada por la expresión (4.19):

$$vel^2 = \left(\frac{\det\_t_m}{\det\_b_m} \right) \tag{4.19}$$

además, su valor de profundidad de bit restringida en esta etapa es:

$$x_{8\_r} = \frac{2^{O_{8\_r}}}{2^{(O_{8\_nr})}} x_{8\_nr} \tag{4.20}$$

VIII.II Fase de la velocidad (III) Cálculo Final

Es necesario calcular la función trigonométrica siguiente:

$$angle = \arctan\left[\frac{M(1) + Mo(0)}{M(0) - Mo(1)} \right] \tag{4.21}$$

la exactitud de esta expresión depende del número de bits de precisión de la LUT, que se usará para calcular la función *arctan* .

4.2.3. Resumen de parámetros más importantes usados en la plataforma experimental

A continuación se detallan los parámetros más relevantes de la plataforma *software* desarrollada para la evaluación de la precisión del modelo:

- `T_CONV, CONV` (entero): longitud de los filtros temporales y espaciales ecualizados apropiadamente con media cero; los valores por defecto son 23 y 21, respectivamente;
- `ALFA` (entero): respuesta al pico del filtro; el valor por defecto es 10;
- `TAU` (flotante): anchura del filtro. `ALFA` y `TAU` tienen efectos correlativos siendo el valor por defecto de `TAU` es 0.275;

- `CPP` (flotante): escala de los filtros espaciales. Un variación de la escala tiene un efecto pobre sobre patrones simples, como una retícula sinusoidal, ya que un suavizado no altera la orientación espacio-temporal del patrón. Su valor por defecto es 1.5**;**
- `ZONA_T, ZONA_Y, ZONA_X` (entero): definen la zona de integración, (normalmente simétrica, la asimetría requiere una arquitectura más elaborada y más tiempo de cálculo). Estos parámetro son críticos y su cambio puede producir efectos no deseado. El valor por defecto es 11;
- `NUM_FILTROS_TEMPORALES` (entero): parámetro estructural. Con 2 filtros (derivada de orden 0,1) la computación de movimiento es efectiva, pero añadiendo un tercer filtro se reduce la influencia del patrón estático. El valor por defecto es 2**;**
- `ORDEN_T, ORDEN_Y, ORDEN_X` (entero): estos son parámetros meramente estructurales, ya que a mayor número de filtros la aproximación al desarrollo de Taylor será mejor y, en consecuencia, se obtendrá un ajuste más fino para distancias cortas. Los valores por defecto son 2, 2 y 5, respectivamente;
- `ANGULOS` (entero): número de orientaciones calculadas explícitamente, afecta a la exactitud en el dominio del espacio. Este parámetro tiene poca importancia en secuencias muy regulares, sin embargo su importancia para patrones más complejos, es importante. Su valor por defecto es 24**;**
- `LIMITE_NUM, LIMITE_DEN` (doble): especifica un límite en el cálculo del numerador y denominador. (Si el valor es más pequeño que `LIMITE_NUM, LIMITE_DEN` el valor es cero explica esto bien). Los valores por defecto son 0.000001 y 0.0001, respectivamente**;**
- `VALOR_MIN_METRICA` (doble): es un valor umbral para métricas de error a considerar y estadísticas finales. El valor por defecto es 0.000000001;
- `N1, N2, …, N9` (entero): definen la profundidad de salida de cada etapa conceptual. `Saturador_ni` es el valor máximo que toma cada etapa si es necesario usarlo;
- `NUMEROBITS` (entero): proporciona el número de bits usado en la etapa inicial. Este parámetro simula el uso de cámaras con diferente exactitud debido al tipo de sensor óptico empleado, siendo su valor por defecto 8 bits;
- `NUMBERODEBITS_TEMP` (entero): profundidad de bit del filtro temporal;
- `NUMEROBITS_ESP` (entero); profundidad de bit del filtro espacial;

- NUMEROBITS_STEER (entero): profundidad de bit de los pesos en el proceso de *Steering*;
- NUMEROFBITS_TAYLOR (entero): profundidad de bit de los pesos en el desarrollo de Taylor;
- USAR_UNIDAD_PESOS_TAYLOR (booleana): permite ajustar cada peso del desarrollo de Taylor a 1. En este caso, se pondera de la misma manera la importancia de cada derivada. Por defecto está desactivada;
- HACER_PREBLUR (booleana): permite aplicar una etapa de pre-filtrado aplicando un filtro sencillo paso-baja ajustable que elimina el ruido espurio y suaviza los bordes, estando activada por defecto;
- SUBMUESTREADOR (entero): modula la tasa de muestreo espacial. Combinando este parámetro con el anterior, es posible atenuar el movimiento para medir mejor la velocidad. Su valor por defecto es 1**;**
- UMBRAL_MASCARA1, UMBRAL_MASCARA2, UMBRAL_MASCARA3 (entero): umbrales que operan sobre la primera y segunda derivadas temporales, así como el valor absoluto de la suma de éstas, mejorando el resultado final si se discriminan los datos que no han tenido un cambio significativo en el tiempo. El valor por defecto es el triplete (0,0,0), indicando que no se filtra ningún valor correspondiente a las derivadas temporales. Al incrementar estos coeficientes, se seleccionan los puntos que pasan a través del camino de datos, aunque como contrapartida hay que reseñar que disminuye la densidad de puntos a considerar. Teniendo como objetivo no sólo obtener los mejores resultados sino además llegar a una densidad de cálculo próxima a 100%.
- TAMAÑOX, TAMAÑOY (entero): tamaño de cada fotograma de la secuencia sin tener en cuenta el parámetro SUBMUESTREADOR;
- FOTOGRAMAS (entero): número de fotogramas de la secuencia para estimar el flujo óptico. El valor por defecto es 3;

El modelo cuenta con otros parámetros, que son mucho más específicos y relativos al algoritmo y al entorno de desarrollo usado, *Visual C++ 6.0*, con lo que resultan irrelevantes para mostrarlos en este trabajo. Se puede consultar más detalles en [BOT03], donde aparece un análisis pormenorizado de cada parámetro, así como las variables y salidas intermedias de

test que confirman que toda la ruta de datos funciona de forma satisfactoria; también es posible visualizar ejemplos gráficos de salidas intermedias realizadas con secuencias reales de adelantamientos dinámicos de coches, mediante una cámara situada en el coche adelantado.

4.3. Estímulos usados, métricas y promedios

4.3.1. Estímulos usados

A continuación se explican los estímulos usados como banco de pruebas. Todos han sido diseñados mediante *Matlab* [MAT06] como una superposición de ondas planas, cada una con una orientación espacial propia, como se muestran en la ecuación (4.22), la tabla 4.II y en la figura 4.1. Estos estímulos tienen un número suficientemente amplio de diferentes frecuencias temporales y espaciales para ser un buen patrón de test.

$$\left.\begin{aligned}\psi_2(x,t) &= A_2 \sin 2\pi(t/\tau_2 - x/\lambda_2)\\ \psi_1(x,t) &= A_1 \sin 2\pi(t/\tau_1 - x/\lambda_1)\end{aligned}\right| \rightarrow \psi = \psi_2(x,t)\big|_{\theta 2} + \psi_1(x,t)\big|_{\theta 1} \qquad (4.22)$$

4.3.2. Métricas y promedios

Se realizan medidas de error según una métrica sencilla consistente en los errores absolutos y relativos de módulos y fases, tal y como se indican en las ecuaciones (4.23a) y (4.23b) siguientes:

$$\psi_{FAS} = \left|\arctan(\vec{v}_{cy}/\vec{v}_{cx}) - \arctan(\vec{v}_{ey}/\vec{v}_{ex})\right| \qquad (4.23a)$$

$$\psi_{MOD} = \left\|\vec{v}_c - \vec{v}_e\right\| \qquad (4.23b)$$

TABLA 4.II Longitud de onda, velocidad y fase de los 3 estímulos usados en la plataforma de precisión variable.

	λ_1 (puntos)	λ_2 (puntos)	v_1 (puntos/foto)	v_2 (puntos/foto)	$\theta 1$	$\theta 2$
Est. I	6	6	1.63	1.02	54º	-26º
Est. II	60	15	0.66	0.75	210º	130º
Est. III	8	8	1	1	-45º	0

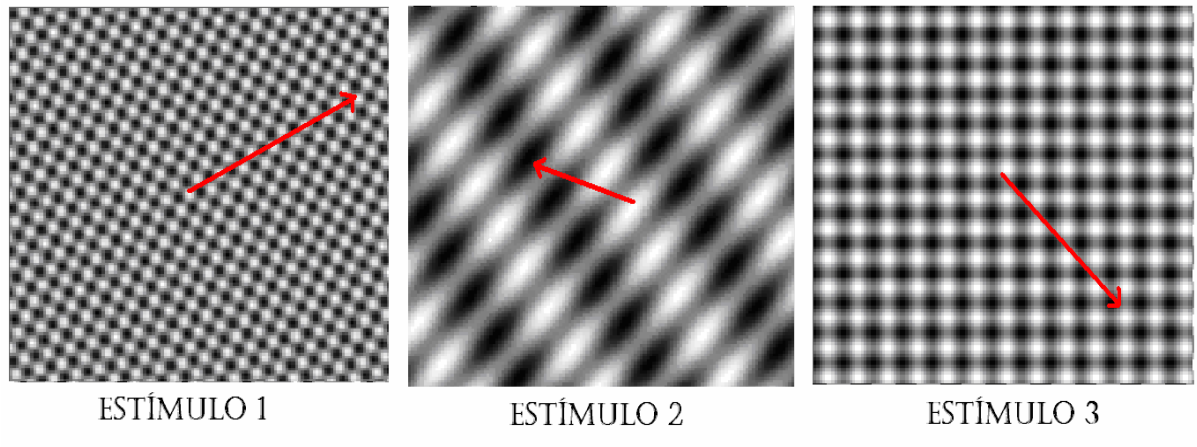


Figura 4.1. Ilustración de los 3 estímulos para pruebas (rotación antihoraria): patrón moviéndose a 2.05 puntos/fotogramas y 28º de orientación (*izquierda*), patrón moviéndose a 1.0 puntos/fotogramas y 162º de orientación (*centro*) y patrón moviéndose a 1.4142 puntos/fotogramas y -45º de orientación (*derecha*).

Barron [BAR94] propone el uso del siguiente vector tridimensional:

$$\vec{v} = \frac{(v_x, v_y, 1)}{\sqrt{v_x^2 + v_y^2 + 1}} \tag{4.24}$$

de forma que se agrupen la velocidad y dirección en una sola medida. El error angular entre la velocidad correcta $\boldsymbol{v_c}$ y la estimada $\boldsymbol{v_e}$ viene dado por:

$$\psi_E = \arccos(\vec{v}_c \cdot \vec{v}_e) \tag{4.25}$$

Esta medida es ampliamente aceptada en el seno de la comunidad científica, relacionando las medidas angulares en el diagrama *x-t* o *y-t*, como se muestra en la figura 4.2, y se expresa según la ecuación siguiente:

$$n = \cos(90 - \alpha_n) \Big/ \sqrt{\left(1 - \cos^2(90 - \alpha_n)\right)} n \in \{x, y\} \tag{4.26}$$

Existen, no obstante, autores importantes que no consideran concluyente esta aproximación [GAL98][ARR03]. y en su lugar, proponen otras 2 medidas alternativas, que quedan reflejadas en las expresiones (4.27, 4.28), donde se modela por una parte la magnitud de la diferencia entre la medida correcta y estimada según indica la ecuación (4.27):

$$\psi_M = \left\| \vec{v_c} - \vec{v_e} \right\| \tag{4.27}$$

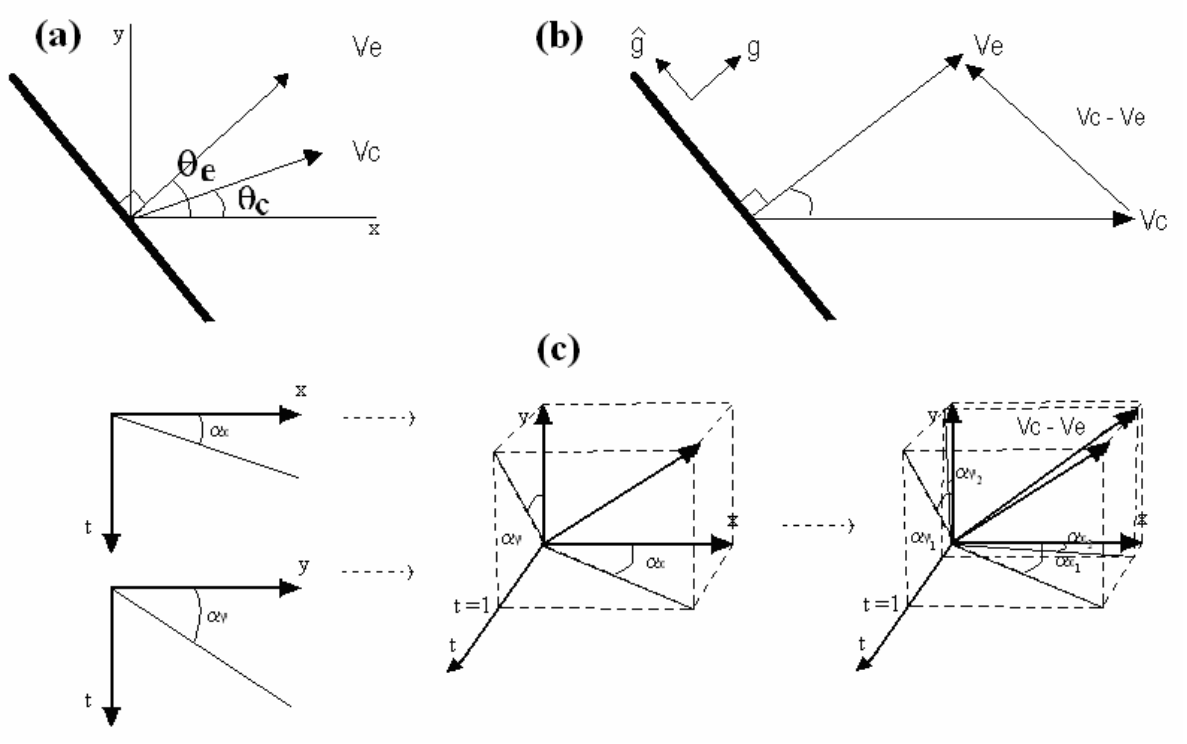


Figura 4.2. Esquema de los diferentes criterios de medida empleados:
(a) Medida basada en diferencias de módulo y fase de vectores teóricos y experimentales .
(b) Medida basada en los criterios de Galvin usual y perpendicular.
(c) Medida basada en los criterios de Barron.

y por otra parte, una medida del error normal a la dirección del gradiente, siendo esta última medida un buen indicativo de la interacción con el problema de apertura [PAJ04].

$$\psi_N = \left\| (\vec{v_c} - \vec{v_e}) \cdot \hat{g}^{\perp} \right\| \tag{4.28}$$

En este capítulo se van a utilizar las expresiones (4.23a, 4.23b) y comprobar *a posteriori* su bondad con el conjunto de métricas consideradas. Este procedimiento se debe a la gran disparidad de estímulos de prueba y métricas usadas por los investigadores.

Con estas métricas de partida, se puede controlar el error en cada una de las etapas intermedias, con objeto de llegar a una solución final, donde se verificará la bondad de estos resultados mediante pruebas con otras métricas adicionales más complejas (expresiones 4.24, 4.25, 4.27, 4.28) para homologar y contrastar las conclusiones numéricas obtenidas respecto a otros trabajos del área de conocimiento.

Por último se va configurar esta plataforma experimental para aplicar como entrada cada uno de los tres estímulos mostrados en la figura 4.1, donde se repite todo el proceso para un fotograma superior y un fotograma inferior al de referencia, es decir con una iteración global de 3 fotogramas, obteniendo un punto cuyo valor será la media aritmética de cada una de los componentes individuales del macropunto de información (9 unidades).

4.4. Simplificaciones posibles y relación con la precisión de cálculo

4.4.1. Algoritmo simplificado

A continuación se escribe en pseudocódigo el protocolo a implementar:

Algoritmo 4.1. Algoritmo donde se muestra el proceso a efectuar siendo $E(0)$ y $E(1)$ los errores en fase y módulo, respectivamente, y $M(z)$ el tipo de métrica a usar.

```
Para etapa i, que contiene parámetros p1…pn
 {
    Para cada pj (j=1:n)
     {
          Para cada grupo pa (a=1:j)
          {
                Cuantización pa , conjunto { p1…pn }- pa en doble prec.
                {Elección de pa de forma que M(z)(i, pa )<E(z)(i)}
                Saturación mediante Si
                {Elección de Si forma que M(z)(i, pa ,Si)<E(z)(i)}
          }
     }
 }
```

4.4.2. Análisis de cada una de las etapas

4.4.2.1. Etapa I: Filtrado temporal triple

Es necesario estudiar exhaustivamente el comportamiento de esta etapa frente a varios factores, como el número de bits del filtro temporal, de la etapa I y del saturador. Se aplicarán entonces tres procesos encadenados, teniendo como operaciones tres convoluciones en el eje temporal (*v.* 3.2.2.1):

a) Cuantización del filtro temporal manteniendo la etapa I con valores a doble precisión.
b) Cuantización de la etapa I según la precisión escogida en (a).
c) Saturación de la etapa actual (este proceso equivale cuantizar de forma no uniforme a partir de una cota S_i) de acuerdo con la decisión en (b).

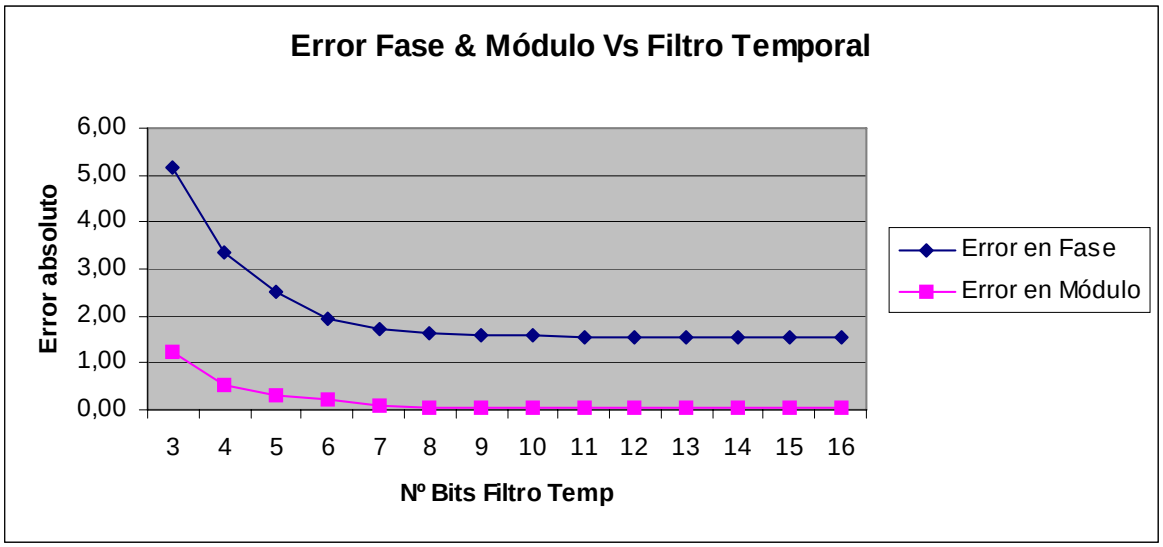


Figura 4.3. Error absoluto Etapa I vs precisión filtro FIR temporal. Proceso de cuantización.

A continuación se representan cada uno de los resultados referidos a estos procesos de cuantización y saturación. Según la figura 4.3, se puede observar que tanto $E(0)$ como $E(1)$ evidencian un comportamiento monótonamente decreciente a medida que el número de bits del filtro temporal se incrementa, mientras que el resto de parámetros la etapa continúa a doble precisión.

Se realiza una elección correspondiente a 6 bits de precisión y se muestran los resultados de cuantizar esta convolución mediante la aplicación de la salida restringida (O_{1_r}) en las escalas absolutas (figura 4.4) y relativas (figura 4.5).

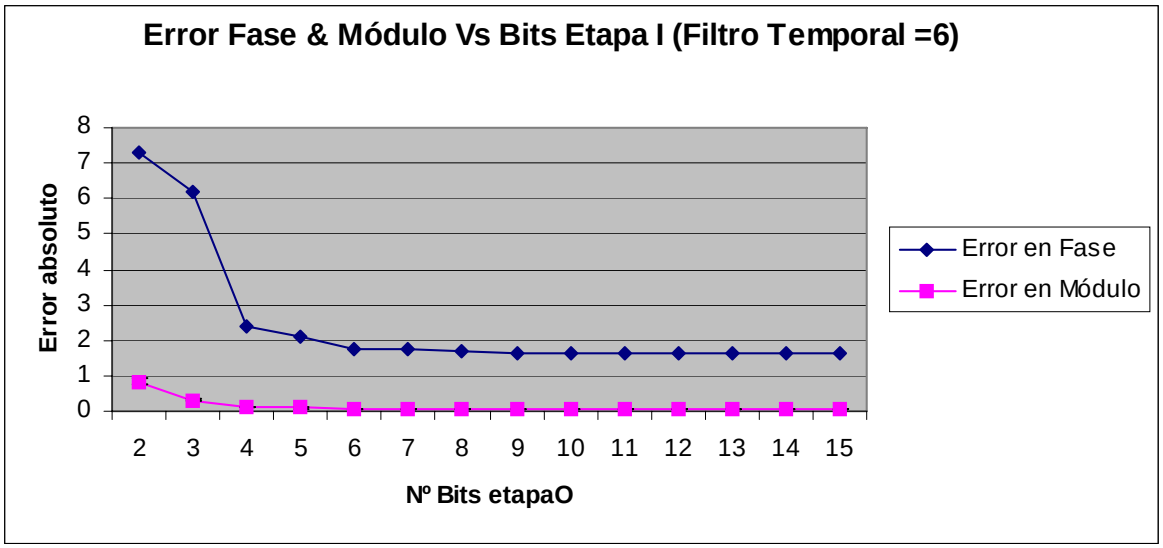


Figura 4.4. Error absoluto Etapa I vs precisión O_{1_r} (filtro FIR temporal con 6 bits). Proceso de cuantización.

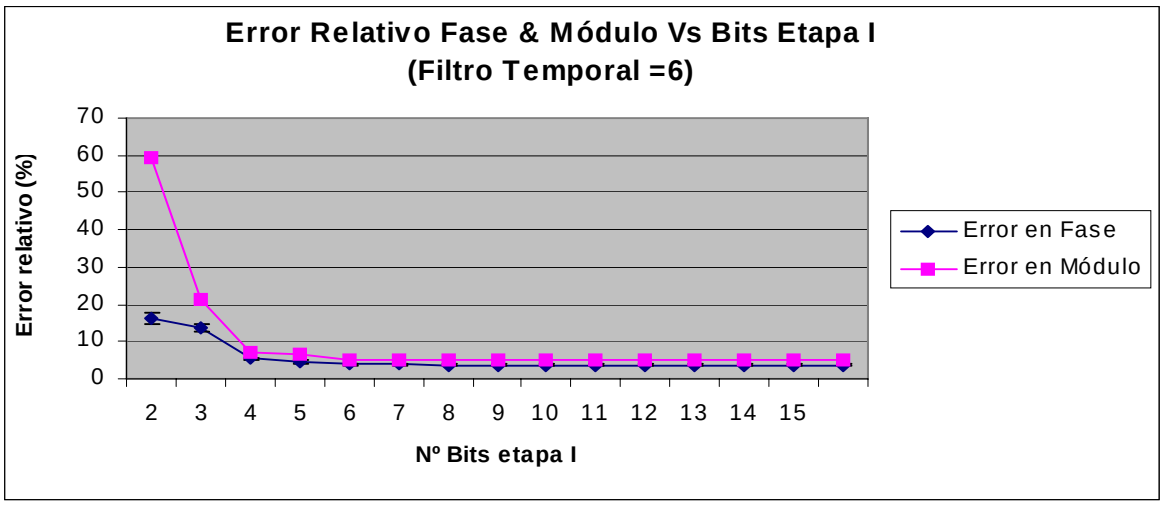


Figura 4.5. Error relativo Etapa I vs precisión O_{1_r} (filtro FIR temporal con 6 bits). Proceso de cuantización.

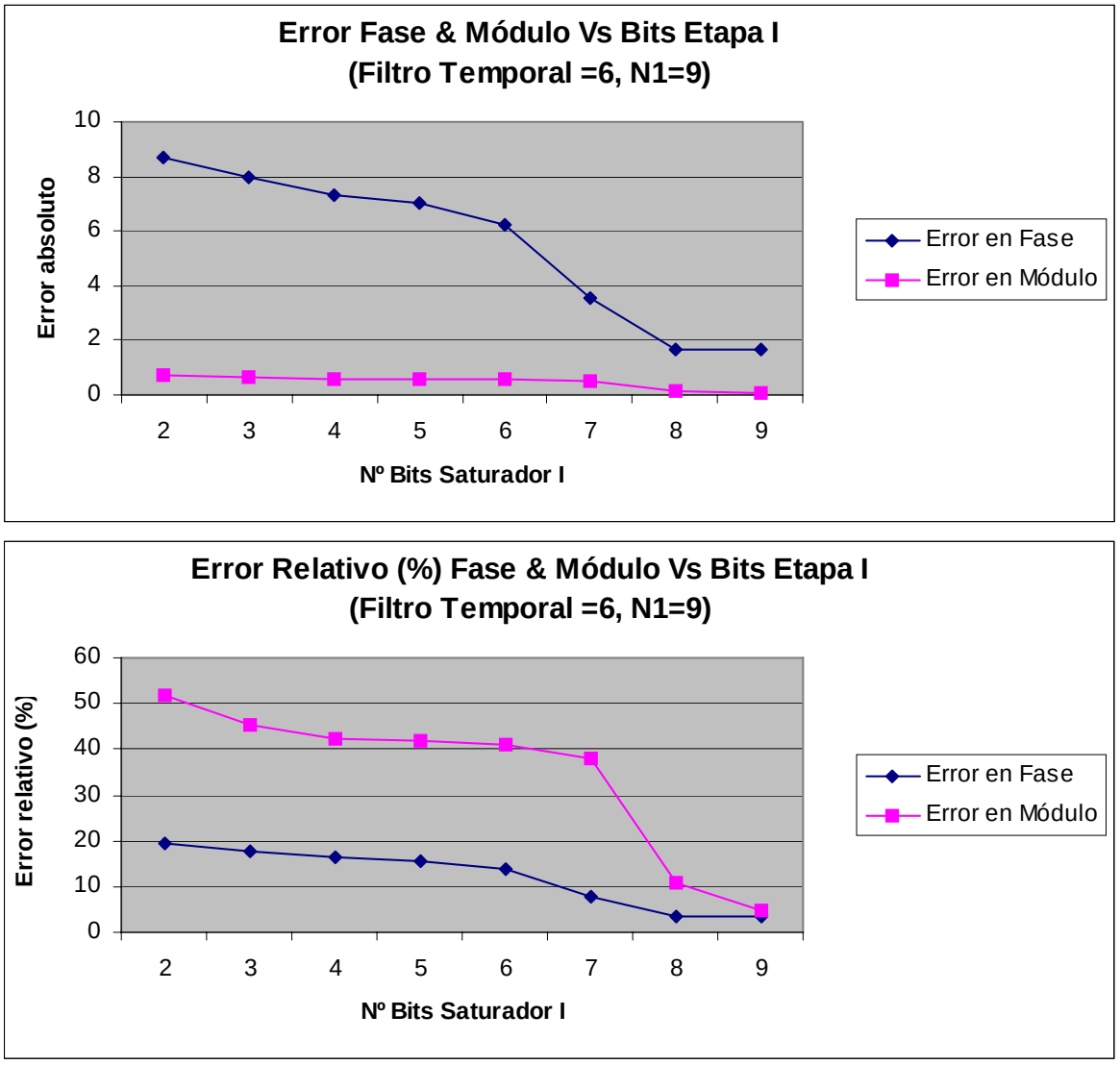


Figura 4.6. Proceso de saturación en la Etapa I y evolución del error absoluto (superior) y relativo (inferior) conforme aumenta el número de bits del saturador. Se eligen 9 bits para la Etapa I con objeto de no perder precisión todavía.

En este caso se observa igualmente un comportamiento decreciente, estabilizándose el sistema para unos valores 1.64/0.07 (3.64/4.95) correspondientes al error absoluto (relativo) para $E(0)/E(1)$. Si se opta por O_{1_r}=9 bits y por último, se aplica el proceso de saturación, se obtienen los resultados que indican las figuras 4.6 y 4.7, manteniendo en este caso el proceso de saturación con los mismos bits de partida.

4.4.2.2. Etapa II: Doble filtrado espacial separable

A continuación se va a estudiar el comportamiento de la etapa II frente al número de bits del filtro espacial y del saturador. Para ello se van a aplicar de forma encadenada 4 procesos (*v*. 3.2.2.2):

a) Cuantización del filtro manteniendo la etapa II con el resto de valores de doble precisión.
b) Cuantización de la etapa II_1 (primera parte de la convolución separable realizada de forma paralela por filas) según la precisión escogida en (a).
c) Cuantización de la etapa II_2 (segunda parte, consta de una convolución realizada por columnas) según la precisión escogida en (b).
d) Saturación de la etapa II de acuerdo con (c).

Se procede a ilustrar (figura 4.7) los resultados de estos procesos, apreciándose de forma coherente un decremento de $E(0)$ y $E(1)$ a medida que el número de bits se va incrementando. En el proceso convolutivo se aplican las salidas restringidas ($O_{2\_r\_I}$, $O_{2\_r\_II}$) con una elección correspondiente a 8 bits de precisión en las escalas absolutas y relativas, estabilizándose el sistema para unos valores de error 1.71/0.07 (3.90/4.95) correspondientes a error absoluto (relativo) para $E(0)/E(1)$.

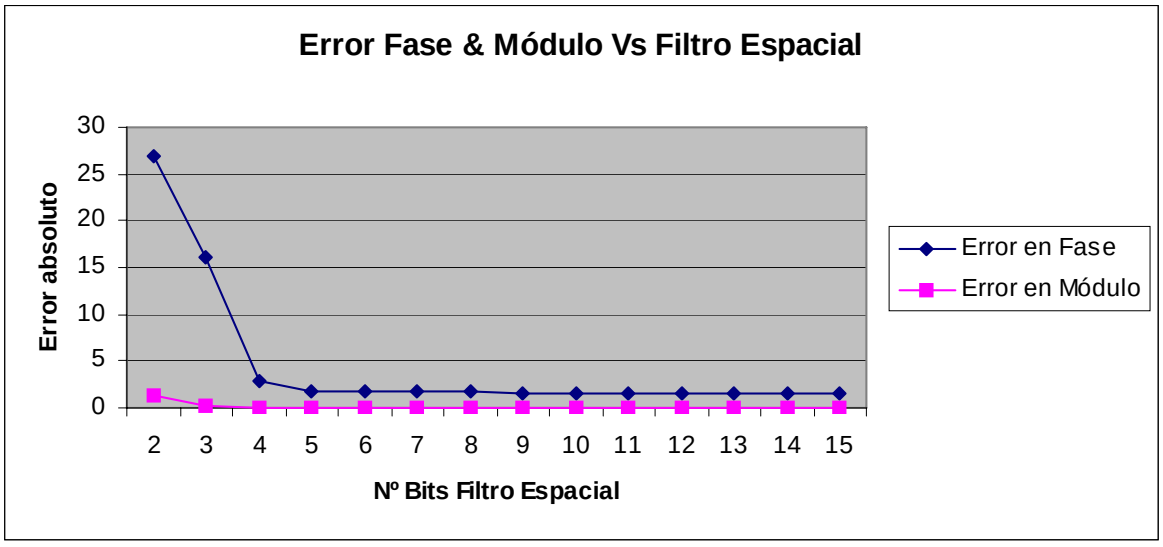


Figura 4.7. Error absoluto Etapa II vs precisión filtro FIR espacial. Proceso de cuantización.

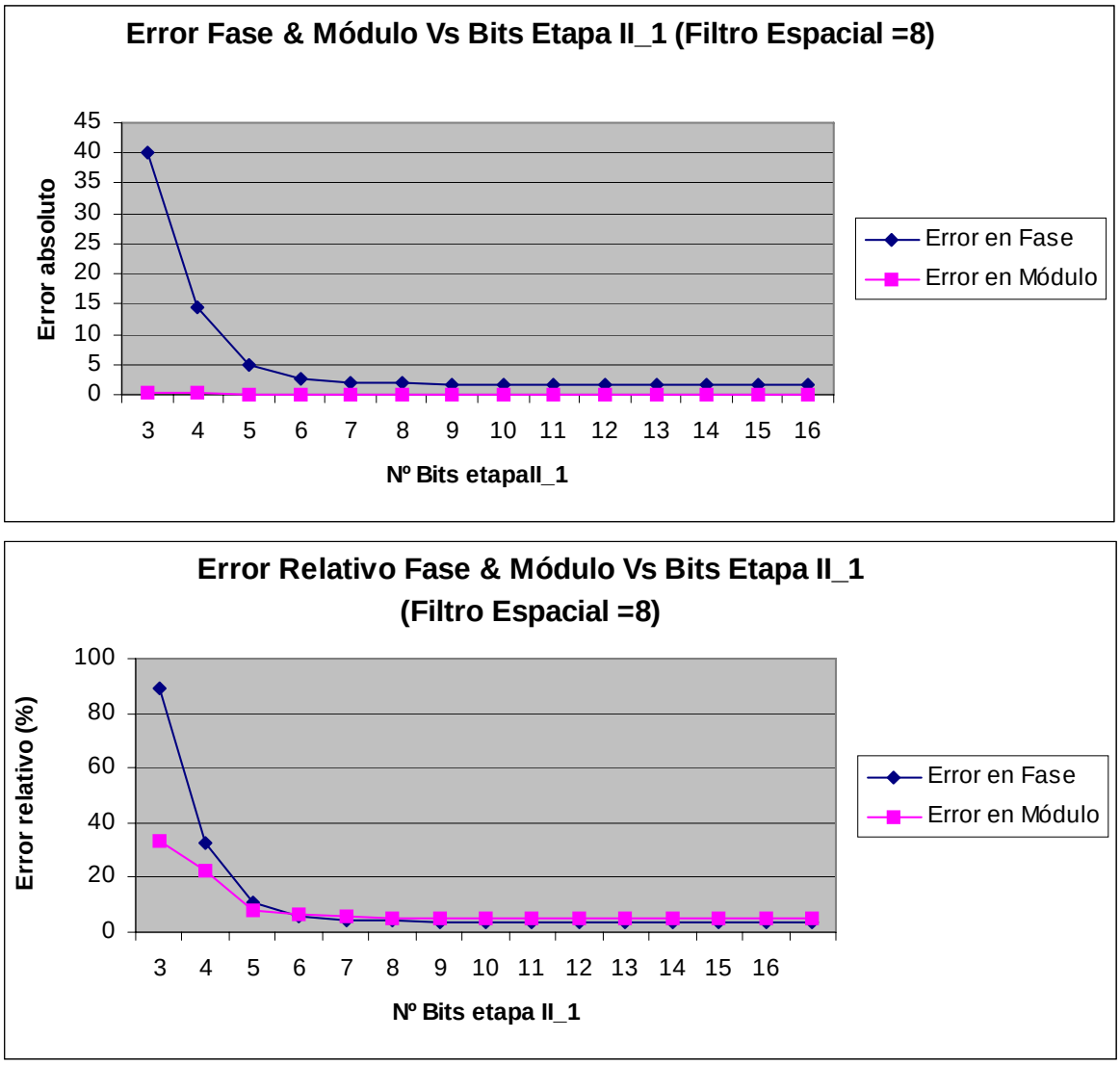
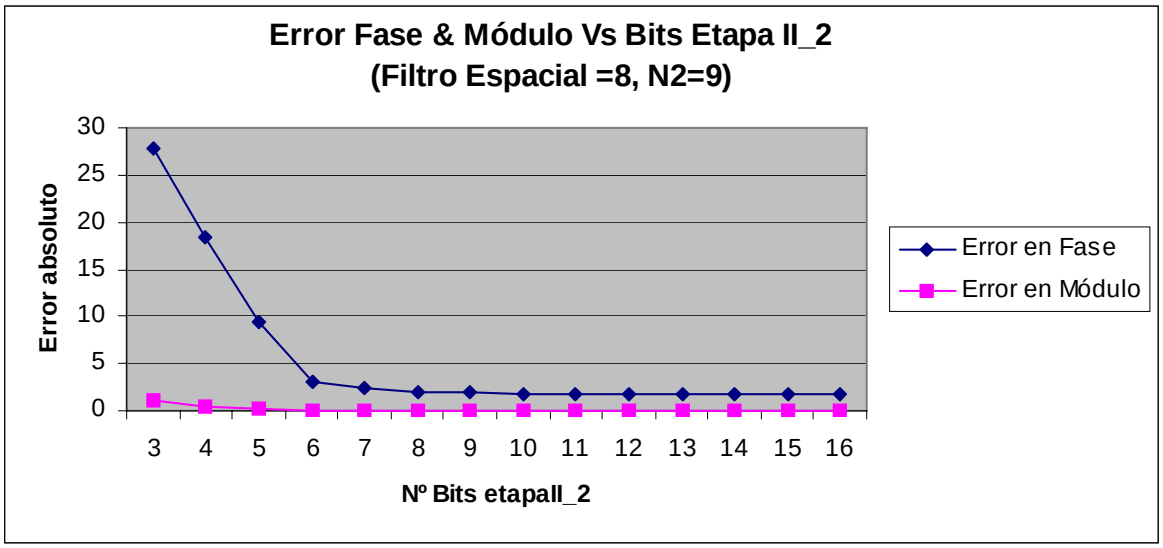


Figura 4.8. Error absoluto (superior) y relativo (inferior) de la Etapa II vs precisión $O_{2\_r\_I}$ (filtro FIR espacial con 8 bits). Proceso de cuantización.

A partir de estos parámetros, se realiza una segunda elección que se materializa con $O_{2\_r\_I}$=9 bits y 1.81/0.07 (4/4.95) de error absoluto (relativo) para $E(0)/E(1)$, con $O_{2\_r\_II}$=10 bits, en este caso, el resultado del error final está por debajo de 5%, manteniendo cotas de calidad satisfactorias.

En la figura 4.8 se representa la cuantización en el proceso primario de convolución. En la figura 4.9 se representa la cuantización respecto a la segunda parte de convolución separable; adicionalmente la figura 4.10 ilustra la evolución de la pérdida de densidad debida a puntos de cálculo, no alcanzándose un 100% de puntos calculados hasta los 6 bits. Por último, la aplicación del proceso de saturación queda reflejada en la figura 4.11, donde se opta por 8 bits, repercutiendo en un ahorro global de 2 bits adicionales.



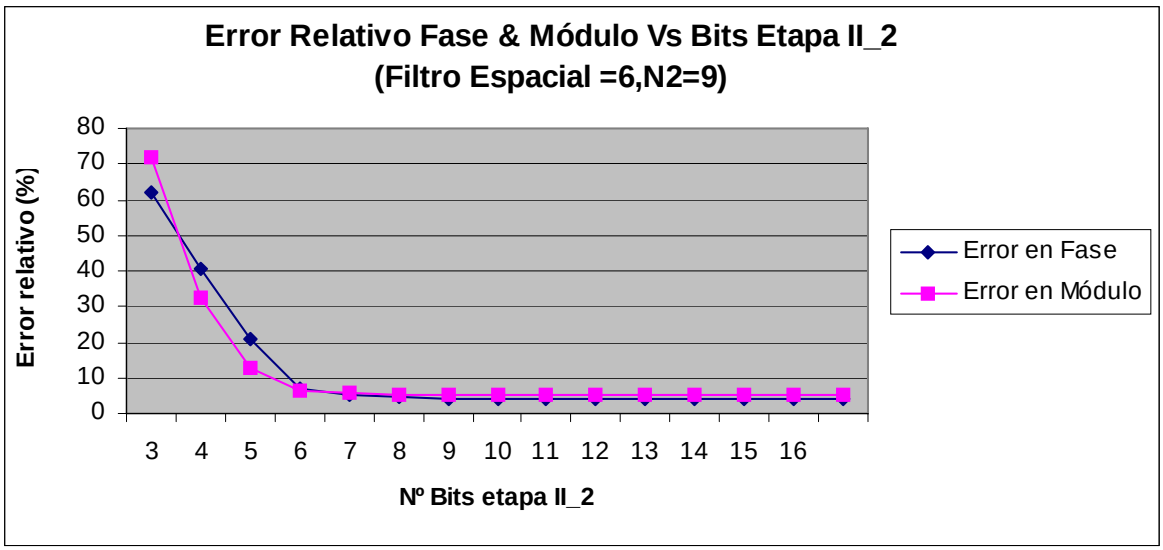


Figura 4.9. Error absoluto (superior) y relativo (inferior).
Etapa II vs precisión $O_{2\_r\_II}$ (filtro FIR espacial con 8 bits). Proceso de cuantización.

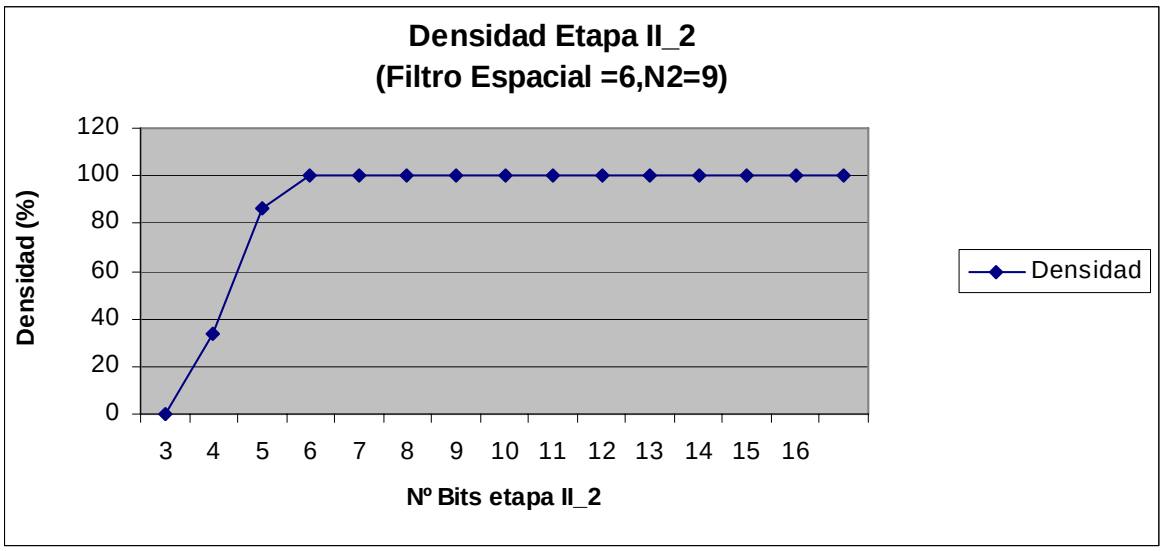


Figura 4.10. Densidad de puntos restantes debido a la precisión.

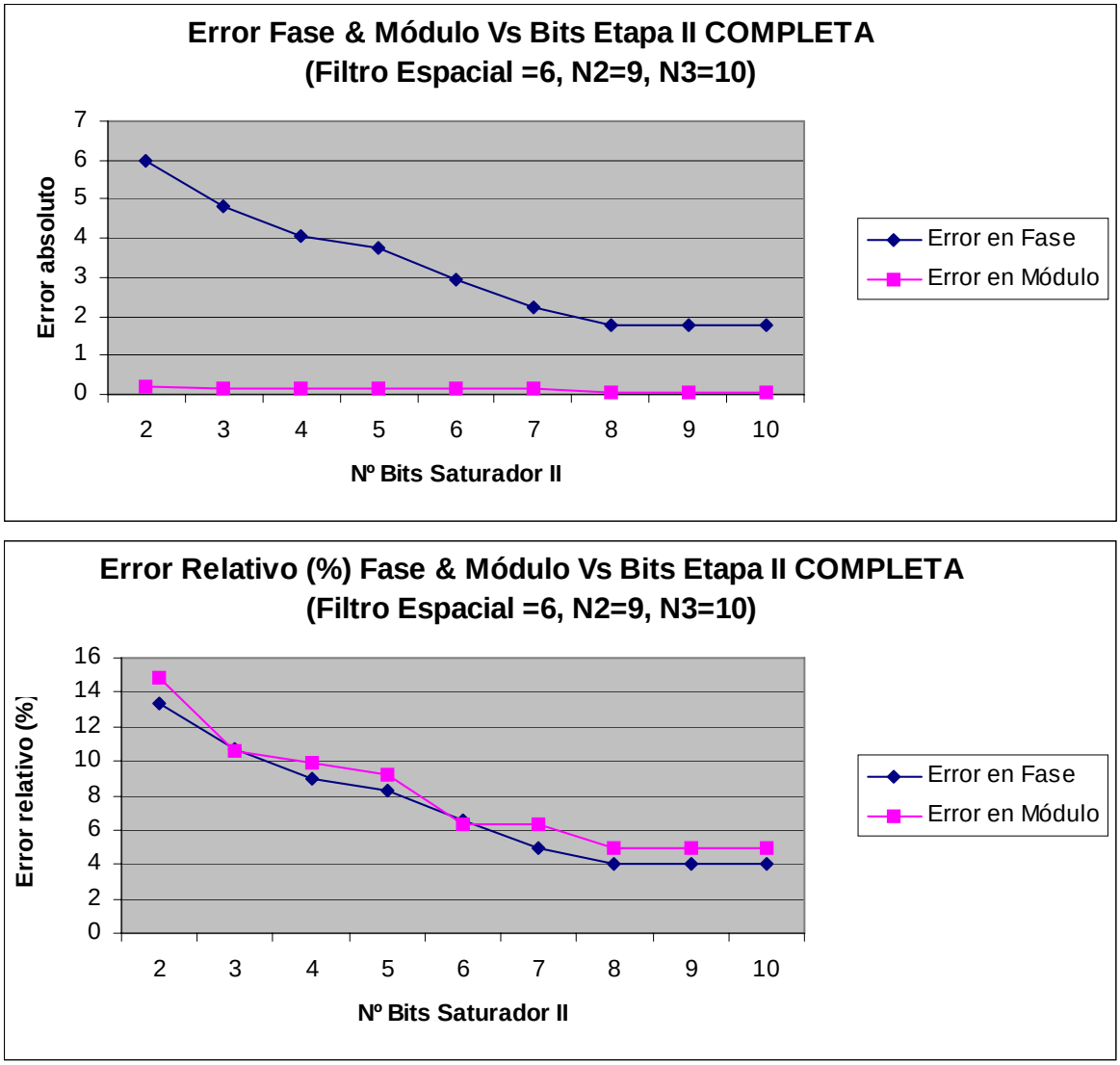


Figura 4.11. Proceso de saturación en la Etapa II . Evolución de los errores absolutos (superior) y relativos (inferior) conforme aumenta el número de bits del saturador. Elección de 8 bits para esta etapa (zona de estabilización) y ahorro de 2 bits globales.

4.4.2.3. Etapa III: Derivadas Orientadas. *Steering*

A continuación, se procede al estudio paramétrico del comportamiento de la etapa de orientación de las derivadas espacio-temporales frente al número de bits de los pesos, bits de la etapa III y número de bits del saturador. (*v*. 3.2.2.4.1-6), aplicando 3 procesos encadenados :

a) Cuantización de los valores de los pesos, manteniendo la etapa III en doble precisión.

b) Cuantización de etapa III según la precisión escogida en (a).

c) Saturación de la etapa III de acuerdo con la decisión en (b).

En la figura 4.12 se muestra la evolución de $E(0)$ y $E(1)$ a medida que el número de bits se incrementa, cabe resaltar en este caso una tendencia a mantener el error del valor absoluto del módulo prácticamente constante.

Si se escogen 6 bits de precisión para estos pesos, los resultados siguen una evolución debida a cuantizar la etapa previa aplicando la salida restrictiva O_{3_r} para las escalas absoluta y relativa como ilustran las figuras: 4.12 y 4.14. El sistema llega a ser estable para unos valores de error 1.95/0.09 (4.33/6.36) de error absoluto (relativo) para $E(0)/E(1)$ si la elección de O_{3_r} se efectúa con 10 bits. Con respecto a la densidad, se necesitan 6 bits para alcanzar el 100% según aprecia en la figura 4.13. Relativo al proceso de saturación, se alcanza un estado donde los errores ascienden a 2.23/0.09 (4.95/6.36) consiguiendo esta vez un ahorro neto de 4 bits, según se muestra en la figura 4.15.

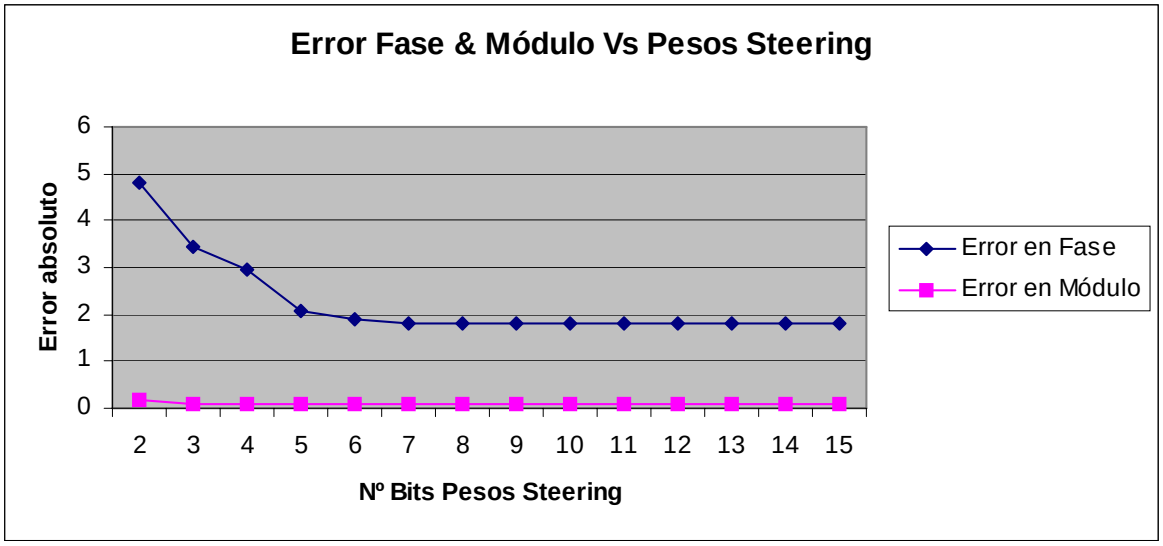


Figura 4.12. Pesos de la combinación lineal. Proceso de cuantización.

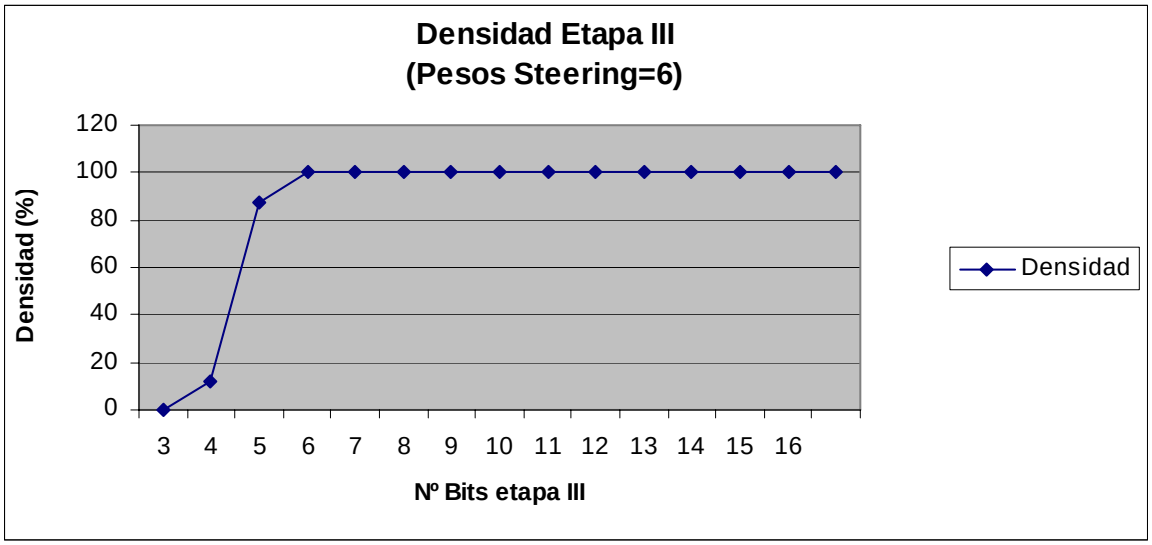


Figura 4.13. Estudio de la densidad en función de O_{3_r}.

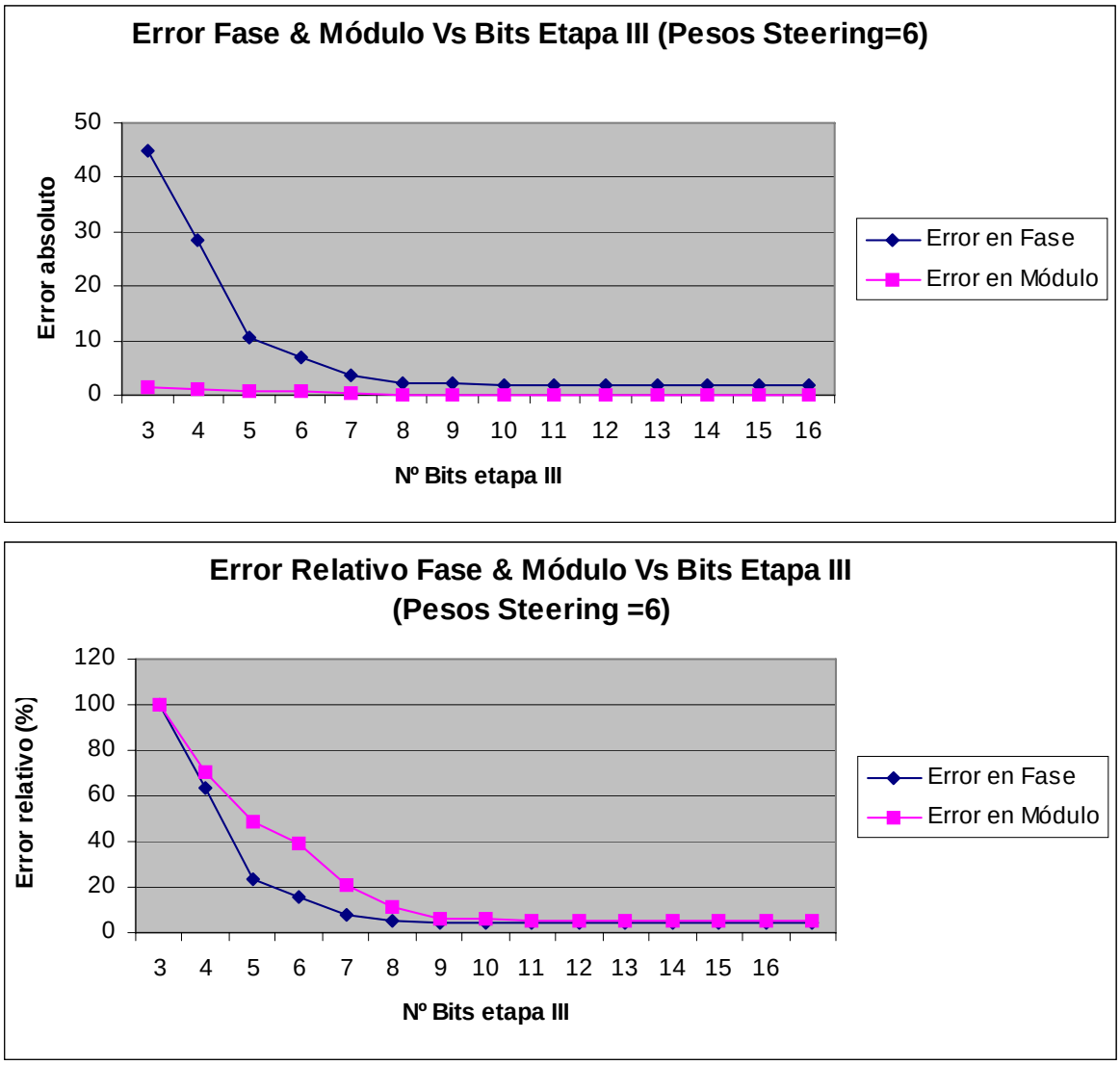


Figura 4.14. Error absoluto (arriba) y relativo (medio). Etapa III vs precisión O_{3_r} (etapa *steering*, usando pesos con 6 bits).

4.4.2.4. Etapa IV: Desarrollo de Taylor. Integración

A continuación se procede al estudio del comportamiento de la etapa IV frente al número de bits de los pesos del desarrollo de Taylor, el número de bits de la etapa IV y el número de bits del saturador (*v.* 3.2.2.4.8), cuyo objetivo es la construcción de una serie de Taylor truncada. Para ello se aplican 3 procesos:

a) Cuantización de los pesos del desarrollo de Taylor manteniendo la etapa IV con los valores en doble precisión. En esta etapa también se realiza un estudio de la densidad de puntos calculados aplicando el parámetro `UMBRAL_MASCARA`, explicando su significado y utilidad.

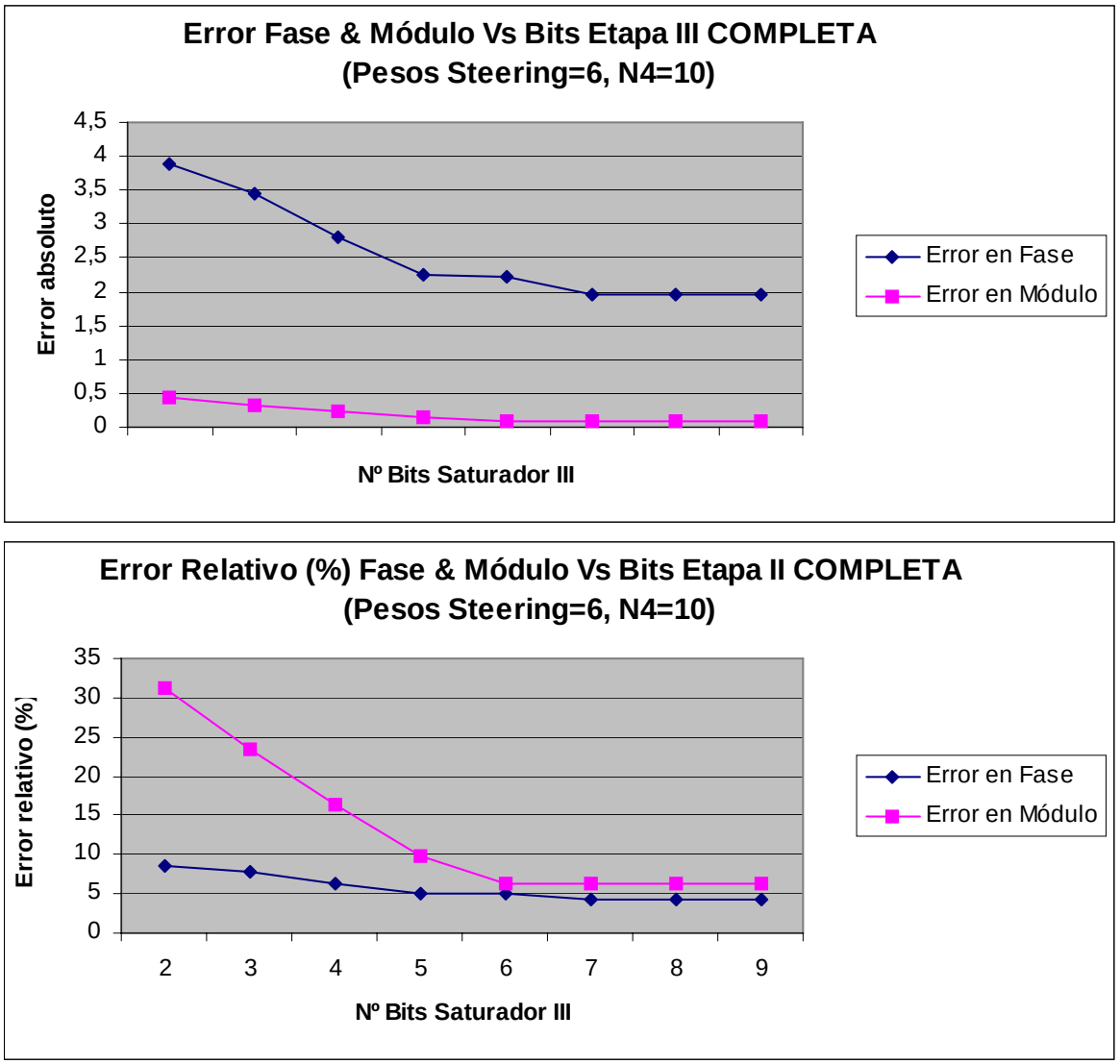


Figura 4.15. Proceso de saturación en la Etapa III. Evolución de los errores absolutos y relativos conforme aumenta el número de bits del saturador.

b) Cuantización de la etapa IV según la precisión escogida en (a).

c) Saturación de la etapa IV de acuerdo con la decisión en (b).

Posteriormente se procede a realizar un estudio paramétrico donde se representan los errores frente a la precisión, introduciendo un eje adicional de densidad debida a puntos filtrados en el camino de datos, empleando la expresión siguiente para modular la densidad de cálculo (*v.* 4.2.3).

Cuando existen variaciones temporales de intensidad muy pequeñas, existe una causa de error añadido correspondiente por un lado a las fuentes de ruido debidas a muy diversos motivos y por otro, a movimientos muy lentos. Por lo tanto se procede a modelar una "*medida de confianza*" definida por la expresión siguiente:

$$\left|\frac{dI}{dt}\right| + \left|\frac{d^2I}{dt^2}\right| \leq \text{UMBRAL\_MASCARA} \tag{4.29}$$

Esta medida de confianza filtrará los puntos cuyas variaciones temporales asociadas no sean superiores a un umbral definido. [BARR96] sugiere algo similar. Se comprueba experimentalmente una mejora significativa de los errores respecto a este umbral, (como se mostrará al final del presente capítulo) aunque como contrapartida hay una pérdida de puntos significativa debido al filtrado (no debe confundirse con la pérdida de puntos debida a la precisión). A lo largo de los capítulos que restan se volverá a hacer hincapié en este concepto.

Se muestra a continuación en la figura 4.16, un estudio del número de términos no contributivos frente al número de bits en los pesos de Taylor, como se aprecia a medida que se escogen más bits el número de términos no contributivos disminuye notablemente y consecuentemente el desarrollo de Taylor que almacena la información espacio-temporal según se explicó en el capítulo III.

También se muestra en la figura 4.17 un análisis del error relativo en función de este número de bits, pero realizando un barrido para diferentes valores del parámetro UMBRAL_MASCARA, de acuerdo con la expresión (4.29), es decir, diferentes valores de densidad de puntos que atraviesan el camino de datos. Por, por último en la figura 4.18 se realiza un estudio de la densidad de cuantización frente a la precisión de los pesos del desarrollo de Taylor, necesitando 16 bits para tener un 100% de densidad.

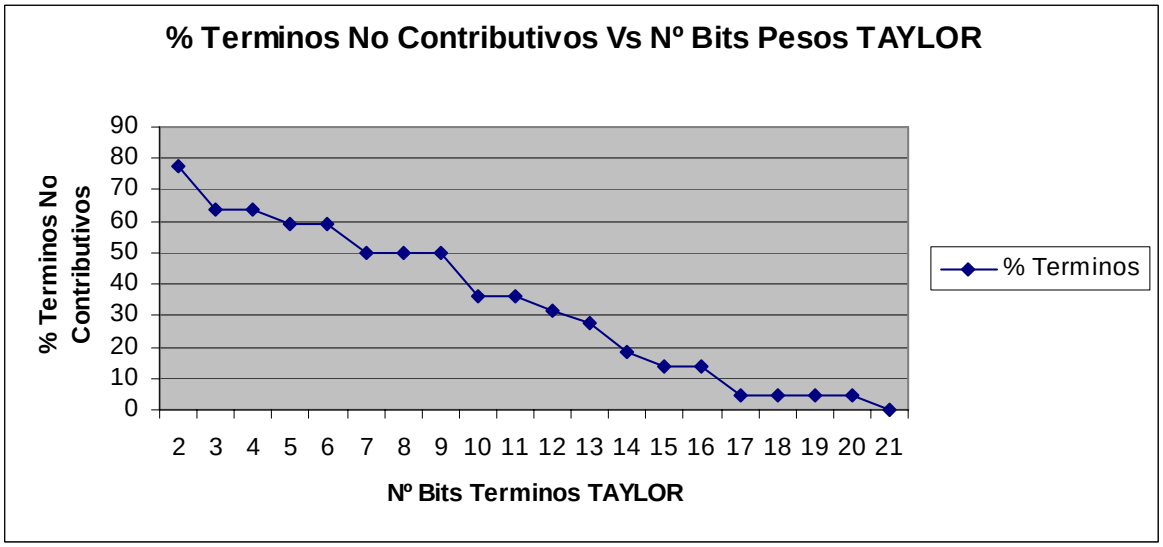


Figura 4.16. % Pesos contributivos del desarrollo de Taylor (cuantización)

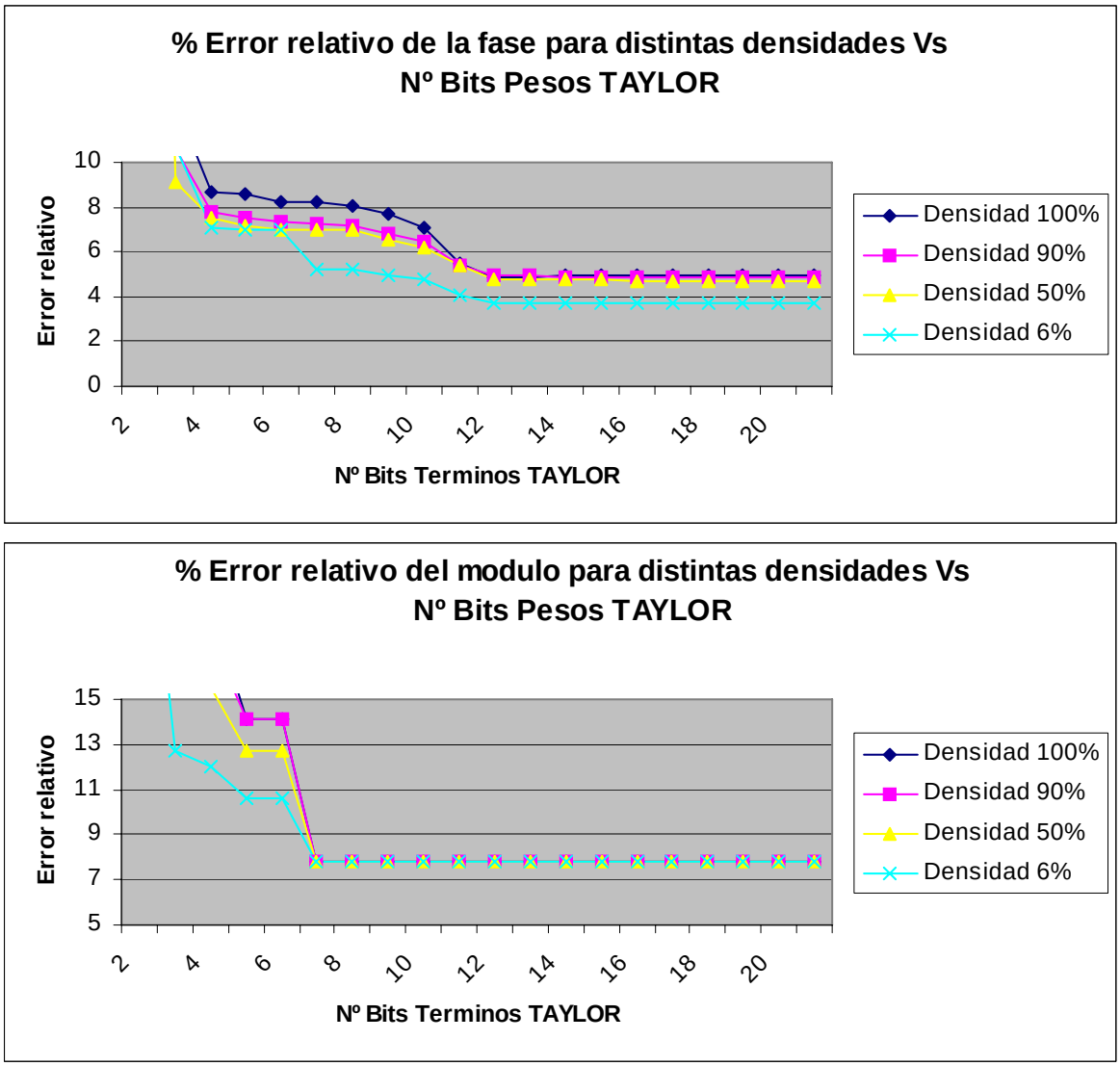


Figura 4.17. Error relativo fase (superior) y módulo (abajo) debido al numero de bit de pesos de Taylor para diferentes densidades debidas al UMBRAL MASCARA.

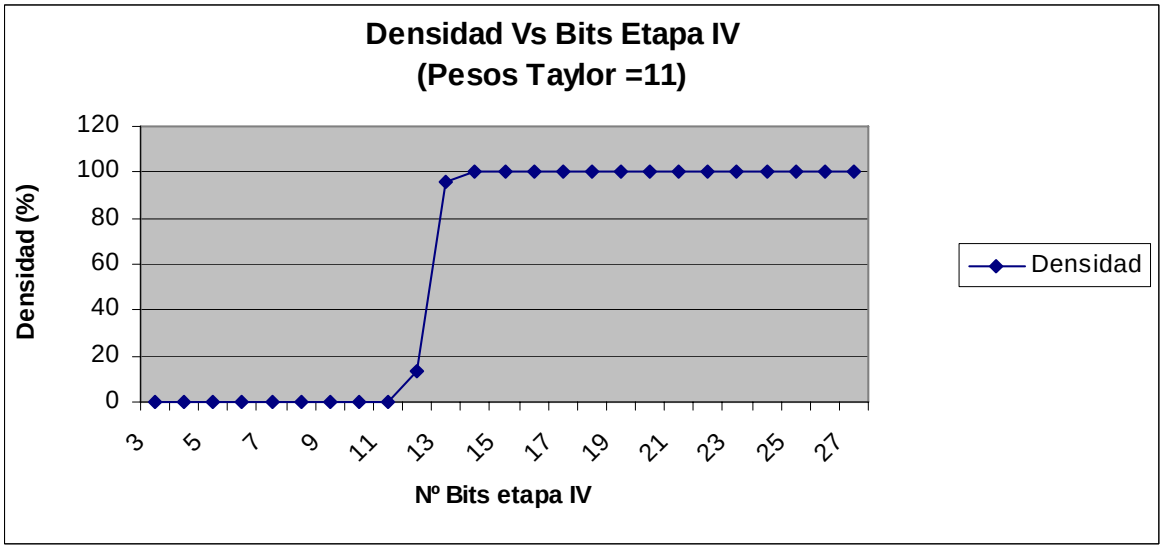


Figura 4.18. Estudio de la etapa de densidad (cuantización)

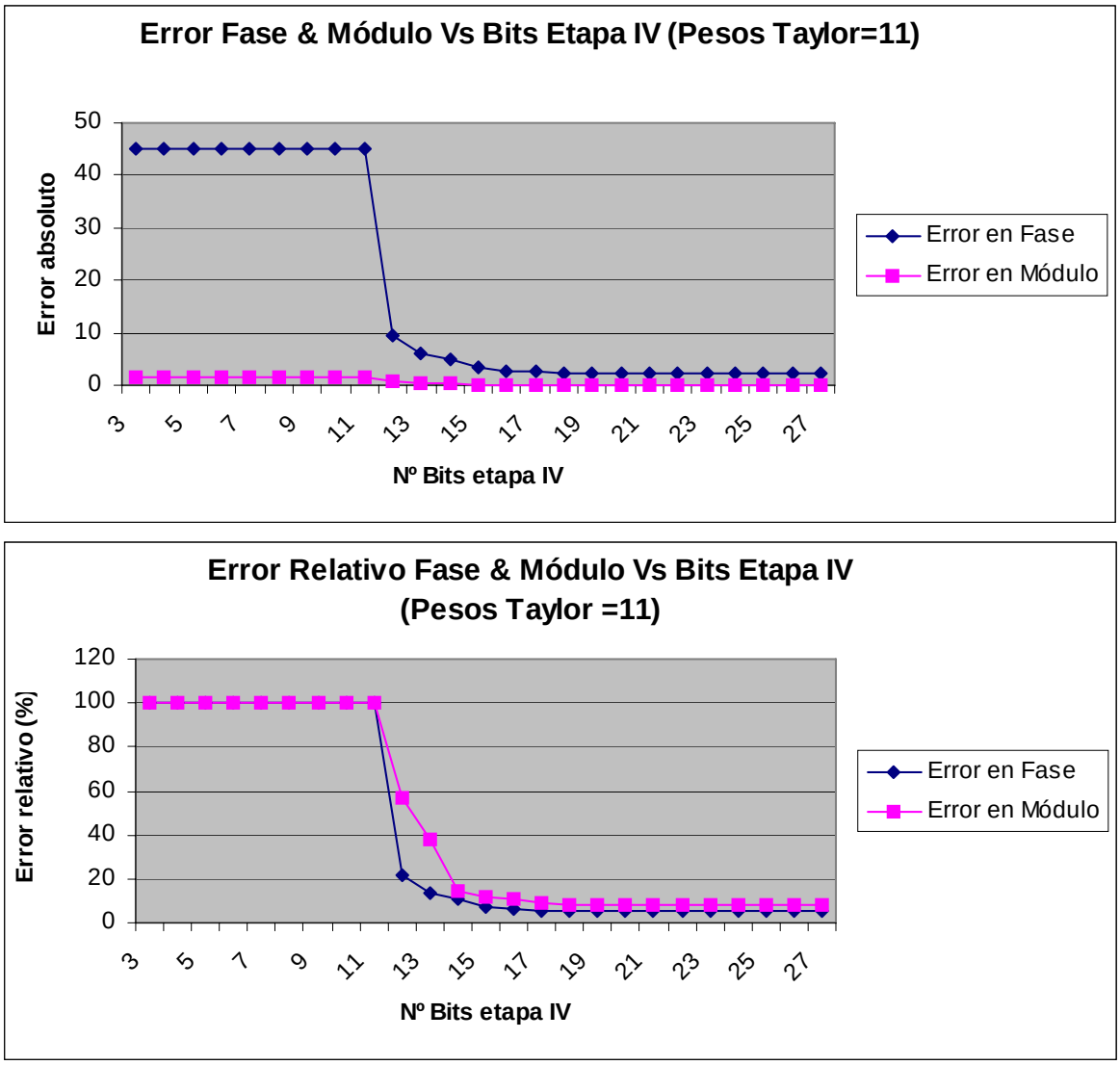
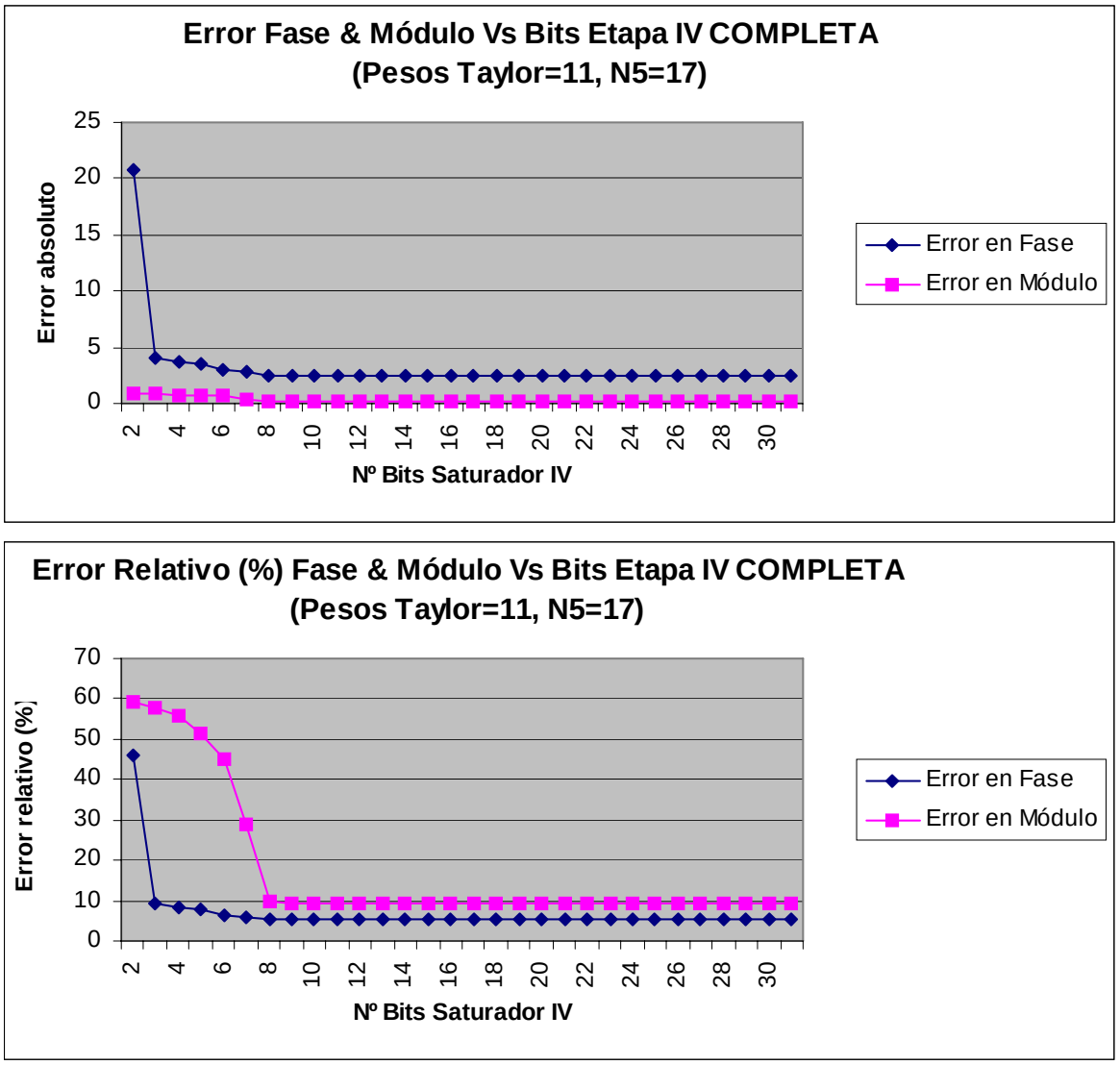


Figura 4.19. Error absoluto (arriba) y relativo (abajo) Etapa IV vs precisión O_{4_r} (para pesos Taylor con 11 bits). Proceso de cuantización.

Se puede corroborar como una elección de 11 bits de precisión para los pesos de Taylor, (que equivale a un 65% de los pesos originales), implica una medida de error absoluto (relativo) de 2.45/0.11 (5.44%/7.78%).(los valores están expresados, según la notación usual que se sigue en este capítulo).

Con respecto a los errores de cuantización, éstos ascienden a 2.49/0.13 (5.33/9.19) si se opta por O_{4_r}=17 bits, según muestra la figura 4.19.

Posteriormente, como resultado de la aplicación del proceso de saturación, se consiguen estos mismos resultados, aunque ahorrando una considerable cantidad de bits (concretamente 9), la figura 4.20 expresa gráficamente como la mayor parte de la información en esta etapa del algoritmo está acotada inferiormente por el saturador.



Figuras 4.20 Proceso de saturación en la Etapa IV . Evolución de los errores absolutos (arriba) y relativos (abajo) conforme aumenta el número de bits del saturador. O_{4_r} =10 bits.

4.4.2.5. Etapa V: Primitivas de flujo óptico

Continuando con el planteamiento que se viene repitiendo desde el apartado 4.4.2 se procede al estudio del comportamiento de la etapa V frente al número de bits de la misma y del saturador. Para ello se aplicarán dos procesos:

a) Cuantización de la etapa V.

b) Saturación de acuerdo con (a).

La figura 4.21 representa los resultados del proceso de cuantización obteniéndose un error de 2.60/0.19 (5.77/13) si la elección de O_{5_r}=12 bits. Una vez aplicado este proceso, se procede a saturar mediante 6 bits, permitiendo ahorrar otros 6, según se aprecia en la figura 4.22.

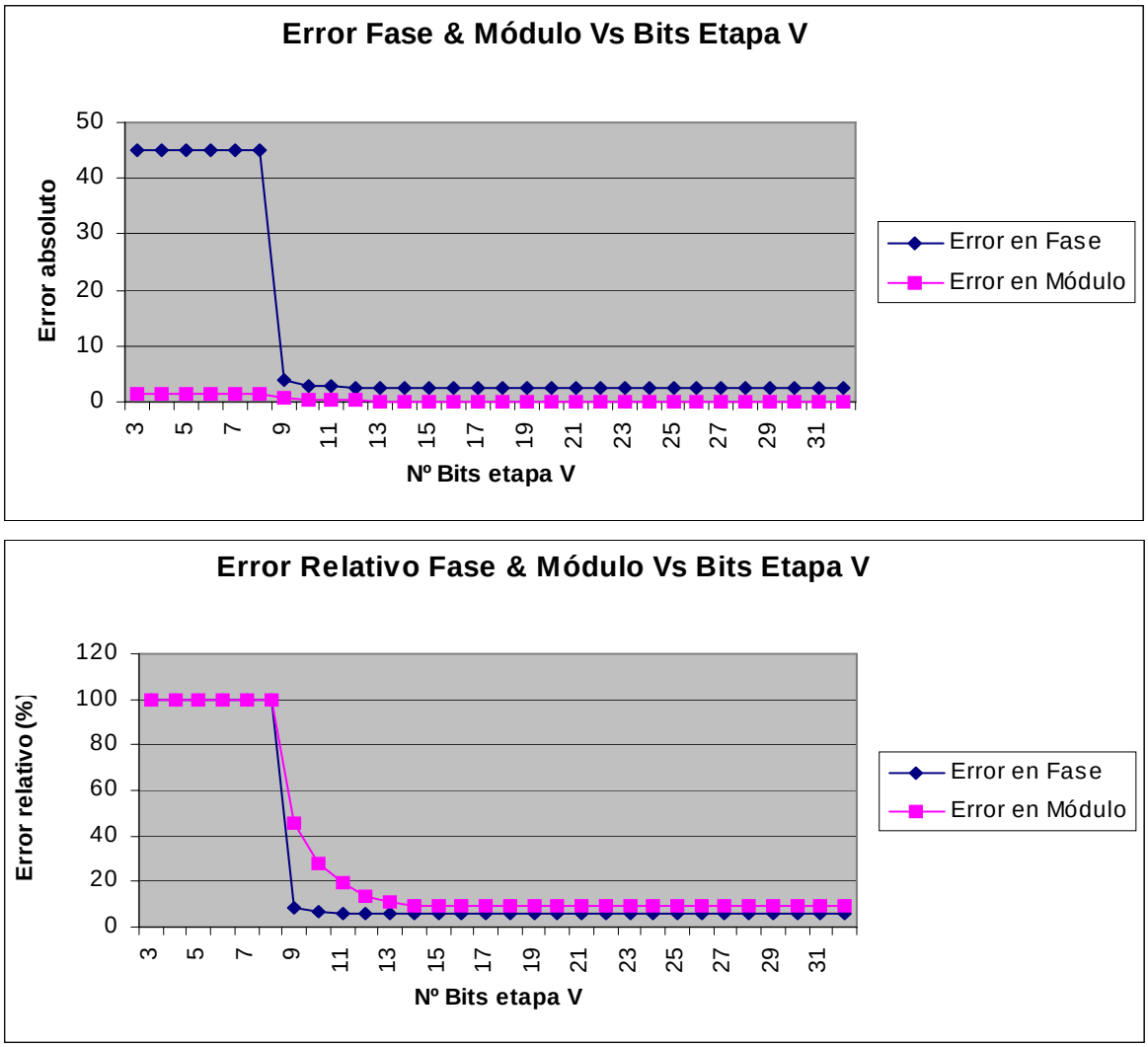


Figura 4.21. Error absoluto (superior) y relativo (inferior). Etapa V vs precisión O_{5_r}. Proceso de cuantización.

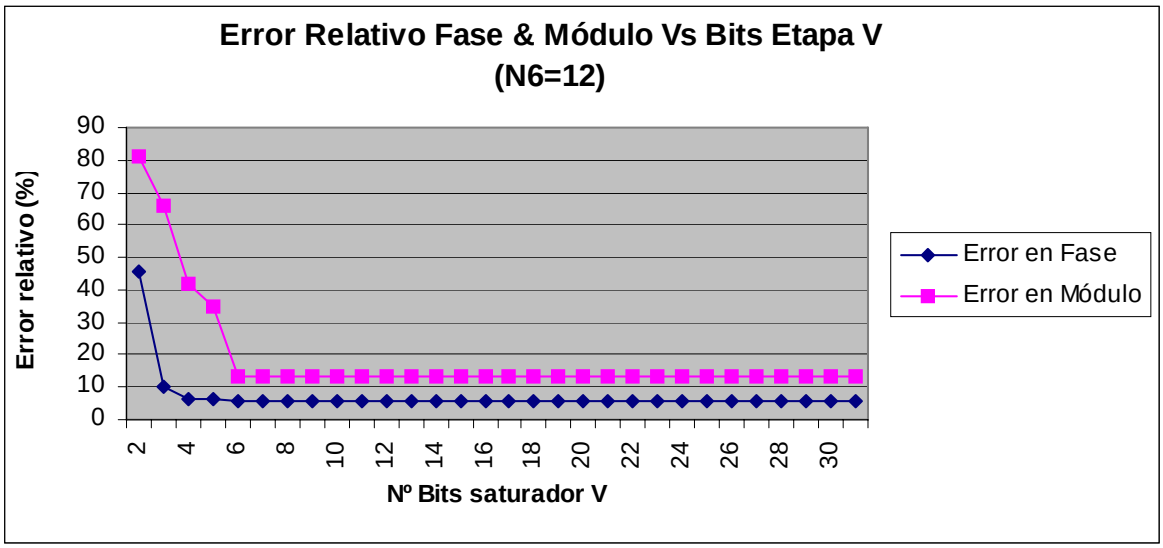


Figura 4.22. Proceso de saturación en la Etapa V. Evolución de los errores relativos conforme aumenta el número de bits del saturador. Elección de 6 bits para esta etapa (zona de estabilización) y ahorro de 6 bits globales.

4.4.2.6. Etapa VI: Extractores de módulo y fase (I)

4.4.2.6.1. Etapa VI.I: Extractores de módulo I

A partir de la etapa anterior, se divide el tronco común de operaciones en dos ramas independientes, que llevan a calcular módulo y fase. En esta etapa se va a calcular, las primitivas extractoras de módulo, mediante un estudio de precisión de la etapa VI.I. (*v.* 4.2.2.VI) aplicando sendos procesos de cuantización:

a) Cuantización de la etapa VI.I. (expresiones (4.10.a-d)).

b) Cuantización de la etapa VI.I. (expresiones (4.11 a-d)).

El análisis del módulo se representa en la figura 4.23. Escogiendo 13 bits para N7t y10 para N7b, el error absoluto (relativo) asciende hasta 0.65 (46.01%). Para mantener el error dentro del margen anterior hay que incrementar el valor de la precisión hasta 66 bits.

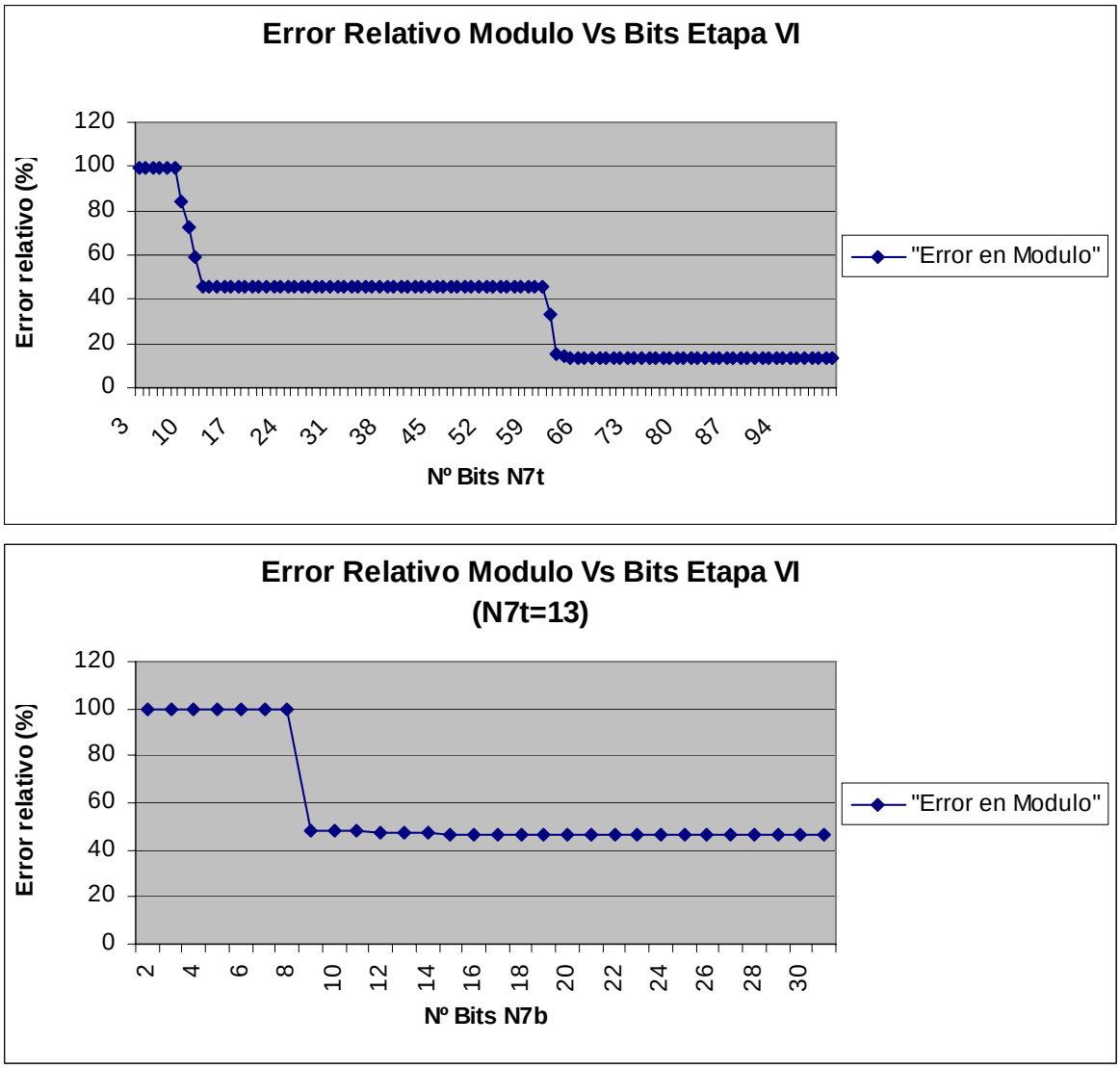


Figura 4.23. Error relativo Etapa VI (Módulo) vs precisión $O_{6\_T\_r}$.(arriba).$O_{6\_B\_r}$.(abajo). Proceso de cuantización.

4.4.2.6.2. Etapa VI.II: Extractores de fase I

Idénticamente que se ha hecho en la etapa anterior, se procede aquí a estudiar el comportamiento de la etapa VI.II aplicando 2 procesos:

a) Cuantización la etapa VI.II;ecuaciones (4.13a, 4.13b).

b) Saturación de la etapa VI.II de acuerdo con (a).

Se representa en la figura 4.24 el comportamiento de la cuantización. Con una opción de 13 bits de precisión, se obtiene un error absoluto (relativo) de 2.73 (6.06%), con respecto a la saturación (figura 4.25) se mantiene el mismo error 2.73 (6.06%) con un ahorro neto de 7 bits.

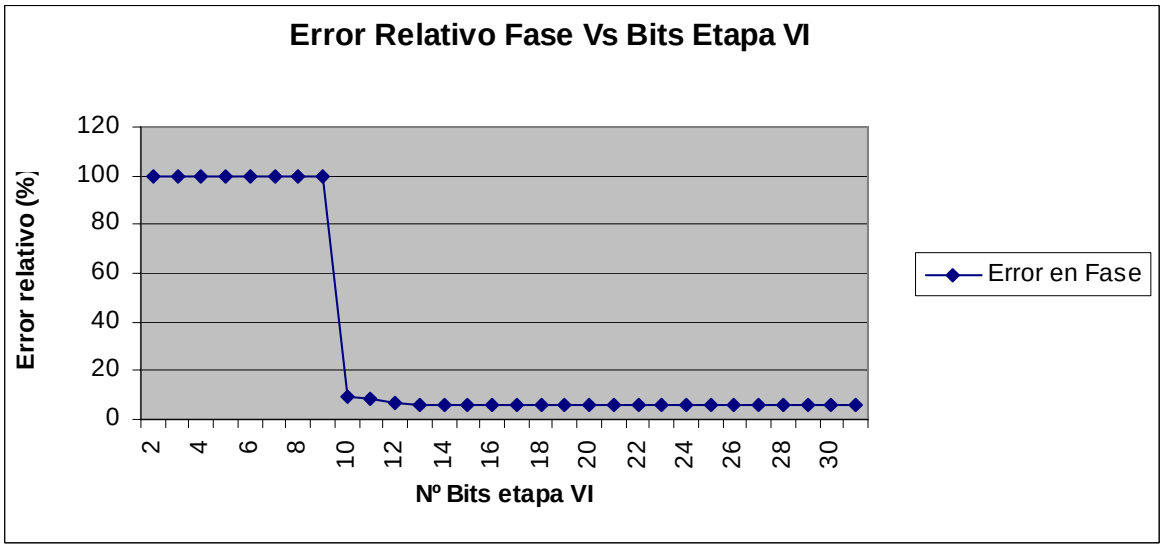


Figura 4.24. Error relativo Etapa VI (Fase) vs precisión O_{6M_r}. Proceso de cuantización.

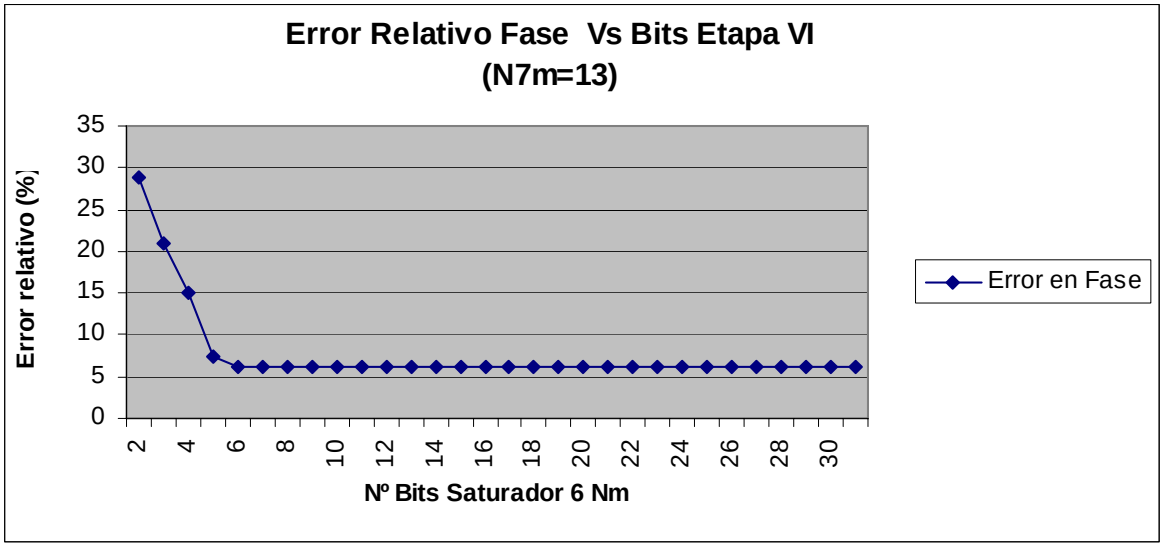


Figura 4.25. Proceso de saturación en la Etapa VI (Fase). Evolución del error relativo conforme aumenta el número de bits del saturador. Se eligen 6 bits (zona de estabilización) y se ahorran 7 bits globales.

4.4.2.7. Etapa VII: Extractores de módulo y fase (II)

4.4.2.7.1. Etapa VII.I: Extractores de módulo II

Estudio de la etapa VII.I aplicando 2 procesos:

a) Cuantización de la etapa VII.I. Ecuación. (4.15.a);

b) Cuantización de la etapa VII.I .Ecuación .(4.15.b).

Se opta por 14 bits de precisión, con un error absoluto (relativo) de 0.68 (48.03%). En la figura 4.26 se muestra como una elección de 13 bits de precisión, implica incrementar el error absoluto (relativo) hasta unos valores de 0.7 (49%), alcanzándose un más que aceptable 98% de densidad, según indica la evolución de la gráfica correspondiente a la figura 4.27.

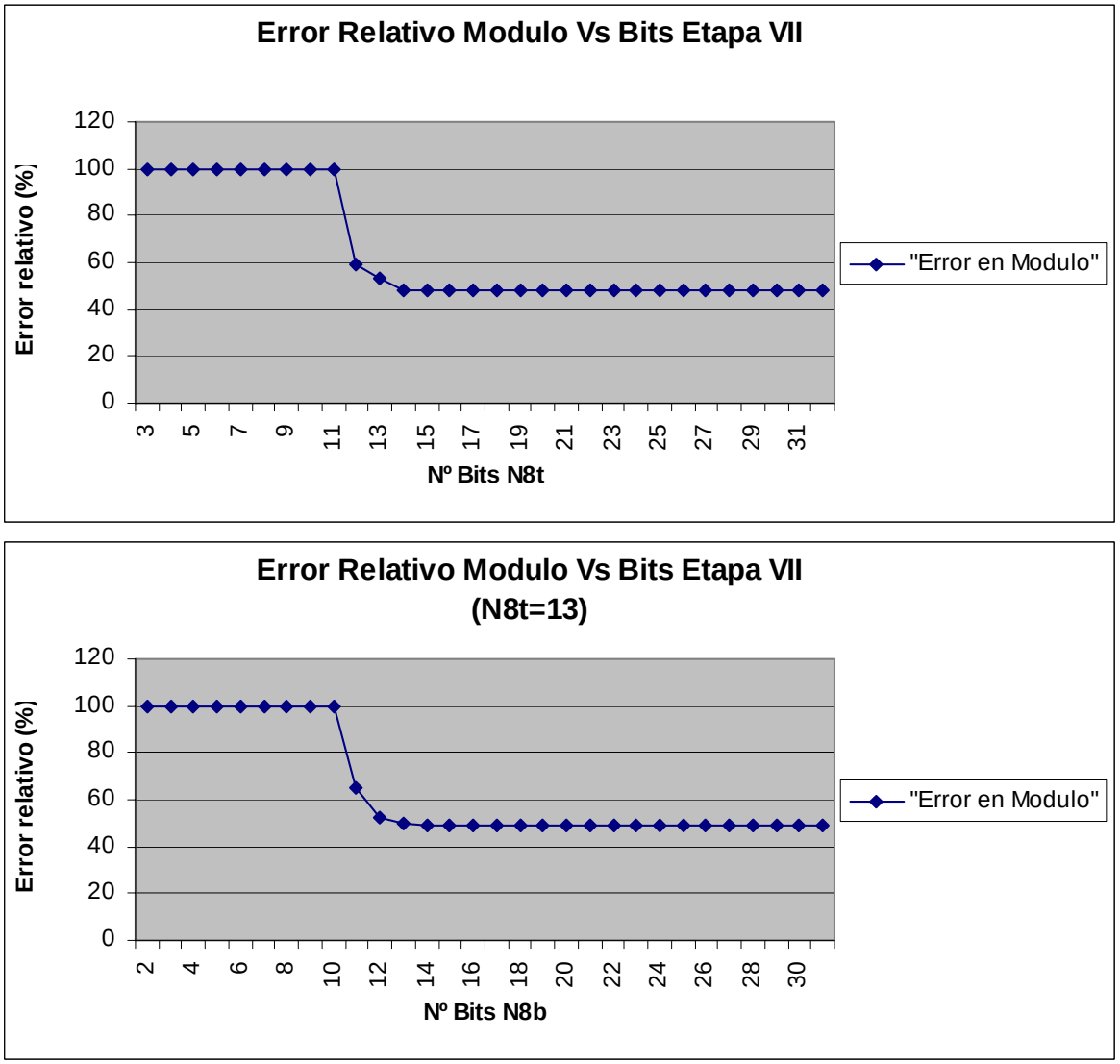


Figura 4.26. Error relativo (arriba) Etapa VII (Módulo) vs precisión $O_{7\_T\_r}$. (abajo) $O_{7\_B\_b}$. Proceso de cuantización.

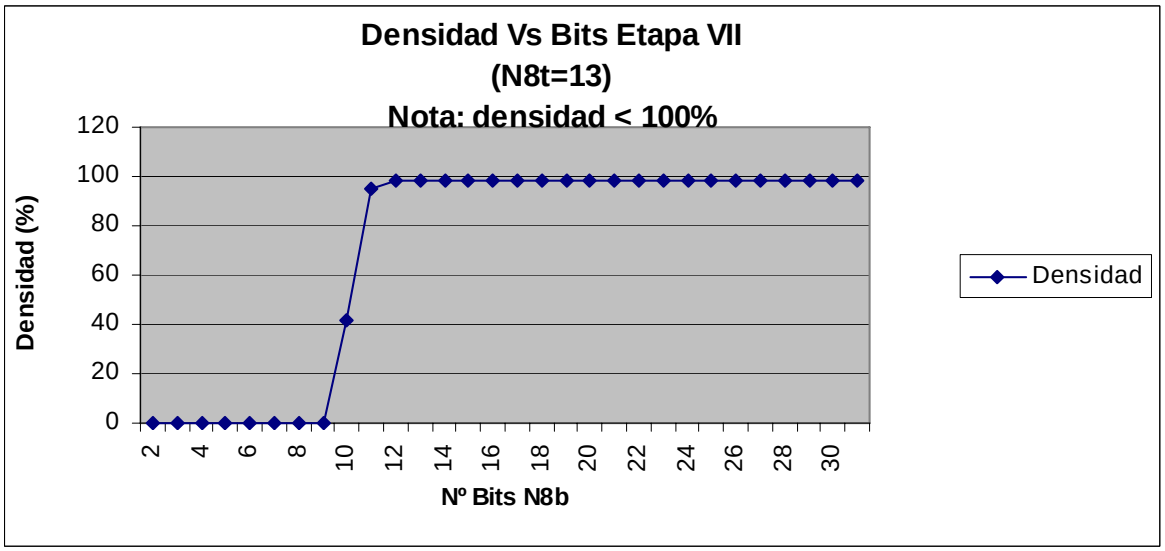


Figura 4.27. Evolución de la densidad, Etapa VII (Módulo) vs precisión $O_{7\_B\_b}$. Proceso de cuantización.

4.4.2.7.2. Etapa VII.II: Extractores de fase II

Estudio del comportamiento de VII.II. aplicando 2 procesos:

(a) Cuantización de la etapa VII. II. Ecuación (4.17).

(b) Saturación de la etapa VII.II.

La elección escogida en este caso, corresponde a unos valores de de 11 bits para cuantización y 6 bits para la etapa de saturación, alcanzándose entonces un estado cuyo error absoluto (relativo) asciende a 2.76 (6.13%), según puede observarse en las figuras 4.28 y 4.29.

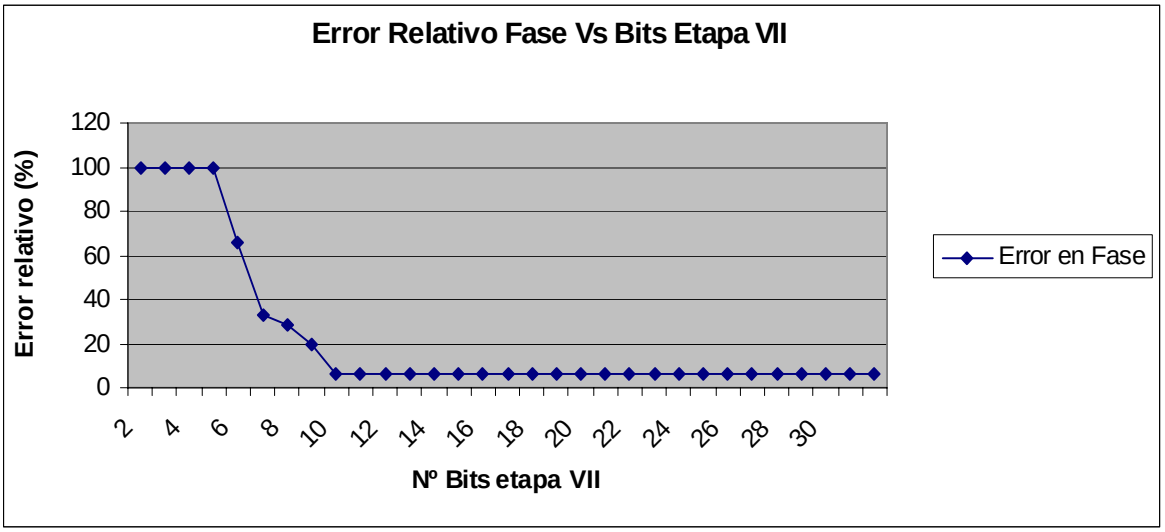


Figura 4.28. Error relativo. Etapa VII.II (Fase) vs precisión $O_{7M\ r}$. Proceso de cuantización.

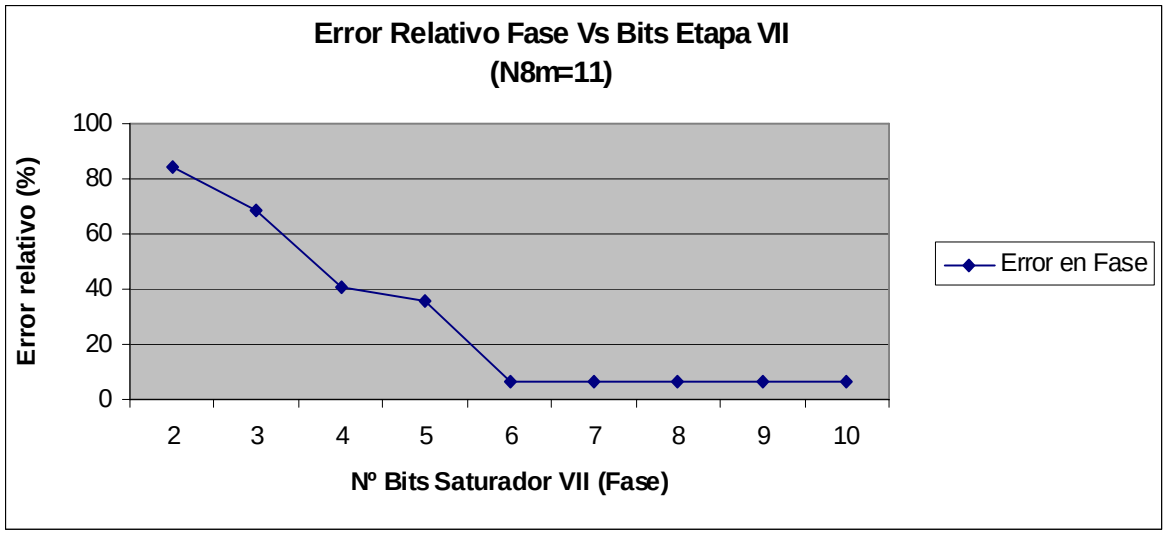


Figura 4.29. Proceso de saturación. Elección de 6 bits (zona de estabilización) y ahorro de 5 bits.

En la figura 4.30 se representa una figura tridimensional que da cuenta de la evolución del error frente a la densidad de puntos (o la medida de confianza de la expresión 4.29) y el número de bits para la fase. Se puede apreciar claramente una tendencia cualitativa a disminuir el error a medida que lo hace la densidad, indicando un comportamiento similar respecto al número de bits de cuantización usados en esta etapa, según indica la figura 4.31. En cualquier caso para 100% de densidad el error total se mantiene en torno a un 6%.

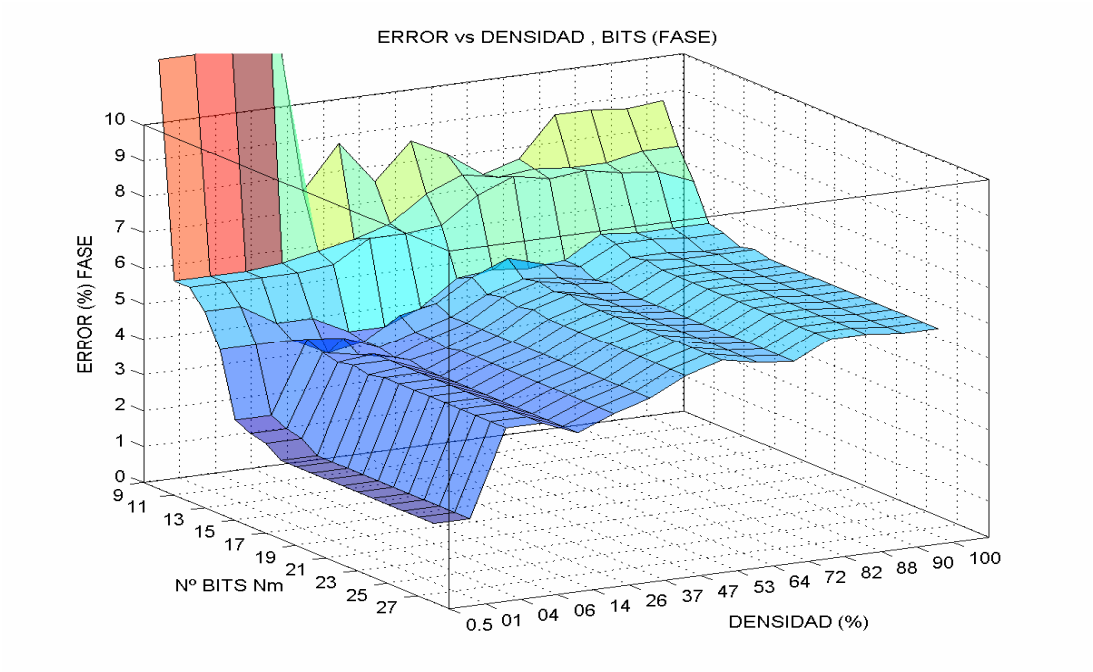


Figura 4.30. Evolución del error relativo en función de la densidad debida a la "medida de confianza" y el número de bits de cuantización.

4.4.2.8. Etapa VIII Extractores de módulo y fase III

4.4.2.8.1. Etapa VIII.I. Extractores de módulo III

Esta es la última etapa de procesamiento donde se calcula el valor módulo, finalizando con este proceso una de las ramas del análisis global por etapas. El estudio de precisión controlada implica una cuantización final de acuerdo con la expresión (4.19). Con una elección de 14 bits (figura 4.31) se obtienen unos errores relativos correspondientes a 49.49%.

4.4.2.8.2 Etapa VIII.II. Extractores de fase III

Esta es la última etapa de procesamiento donde se calcula la fase de acuerdo a la ecuación 4.21. Se requieren 11 bits para mantener el error alrededor de 6.19%, según ilustra la figura 4.31 y abandonar el error final en un nivel aceptable. Con este último cálculo se completan todas las etapas del algoritmo, finalizando el segundo ramal perteneciente al cálculo de la fase de cada punto en todo el camino de datos.

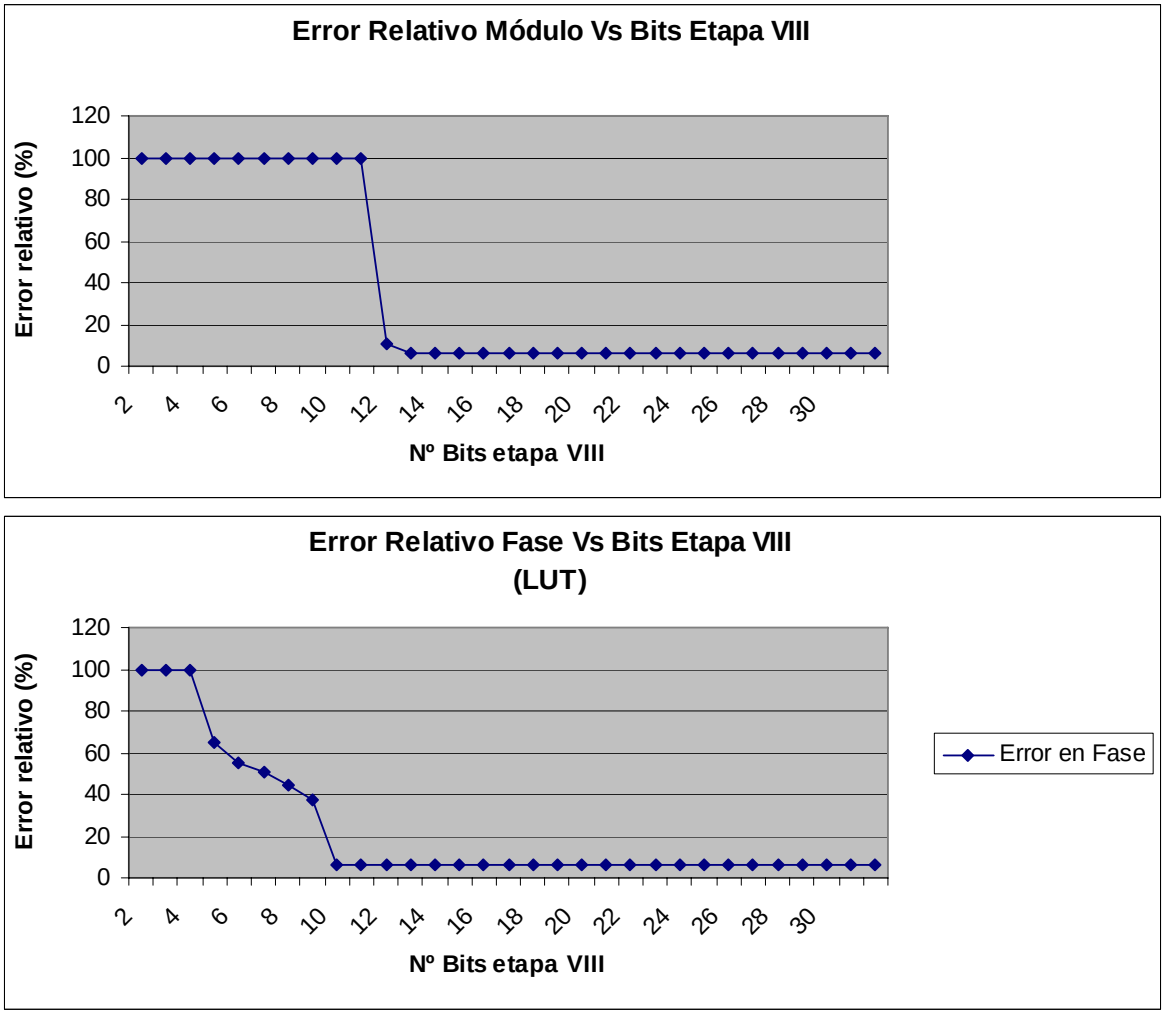


Figura 4.31. Error relativo (superior) Etapa VIII (Módulo) vs precisión O_{8_r}. Proceso de cuantización. Error relativo (inferior) Etapa VIII (Fase) vs precisión de la LUT para la función arcotangente.

4.4.3. Resumen de todas las etapas anteriores: comunes I,II,III,IV,V y propias del módulo (VI Top, VI Bot, VII Top, VII Bot, VIII) y la fase (VIm, VII m, VIIIm)

A continuación, se muestran superpuestas a modo de resumen general, todas las gráficas obtenidas en módulo y fase para varios intervalos suficientemente ilustrativos, de forma que se pueda inspeccionar adecuadamente el comportamiento de cada una de estas etapas a diferentes niveles de profundidad espacial. En las dos figuras siguientes (4.32-4.33) se ilustra el error relativo en módulo y fase para cada una de las etapas encadenadas con un rango de error porcentual hasta 100. En las figuras 4.34 aparece el error del módulo correspondiente a los tramos 0-10%, 45-55% apreciándose como cada etapa conceptual introduce un pequeño error incremental con respecto a la anterior.

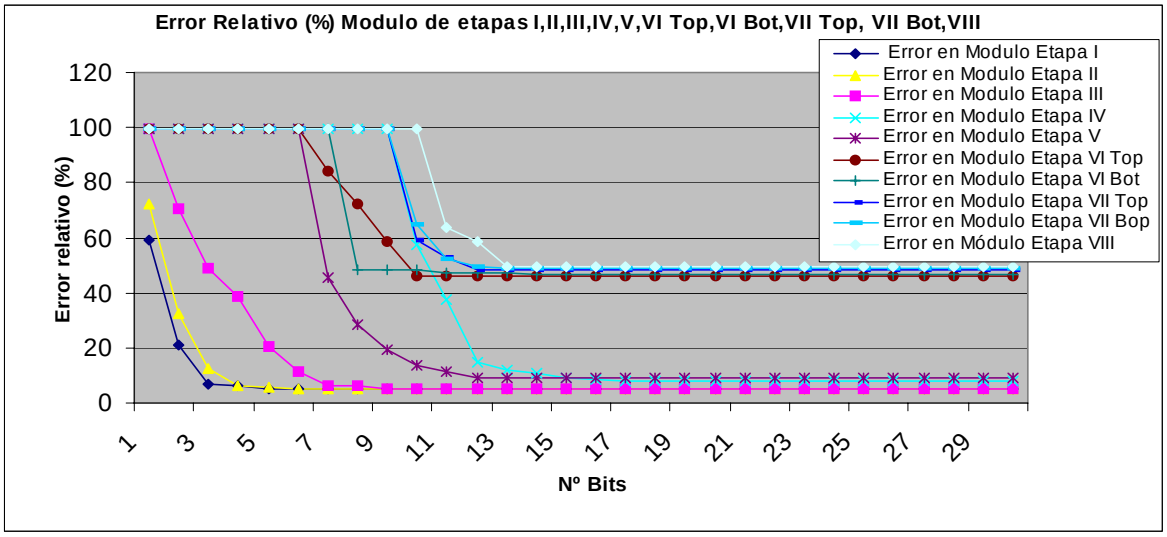


Figura 4.32. Error relativo (Módulo) para cada una de las etapas conceptuales.

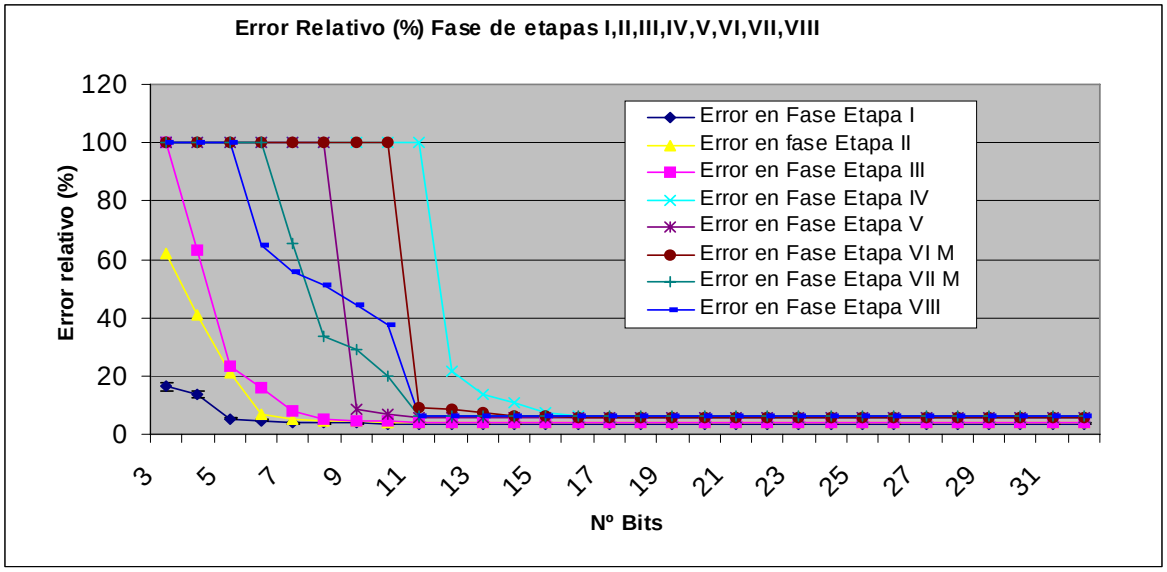


Figura 4.33. Error relativo (Fase) para cada una de las etapas conceptuales.

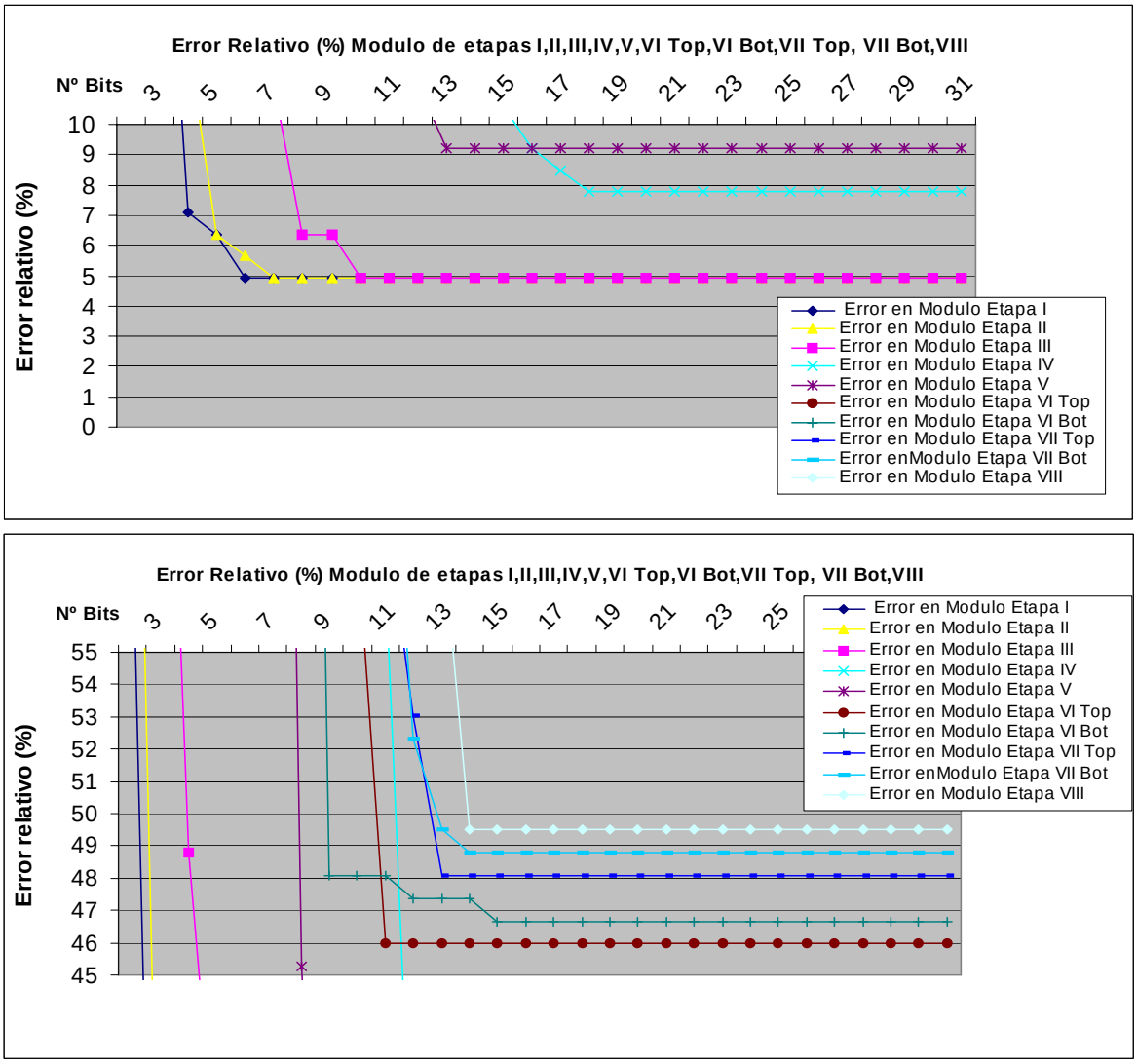


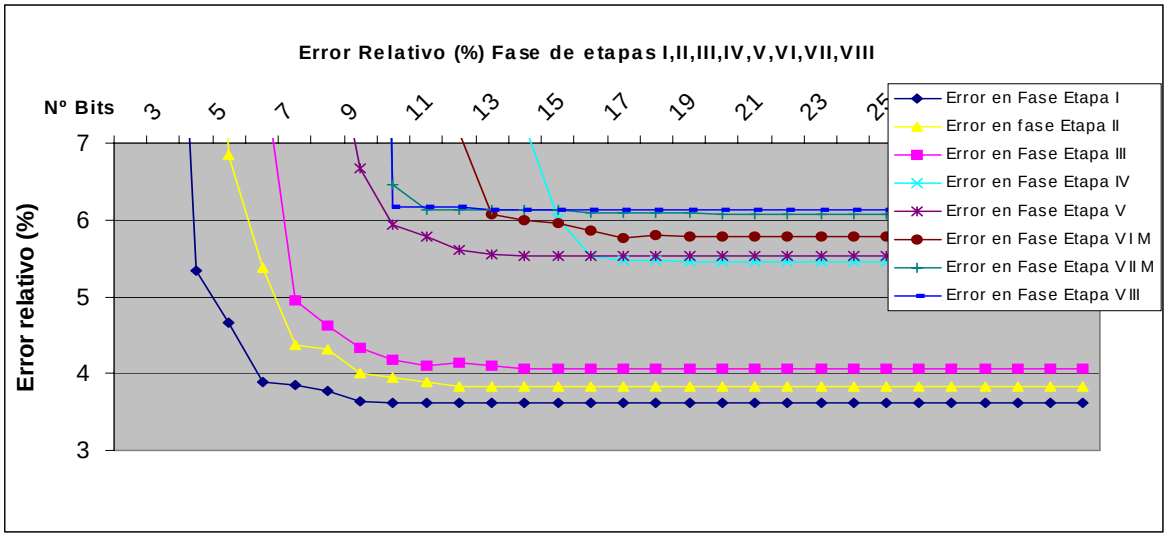
Figura 4.34. Error relativo (Módulo) para diferentes rangos de error (eje de ordenadas), 0%-10% (arriba), 45%-55% (abajo) .

El salto de error se hace más evidente a partir de la etapa VI, debido a los cocientes asociados. En la figura 4.35 amplia la representación de la figura 4.33 en el recorrido de 3-7%, apareciendo cambio más pronunciado a partir de la etapa V (serie de Taylor y elección del número de términos representativos de los canales portadores de información).

4.5. Resultados obtenidos

4.5.1. Introducción

La tabla 4.II muestra el resumen de los parámetros obtenidos con el análisis expuesto en este capítulo. Los bordes discontinuos en la representación de las celdas correspondientes a las etapas I-IV, indican que existe un cauce común de procesamiento. A partir de ahí, se distinguen 3 etapas diferenciadas para módulo y fase netamente independientes unas de otras.



Figuras 4.35. Error relativo (Fase) para un rango de visualización correspondiente a 3%-7%.

TABLA 4.II Resumen de parámetros obtenidos en la plataforma experimental de precisión controlada.

	Parámetros:	***Error absoluto (Fase) Mod***	***Error relativo (%) (Fase) Mod***
ETAPA I	$I_1 = 8$ $F_1=6$ $O_{1\_r}=9$ $S_1=9$	(1.64)	(3.64)
		0.07	4.95
ETAPA II	$F2=8$ $O_{2\_r\_I}=9$ $O_{2\_r\_II}=10$ $S_2=8$	(1.8)	(4)
		0.07	4.95
ETAPA III	$W_3=6$ $O_{3\_r}=10$ $S_3=6$	(2.23)	(4.95)
		0.09	6.36
ETAPA IV	$W_4=11$ $O_{4\_r}=17$ $S_4=9$	(2.49)	(5.53)
		0.13	9.19
ETAPA V	$O_{5\_r}=12$ $S_5=6$	(2.60)	(5.77)
		0.19	13
ETAPA VI.I	$O_{6T\_r}=13$ $O_{6B\_r}=10$	0.68	48
ETAPA VI.II	$O_{6M\_r}=13$ $S_{6M}=6$	(2.73)	(6.06)
ETAPA VII.I	$O_{7T\_r}=14$ $O_{7B\_r}=13$	0.7	49
ETAPA VII.II	$O_{7M\_r}=11$ $S_{7M}=7$	(2.76)	(6.13)
ETAPA VIII.I	$O_{8\_r}=14$	0.72	49.54
ETAPA VIII.II	$O_{8M\_r}=10$	(2.78)	(6.19)

4.5.2. Comparativas con otras métricas y estímulos

A los tres estímulos iniciales diseñados con Matlab, se han añadido otros dos creados por David Fleet, de la Universidad de Toronto [FLE92] que suelen usarse al hacer comparativas de medidas de flujo. Estas imágenes están creadas mediante el movimiento relativo de una cámara y la imagen del paisaje de fondo.

4.5.2.1. Estímulo "Diverging Tree"

Diverging Tree, es una secuencia donde aparece movimiento de un árbol en expansión (a modo de *zoom*) con un rango de velocidades asimétrico dependiendo la localización de los puntos en el fotograma. Este rango varía desde 0 (en el foco de expansión central) hasta 1.4 y 2 puntos/fotogramas en la izquierda y derecha, respectivamente. La figura 4.36 muestra 3 fotogramas de la secuencia así como su salida teórica (representada en una escala de 6×6 puntos).

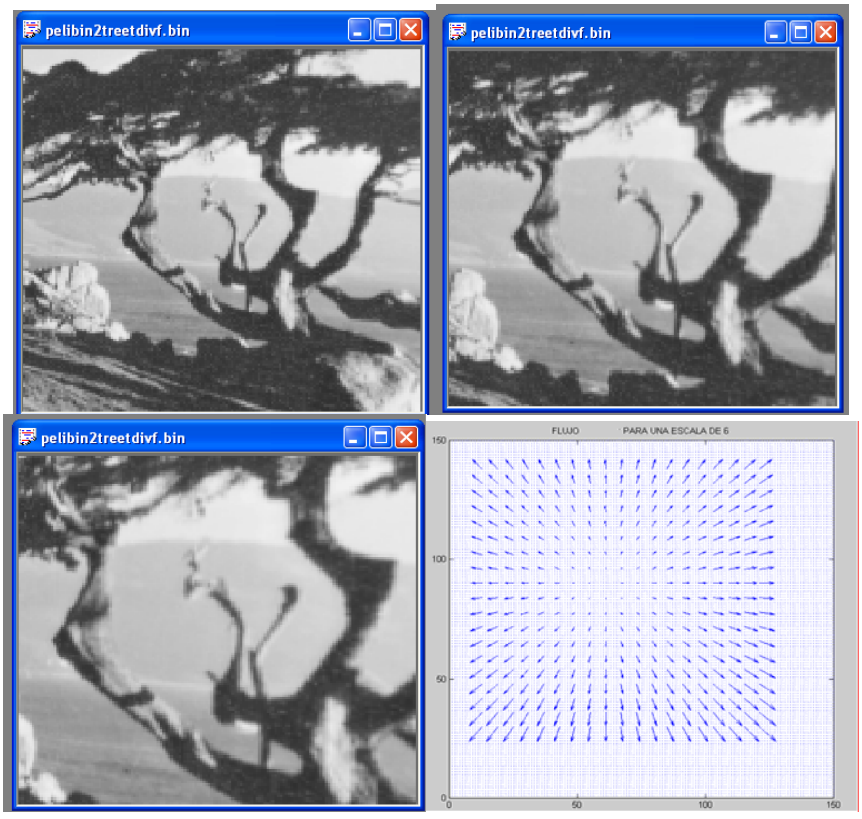


Figura 4.36. 3 Fotogramas del estímulo "*Diverging_Tree*" (150×150) y el flujo teórico agrupado en una representación de 6x6 unidades.(para su mejor representación).

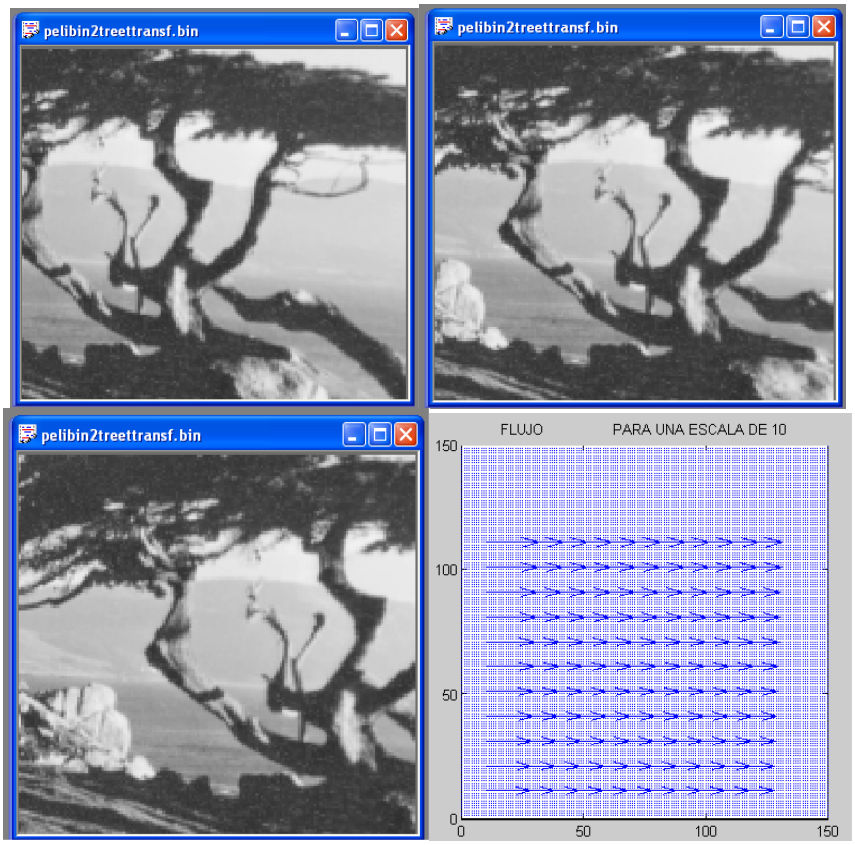


Figura 4.37. 3 Fotogramas del estímulo *"Translating Tree"* (150×150) y el flujo teórico agrupado en macropuntos de 10×10 unidades.

4.5.2.2. Estímulo "Translating Tree"

Translating Tree, es una secuencia donde aparece movimiento de un árbol en translación, aunque también con una rango de velocidades asimétrico dependiendo la localización de los puntos en el fotograma. Este rango varía desde 0 hasta 1.73 y 2.3 puntos/fotogramas en la izquierda y derecha, respectivamente. La figura 4.37 muestra 3 fotogramas de la secuencia, así como su salida teórica (representada en una escala de 10×10).

4.5.2.3. Resultados finales. Comparación entre *software* y la plataforma

En la tabla 4.III de la página siguiente, se muestran los errores según diferentes métricas (la usada en función de la diferencia de módulo y fase, la de Galvin usual, Galvin normal, y la usada por Barron). Como se aprecia, la combinación de estímulos I,II,III es la que proporciona un menor error, esto se puede justificar debido a la regularidad en las frecuencias espaciales y la homogeneidad de las teselaciones del estímulo durante todo el fotograma.

TABLA 4.III Errores según diferentes, métricas para la versión SW y HW (precisión controlada).

	SOFTWARE					**SEMI *HARDWARE***				
	Barron	**Galvin**	**Galvin ⊥**	**Fase**	**Mod**	**Barron**	**Galvin**	**Galvin ⊥**	**Fase**	**Mod**
Estímulos (I+II+III)	1.61±1.00	0.02±0.06	0.019±0.002	1.56±1.30	0.04±0.03	11.39±6	0.32±0.09	0.04±0.03	2.78±2.03	0.70±0.12
Div.Tree	12.46±8.00	0.29±0.15	0.18±0.10	13.9±11.0	0.93±0.21	17.8±11	1.2±0.20	0.98±0.07	15.1±12.03	2.1±0.42
Trans.Tree	18.01±15.00	0.69±0.40	0.13±0.03	9±7	1.52±0.40	21.73±18	2.9±0.49	0.56±0.08	13.76±10.00	3.5±0.72

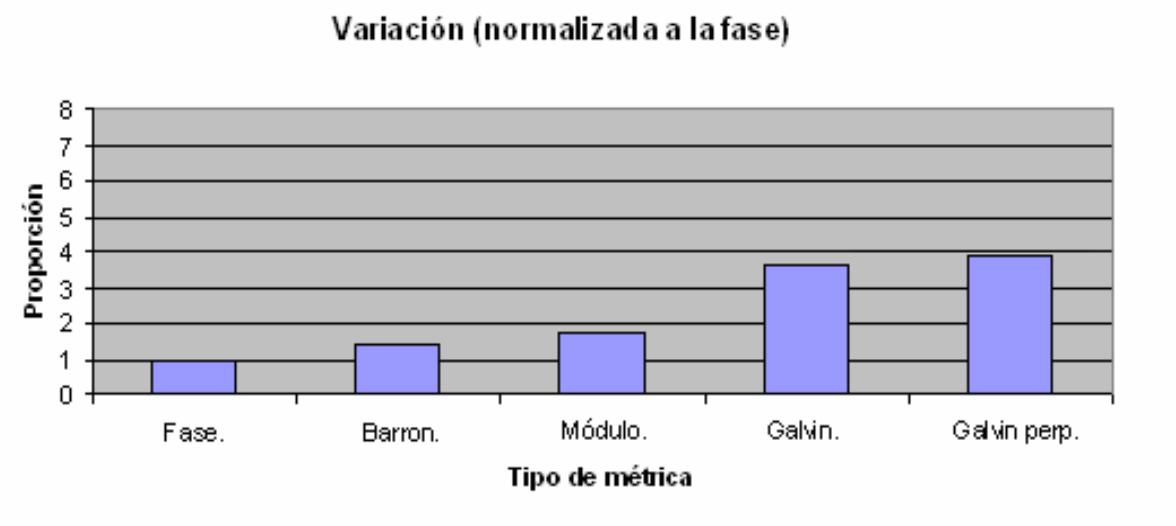


Figura 4.38. Variación de la métrica al pasar del sistema *software* al *semihardware.*

Sin embargo cuando se considera el estímulo “Diverging Tree”, los errores suben de forma notable, llegando elevándose aproximadamente un orden de magnitud en para las plataformas *software* y *semihardware* , debido al gran abanico de frecuencias espaciales y temporales que aparecen. Los peores resultados aparecen con “Translating Tree” incrementándose los errores hasta una media de casi 14 º en fase y 3.5 en módulo frente a 9º y 1.52 en *software.* De nuevo se justifican estas medidas por la dificultad del estímulo a tratar.

Cabe destacar también como según estos experimentos la métrica más conservativa al respecto (la que menos varía respecto al *software*) es la medida de la diferencia de fases y la que más varía es la métrica de Galvin perpendicular, según apunta la figura 4.38. Respecto a la medida de confianza, que venía descrita en la ecuación (4.29), se ha realizado un análisis de su bondad midiendo como evoluciona el error según las diferentes métricas respecto a los puntos adquiridos según este umbral.

Para ello se ha representado en las figuras 4.39-43 la evolución de los errores del sistema *software* y *semihardware* (cada figura contiene la evolución de 2 gráficas que deben ser interpretadas según cada tipo de eje de ordenadas) frente a la densidad, repitiéndose esta representación para cada tipo de métrica a considerar.

Se establece como conclusión que la "medida de confianza" es apropiada, puesto que disminuye el error en cada una de las métricas a medida que la expresión definida en (4.29) filtra los puntos de cálculo en las etapas preliminares del camino de datos, aunque como contrapartida la densidad total de cálculo de puntos disminuye.

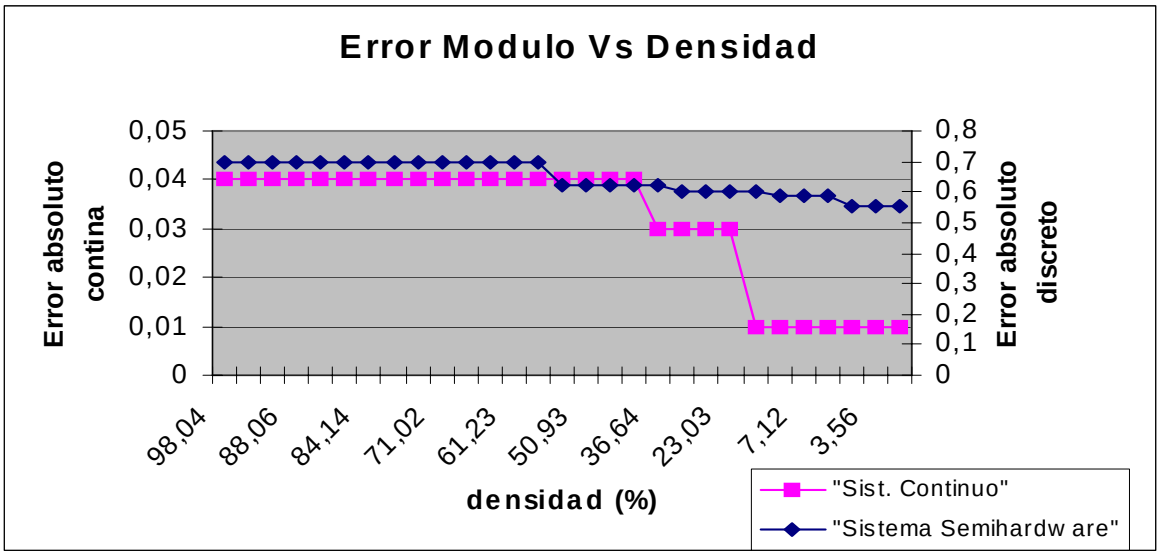


Figura 4.39. Error absoluto para el sistema *software* y *hardware* según la diferencia de módulos.

.

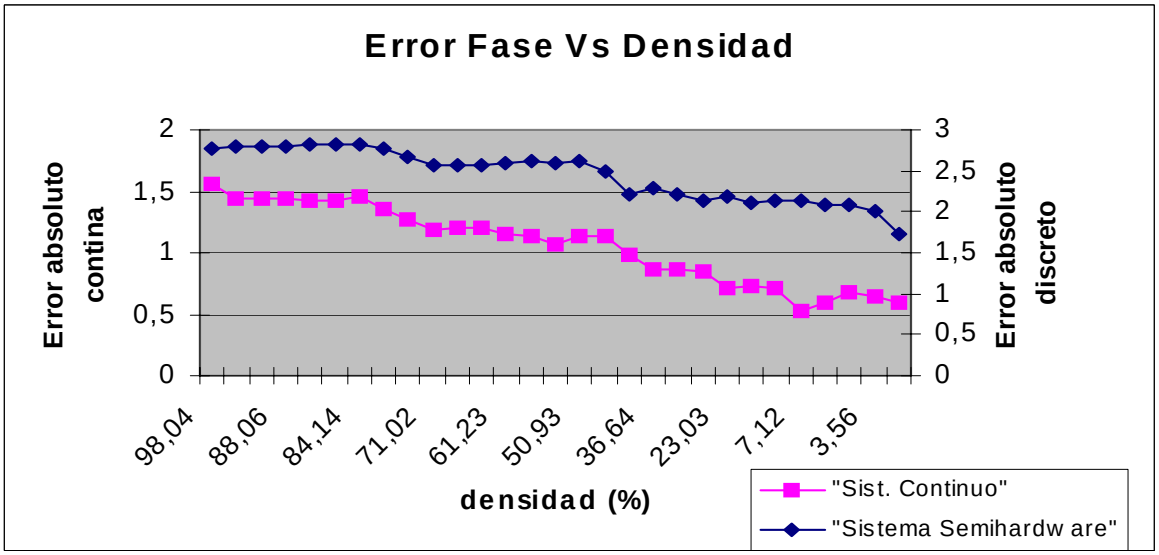


Figura 4.40. Error absoluto para el sistema *software* y *hardware* según la diferencia de fases.

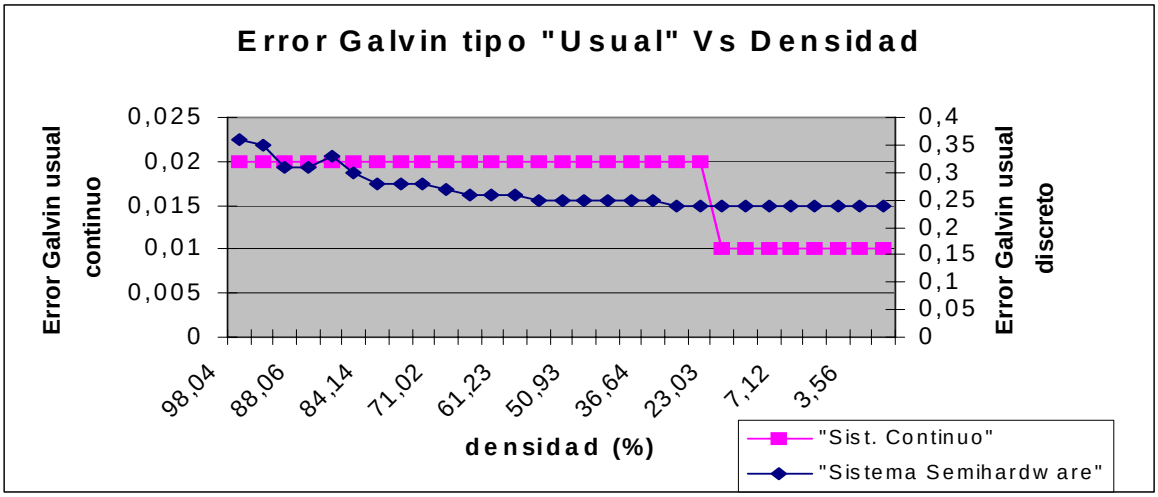


Figura 4.41. Error absoluto para el sistema *software* y *hardware* según la métrica de Galvin.

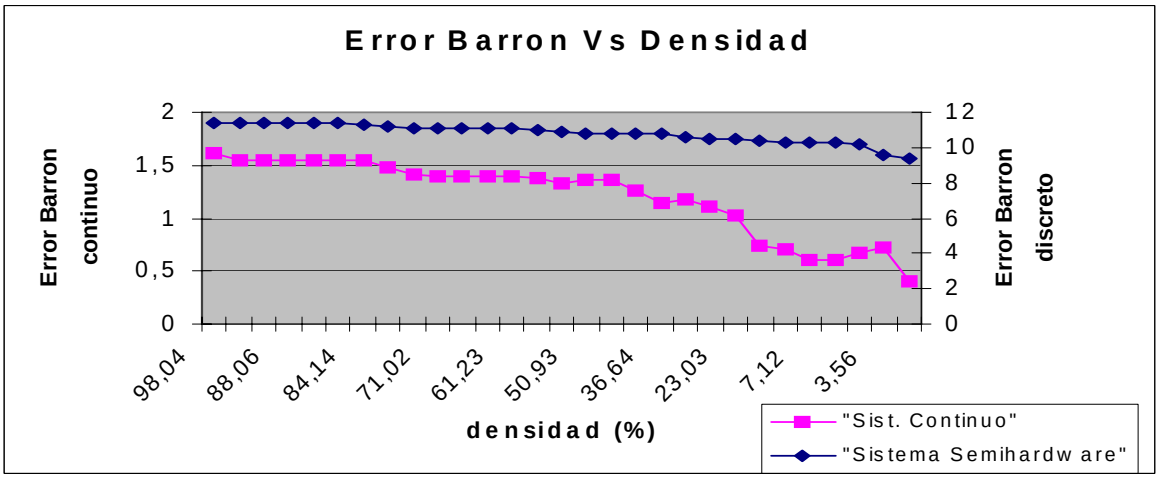


Figura 4.42. Error absoluto para el sistema *software* y *hardware* según la métrica de Barron.

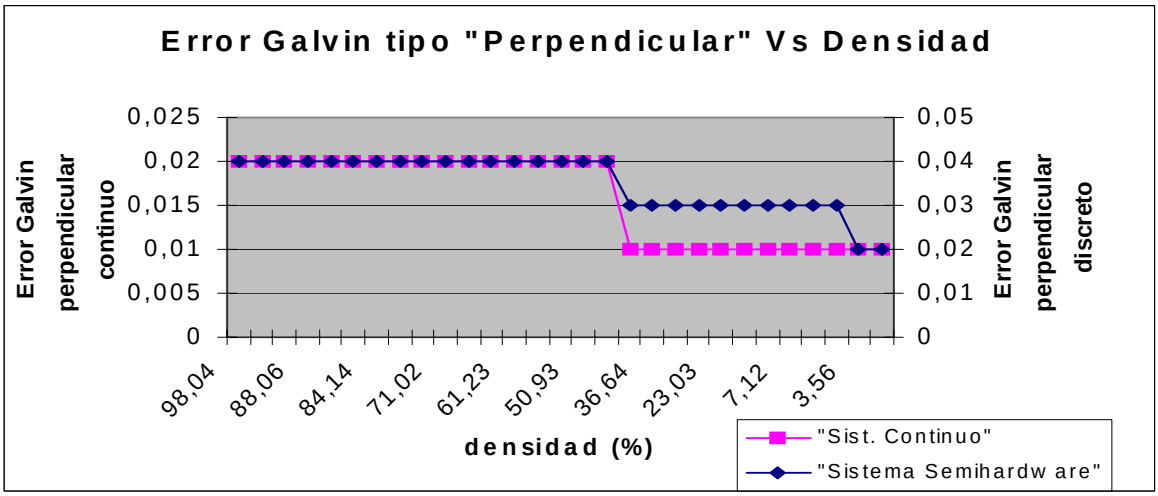


Figura 4.43. Error absoluto para el sistema *software* y *hardware* según la métrica de Galvin perpendicular.

4.6. Conclusiones

Se ha presentado una plataforma que controla el modelo original en todos sus parámetros (biológicos, estructurales, precisión, etc.) y que puede actuar como puente entre *software* y *hardware*, de forma que permite obtener un abanico de parámetros para la posterior implementación en la placa de desarrollo.

Cabe destacar que para la obtención de los resultados anteriores se han realizado más de 17.000 simulaciones. El algoritmo requiere unos 30 segundos de procesamiento por cada fotograma en un Pentium 4 a 3GHz con 1 GB de RAM y ejecución C++ en modo *release*.

Posteriormente, se ha comprobado la calidad de la solución obtenida (la combinación de parámetros candidata a ser implementada) de forma que se pueda estimar el coste en precisión al pasar a punto fijo todas las etapas, basándose en las principales métricas de error consideradas en la literatura especializada.

Con respecto a las medidas de error, se ha realizado un breve análisis del comportamiento de cada métrica utilizada, concluyendo que la métrica usual de diferencia de ángulos (denominada Fase) y la de Barron [BAR96] son las que menos variación sufren para el conjunto de los cinco estímulos considerados. Sin embargo los resultados respecto a las métricas propuestas por Galvin [GAL98] acumulan alteraciones muy diferentes respecto a cada estímulo, esto se puede explicar por la sensibilidad de la métrica Galvin $\perp$ respecto al problema de apertura.

Por último se ha comprobado la bondad de la medida confianza diseñada para filtrar puntos en el camino de datos, este último mecanismo modula la calidad en función del número de puntos cuyo flujo será calculado. Como resultado, se ha verificado una disminución del error independientemente de la métrica, a medida que disminuye la densidad de puntos en el camino de datos, estableciendo las funciones que relacionan la calidad con la densidad de puntos de procesamiento.

Finalizada esta etapa, se tiene una medida de confianza que filtrará toda la ruta de datos, además de un conjunto de valores sobre los cuales se orientará la implementación en *hardware*, aunque es todavía necesario, realizar varias transformaciones al efecto que serán justificadas e implementadas en el siguiente capítulo, debido a las limitaciones relativas a los recursos de la plataforma de prototipado dispuesta.

CAPÍTULO V

Transformación e implementación sobre una plataforma basada en *hardware* reconfigurable

5.1. Introducción

Se presenta el diseño e implementación de arquitecturas específicas para el cálculo del flujo óptico, donde se explican varias modificaciones respecto al modelo original destinadas a su implementación en un chip. La plataforma usada para este trabajo es el modelo *RC1000* de *Celoxica* basado en un chip *Virtex 2000E BG 560-6* con 640Kbits de memoria interna y 8 Mbytes de memoria externa dispuesta en 4 bancos independientes [CEL06] [XIL06]. Se aborda la implementación desde el punto de vista del co-diseño, donde la presente placa actúa como un coprocesador específico, una vez que recibe los datos por el bus PCI, los procesa y reenvía al computador de nuevo. Para ello se ha diseñado *ad hoc* un *micropipeline* (segmentación de cauce basada en paso de mensajes) con una estructura de sincronización basada en el uso de semáforos y canales. Por último se evalúa el sistema con unas medidas del rendimiento, precisión y el coste *hardware* asociado.

5.2. Descripción del sistema utilizado. Fragmentación en etapas del modelo McGM

5.2.1. Introducción. Descripción de cada una de las etapas

A continuación se muestra la arquitectura, transformaciones matemáticas realizadas y estructura de cada una de las etapas implementadas en *hardware*. Se presta especial énfasis al diseño de filtros de forma recursiva y se particulariza al dominio temporal, ya que esto

contribuye a resolver el problema del alto gasto de memoria al convolucionar fotogramas completos a lo largo del tiempo, siendo además, un tópico muy útil en lo referente a implementación de sistemas de procesado de señal en tiempo real.

La unión de todas las etapas involucradas en el circuito total, se hace posible mediante un cauce segmentado asíncrono con diferente número de etapas, (dependiendo este número, del tipo de arquitectura que se va a considerar) donde se explota el paralelismo consustancial a este tipo de plataformas.

5.2.2. Filtrado espacial

Se ha diseñado una célula convolutiva elemental basada en la propiedad de separabilidad de la Gaussiana (*v.* 3.2.2.2), que se replica para implementar el filtro de pre-procesamiento paso baja, y cada una de las pirámides espaciales mostradas en este trabajo sin más que cambiar los coeficientes de los diferentes filtros que intervienen en ellas.

Se realiza el filtrado mediante 2 procesos, siendo el primero una convolución por filas, en esta primera parte los datos correspondientes a la misma fila del fotograma a tratar se van desplazando y multiplicando siguiendo un camino de datos diseñado al efecto. Posteriormente, hay que sumar cada una de las contribuciones contenidas en la longitud del filtro y renormalizar los valores de la adición. Con este primer proceso completado, se tiene un filtrado lineal de los valores del fotograma, aunque con un efecto de bordes inherente a cualquier operación de estas características.

Los datos que van saliendo de este módulo van alimentando a un segundo, donde se desarrolla una convolución por columnas, aunque el avance de la máscara del filtro después de cada convolución, se sigue desarrollando por filas. En este módulo, es necesario usar una *Block* RAM con un controlador de direcciones asociado, que se encarga de indexar cada columna de la RAM y cuando se ha llegado al final, desplazar circularmente los índices de la fila (el último lugar lo ocupa el dato que va llegando), tal como se ilustra en la figura 5.1 para mayor claridad. La operación en esta segunda etapa vuelve a ser la multiplicación, suma de las contribuciones y renormalización de resultados.

Cabe destacar que el diseño es totalmente escalable y tanto el registro correspondiente a la convolución por filas, como la RAM de la convolución por columnas,

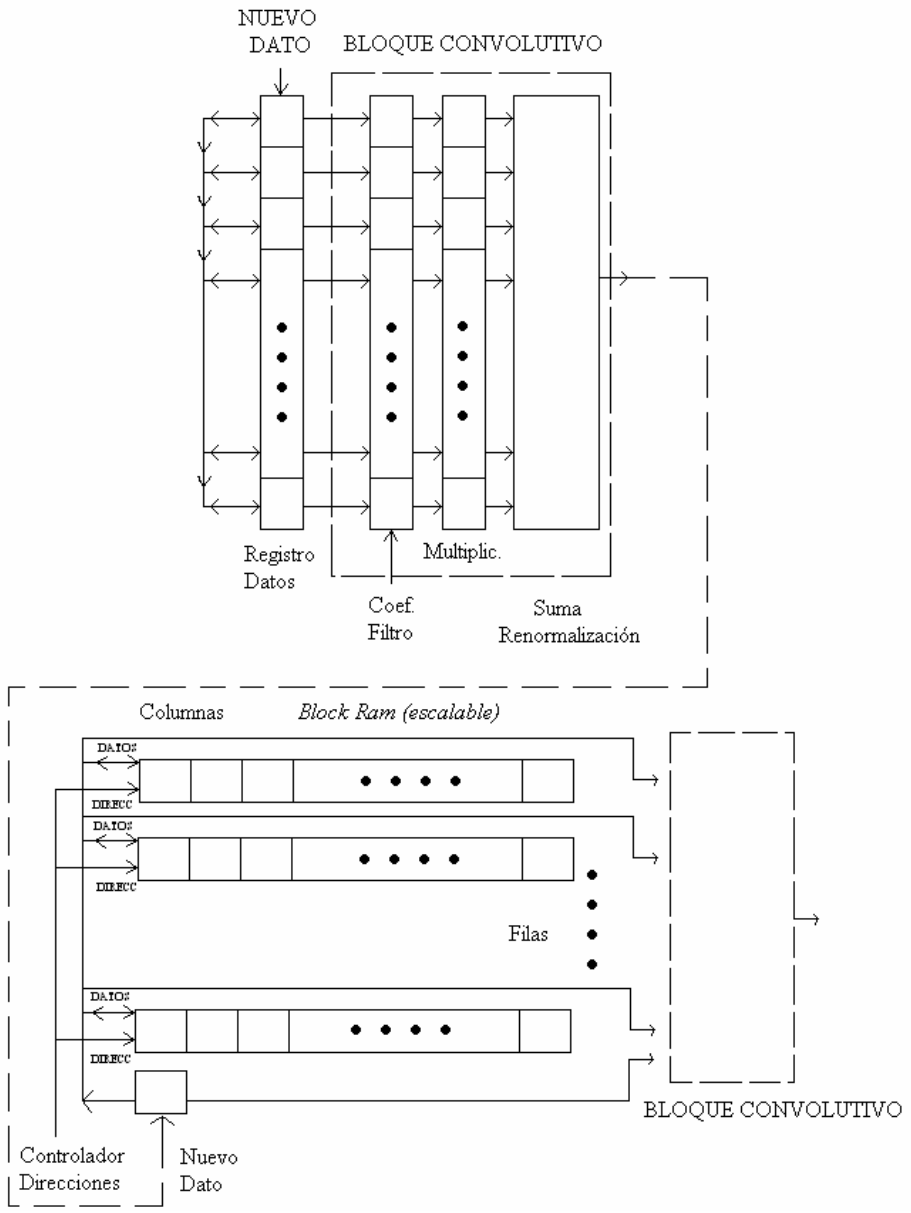


Figura 5.1. Célula elemental convolutiva separable diseñada para este trabajo. (superior) Convolución realizada por filas, (inferior) realización por columnas , donde se usa una *Block Ram* de dimensiones variables. Nótese como el diseño es completamente escalable.

son ajustables a la longitud del filtro y a la anchura del fotograma a tratar, respectivamente. También se puede definir la longitud de palabra de todos estos registros y la precisión con que se realizarán estos cálculos.

Para operar según diferentes filtros, el único cambio a efectuar es modificar los coeficientes de los mismos, según los diferentes operadores derivativos, siendo la presente estructura la que se repetirá para realizar todos los filtros FIR en este trabajo.

Como alternativa se puede ensayar una implementación recursiva que no exija almacenar tantos puntos, esto se usa activamente en la posterior implementación temporal.

Una introducción al problema y un estudio del proceso de diseño se realiza en el siguiente apartado, culminando con la implementación recursiva de los filtros temporales requeridos en este algoritmo.

5.2.3. Diseño IIR frente a la concepción original FIR de forma general. Adaptación a los filtros temporales en este trabajo

Se va a efectuar una breve revisión por los filtros IIR, FIR y su aplicación al problema dinámico del movimiento.

5.2.3.1. Motivación

En la práctica, la convolución temporal podría implementarse multiplicando cada fotograma de la secuencia de entrada por los coeficientes de filtro (mediante un *buffer*), y después sumar todas las contribuciones ponderadas. Hay que tener en cuenta que usando esta estructura,el número de operaciones/punto es proporcional a la extensión temporal del filtro. Cabe destacar adicionalmente, como los sistemas de adquisición de datos (por ejemplo los sistemas de captación de imágenes comerciales) almacenan en memoria posiciones correlativas de la imagen (en términos espaciales). Sin embargo el procesado temporal opera a través de datos no adyacentes entre sí, por lo que este proceso es ineficiente y particularmente mejorable para el procesamiento de señal temporal.

Como ejemplo ilustrativo, se muestra en la tabla 5.I los recursos de memoria externa consumidos al implementar la realización temporal según un esquema FIR (Es necesario almacenar tantos fotogramas como longitud del filtro, como se explicó en 3.2.2.1). Se comprueba, el consumo de recursos nada despreciables, dependiendo de la resolución y precisión de trabajo.

TABLA 5.I Recursos :% *RAM* externa utilizada (Buses Entrada/Salida ocupados) para almacenar una secuencia de imágenes de longitud 23, (longitud por defecto) usando una realización FIR usual en la placa de prototipado descrita al principio.

Resolución	*8 bits*	*10 bits*	*12 bits*
128×96	27%(2/4 buses E/S)	33%(2/4 buses E/S)	40%(2/4 buses E/S)
160×120	42%(2/4 buses E/S	52%(2/4 buses E/S)	63%(4/4 buses E/S)
384×288	>100%	>100%	>100%

A modo de ejemplo, una resolución de 128×96 puntos y 8 bits de precisión, llegan a consumir un 27% de la RAM externa (que corresponde a un uso de 2 de los 4 buses de entrada/salida existentes).

Consecuentemente, se piensa en optimizar esta realización mediante otro tipo de filtros menos costosos desde el punto de vista de la memoria usada. En este trabajo se ha diseñado una forma recursiva de derivada temporal mediante la aproximación de una función más sencilla, cuya transformada z es posible calcular de forma inmediata.

5.2.3.2. Introducción

La ecuación (5.1) modela la convolución de un filtro de respuesta impulsiva finita (FIR), donde *x(n)* es la señal de entrada, *y(n)* la salida y *h(k)* son los coeficientes del filtro. La salida no depende de valores futuros de la entrada (realización causal).

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k) \tag{5.1}$$

Los sistemas IIR (respuesta impulso infinita) vienen descritos por la ecuación (5.2), donde a_k y b_k son los coeficientes del filtro y el valor mayor de M o N es el orden de recursión. Esta descripción incluye una realimentación infinita (teóricamente) de las salidas correspondientes a instantes previos, aplicando en la práctica un umbral. La ventaja de estos sistemas, es que normalmente se pueden implementar con pocos coeficientes (y por consiguiente pocas multiplicaciones), su dificultad sin embargo viene en la estimación de los mismos (a_k y b_k) además de su estabilidad, en este apartado se explicará como adquirir los coeficientes, sin embargo para un análisis detallado de la estabilidad se puede consultar [IFE93].

$$y(n) = \sum_{k=0}^{N} a_k x(n-k) - \sum_{k=0}^{M} b_k y(n-k) \tag{5.2}$$

La derivada de la Gaussiana aparece de forma frecuente en procesamiento de imagen, existiendo varios ejemplos de implementaciones recursivas, cada una con sus peculiaridades,

[VLI98] diseña una implementación recursiva, que requiere $2 \cdot k$ multiplicaciones/puntos para un orden de recursión k, independiente de otros parámetros, como la desviación típica σ, este método proporciona buenos resultados. [DER93] usa métodos numéricos de diferenciación, para aproximar el operador *LOG* (Laplaciana de Gaussiana) a una Gaussiana en el dominio espacial, siendo este método apropiado con respecto a la exactitud aunque difícil de extrapolar a otro tipo de filtros. [JIN97] por otra parte, implementa el operador *LOG* mediante la transformada *z*.

5.2.3.3. Diseño de los filtros recursivos apropiados para este trabajo

La transformada *z* establece una relación entre la entrada y la salida de un sistema en el dominio de la variable compleja *z*. (Se puede considerar esta transformada como una generalización de la transformada de Fourier, una buena introducción se puede encontrar en [IFE93]). La transformada z, para filtros causales viene dada por la ecuación (5.3), consistiendo esta ecuación en una serie de potencias que no siempre converge para todos los valores de *z*, los valores donde *X(z)* se hace infinito se denominan polos, y la región donde *X(z)* converge se llama región de convergencia (*RDC*).

$$X(z) = \sum_{n=0}^{\infty} x(n) z^{-n} \tag{5.3}$$

Si se aplica la transformada z a las expresiones anteriores (5.1-5.2).se obtiene la función de transferencia para cada tipo de filtro, ecuaciones (5.4-5.5). Donde *H(z)* es la transformada de la función de transferencia *h(i)*. Para la derivada de la Gaussiana $G_n(x)$, se puede desarrollar (5.4) en términos de z^{-1}según versa la expresión (5.6).

$$H(z) = \sum_{k=0}^{k=N} h(k) z^{-k} \tag{5.4}$$

$$H(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_N z^{-N}}{1 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_M z^{-M}} = \frac{\sum_{k=0}^{N} a_k z^{-k}}{1 + \sum_{k=1}^{M} b_k z^{-k}} \tag{5.5}$$

$$H(z) = G_n(0) + G_n(1)z^{-1} + G_n(2)z^{-2} + \ldots G_n(N)z^{-N} \tag{5.6}$$

Llegado a este punto, es deseable relacionar los coeficientes a_k ,b_k ,G_n, para ello, se realiza la división polinomial de (5.5) para obtener una serie de potencias de z^{-1}que puede ser comparada directamente con (5.6), permitiendo resolver los coeficientes a_k ,b_k .Dividiendo iterativamente, es posible generalizar la relación:

$$G_0 = \frac{a_0}{b_0}$$

$$G_1 = \frac{1}{b_0}\left(G_0 - \tfrac{a_0}{b_0} b_1\right)$$

$$G_2 = \frac{1}{b_0}\left(a_2 - \tfrac{a_0}{b_0} b_2 - \tfrac{b_1}{b_0}\left(a_1 - \tfrac{a_0}{b_0} b_1\right)\right)$$

$$G_3 = \frac{1}{b_0}\left(a_3 - \tfrac{a_0}{b_0} b_3 - \tfrac{b_3}{b_0}\left(a_1 - \tfrac{a_0}{b_0} b_1\right) - \tfrac{b_3}{b_0}\left(a_2 - \tfrac{a_0}{b_0} b_2 - \tfrac{b_1}{b_0}\left(a_1 - \tfrac{a_0}{b_0} b_1\right)\right)\right)$$

$$G_k = \frac{1}{b_0}\left[a_k - \sum_{i=1}^{k} G_{k-1} b_i\right]; G_0 = \frac{a_0}{b_0} \tag{5.7}$$

aunque sigue existiendo un problema de dependencia de los coeficientes G_k con los b_i, para solventar esta dificultad se intenta simplificar la función de transferencia IIR (5.5), tal y como señala [JIN97] mediante la expresión (5.8). Para sistemas estables, el valor de p se mueve en un entorno controlado ($0<p<1$), donde la principal ventaja de esta transformación es que el denominador es sólo función de una variable:

$$H(z) = \frac{\sum_{k=0}^{N} a_k z^{-k}}{(1 - pz^{-1})^N} \tag{5.8}$$

Se puede separar la derivada de la Gaussiana en 2 secuencias causales $H^+(z)$ y $H^-(z)$ donde una secuencia se aplica en la dirección positiva y otra en la negativa. Siendo $k_s = \frac{1}{\sqrt{2\pi\sigma}}$

$$H(z) = H^+(z) + H^-(z)$$

$$H^+(z) = \frac{Y_1(z)}{X(z)} = k_s \frac{1 + a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_N z^{-N}}{1 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_M z^{-M}} \tag{5.9}$$

$$H^-(z) = \frac{Y_2(z)}{X(z)} = k_s \frac{c_0 + c_1 z^1 + c_2 z^2 + \ldots + c_N z^N}{1 + b_1 z^1 + b_2 z^2 + \ldots + b_M z^M}$$

Si se igualan (5.8) y (5.9), identificando potencias de z^{-1}, se pueden expresar los denominadores en función del parámetro *p*, determinándose su valor por un simple ajuste iterativo. El resultado es un filtro recursivo según (5.10):

$$Y^+(n) = \left[a_0 X(n) + a_1 X(n-1) + a_2 X(n-2)\right] - \left[b_1 Y^+(n-1) + b_2 Y^+(n-2)\right]$$

$$Y^-(n) = \left[a_3 X(n+1) + a_4 X(n+2)\right] - \left[b_1 Y^-(n+1) + b_2 Y^-(n+2)\right] \tag{5.10}$$

$$Y_x(n) = Y^+(n) + Y^-(n)$$

donde a_k y b_k son los coeficientes del filtro. *X(n)* es la imagen de entrada, $Y^+(n)$ es la salida intermedia de la derivada que se aplica en la dirección positiva ($Y^-(n)$ negativa). $Y_x(n)$ es la respuesta de la derivada de Gaussiana, el filtro sólo requiere nueve multiplicaciones/punto independientemente del tamaño σ, recuérdese que la implementación anterior exigía 23 multiplicaciones /punto.

Se demuestra que cada orden adicional de diferenciación requiere al menos un orden más de recursión (2 multiplicaciones más) para hacer los errores tolerables. Los errores se incrementan firmemente a medida que crece el orden, siendo las derivadas de órdenes elevados, computacionalmente ineficientes comparadas con la convolución directa.

A continuación se procede a particularizar la forma del tipo de filtros explicados en este apartado para dotarla de una naturaleza recursiva y propia.

5.2.3.4. Forma recursiva de los filtros temporales

En este trabajo se ha diseñado una forma recursiva de derivada temporal mediante la aproximación de una función más sencilla (5.11) cuya transformada *z* es posible calcular de forma inmediata.

$$F_n(t) = \frac{1}{\alpha}\left(\frac{t}{\alpha}\right)^n e^{-\frac{\beta t}{\alpha}} \tag{5.11}$$

Donde α es la escala temporal usada en la *Gaussiana en el logaritmo del tiempo* (*Gelt*), (véase ecuación 3.9) *n* y β son parámetros cuya función es optimizar el ajuste entre las 2 funciones.

Eligiendo cuidadosamente los parámetros, se puede imponer que ambas derivadas pasen por cero para el mismo valor de t. Para la función *Gelt* esto ocurre cuando $t/\alpha = 1$, la función de aproximación $F_n(t)$ coincide con su pico cuando $t = \alpha n/\beta$. Por consiguiente, las dos funciones pueden hacer coincidir su pico al mismo tiempo si. $\beta = n$ .El valor de *n* toma un papel similar al de τ in la función *Gelt*.

También se busca que $F_n(t)$ tenga su pico con el mismo valor de amplitud que *Gelt*, operando, se llega a que estas 2 condiciones se cumplen según (5.12):

$$\sqrt{\pi}\tau\, \mathrm{e}^{\left(\tau^2/4\right)} = \left(\frac{\beta}{n}\right)^n \mathrm{e}^{(n)} \tag{5.12}$$

Para cada valor de τ se puede elegir *n* de forma que la aproximación es de ajuste óptimo e independiente de α. Este resultado es importante puesto que α controla la anchura de la derivada y la posición de su pico. La ecuación (5.12) es trascendente y no tiene una solución analítica, se va a resolver mediante métodos numéricos (bisección).

La forma recursiva de $F_n(t)$ requiere que n sea entero, esta premisa limita los valores de τ para los que se pueden generar una aproximación óptima, como contrapartida se puede cambiar el tamaño de la derivada, sin más que cambiar α, obviando una etapa de diseño.

Se demuestra en esta sección que la forma recursiva de $F_n(t)$ es de orden n+2 y por ende para máxima eficiencia se busca un valor bajo de *n*.

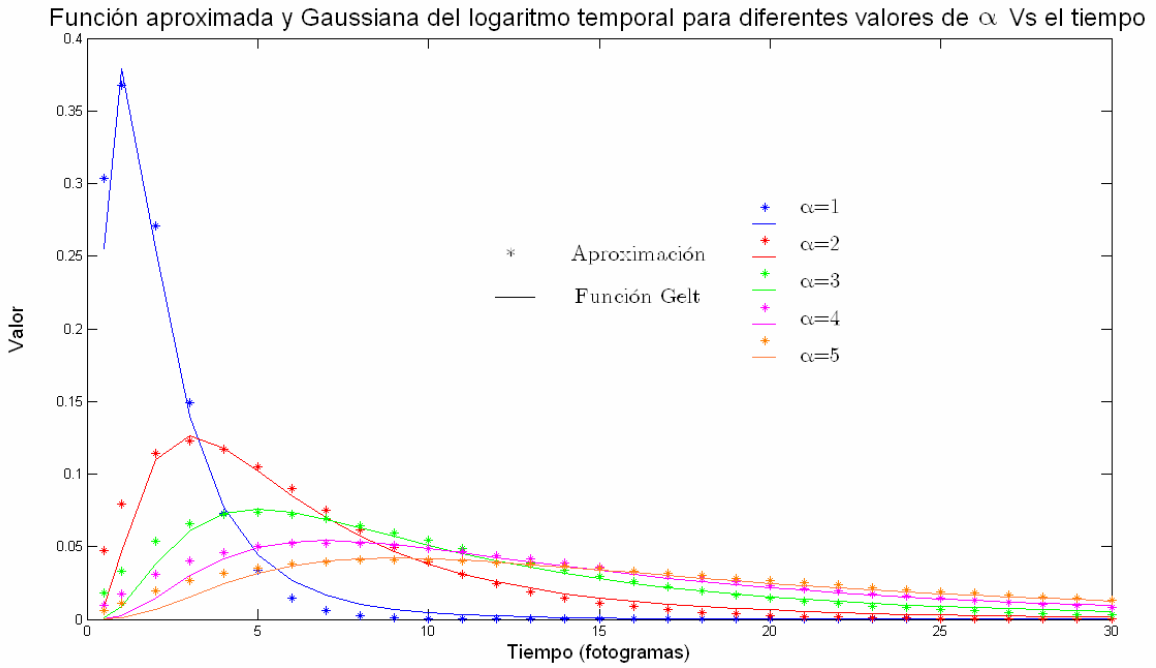


Figura 5.2. Diferentes funciones Gaussianas en el logaritmo temporal y sus funciones aproximadas. Nótese como la aproximación es satisfactoria independientemente del valor de α variable.

Si la elección se hace con n=1, se encuentra, como muestra la tabla 5.II, que esta función se correlaciona de forma máxima con la función *Gelt*, (para τ=1.12), el cual tiene a su vez una anchura entre 4 y 10 fotogramas dependiendo de α. Una comparación de *Gelt* y la función aproximada $F_n(t)$ puede verse en la figura 5.2, mostrándose que la función $F_n(t)$ se ajusta bien a *Gelt* para un rango extenso de α.

Se va a proceder a la implementación de forma recursiva de esta función, para ello se aplicarán 3 procedimientos:

- Se realiza la transformada z de la función FIR $F_n(t)$
- Se iguala esta expresión a la forma canónica de un filtro IIR
- Se obtienen a_k y b_k gracias a una comparación de potencias de z.

TABLA 5.II Solución de la ecuación (5.12) para diferentes valores de n y τ.

n	1	2	3	4	5	6	7
τ	1.12	1.82	2.46	3.04	3.55	5.55	19.64

$$H(z)=\sum_{k=0}^{\infty}F_1(k)z^{-k}=\sum_{k=0}^{\infty}\frac{1}{\alpha^2}ke^{-\frac{k}{\alpha}}z^{-k}=\frac{1}{\alpha^2}\left[z^{-1}e^{-\frac{1}{\alpha}}+\ldots+kz^{-k}e^{-\frac{k}{\alpha}}\right] \tag{5.13}$$

Se procede a transformar la expresión anterior en una fácilmente identificable con la forma canónica de los filtros IIR.

$$\begin{gathered}S_n=\sum_{k=1}^{n}kx^k \\ x=z^{-1}e^{-\frac{1}{\alpha}}\end{gathered} \tag{5.14}$$

Si se realizan las transformaciones siguientes:

$$xS_n=x^2+2x^3+3x^4\ldots nx^{n+1} \tag{5.15}$$

se resta la expresión anterior a S_n:

$$S_n\left(1-x\right)=x+x^2+x^3+x^4\ldots x^n-nx^{n+1} \tag{5.16}$$

y posteriormente se multiplica por *(1-x)*:

$$S_n\left(1-x\right)^2=x+x^n-(n+1)x^{n+1}+nx^{n+2} \tag{5.17}$$

cuando se cumple que $\left|x\right|<1$, como simultáneamente se cumple también que $n\to\infty$, la expresión degenera en (5.18):

$$S_\infty=\frac{x}{\left(1-x\right)^2} \tag{5.18}$$

Sustituyendo $x=z^{-1}e^{-\frac{1}{\alpha}}$ se llega a la expresión general de la transformada *z* de la función aproximada a la función *Gelt*.

$$H(z)=\frac{z^{-1}e^{-\frac{1}{\alpha}}}{\left(1-z^{-1}e^{-\frac{1}{\alpha}}\right)^2}=\frac{ze^{-\frac{1}{\alpha}}}{\left(z-e^{-\frac{1}{\alpha}}\right)^2} \tag{5.19}$$

Si $\gamma={}^{1}/_{\alpha}$ y se iguala a la expresión general de la forma canónica de un filtro IIR:

$$H(z)=\frac{\sum_{k=0}^{N}a_k z^{-k}}{1+\sum_{k=1}^{M}b_k z^{-k}}=\frac{1}{\alpha^2}\frac{ze^{-\gamma}}{(z-e^{-\gamma})^2} \tag{5.20}$$

donde será posible determinar los coeficientes b_k, a_k.:

$$\frac{a_0+a_1z^{-1}+a_2z^{-2}+\ldots+a_Nz^{-N}}{1+b_1z^{-1}+b_2z^{-2}+\ \ldots\ +b_Mz^{-M}}=\frac{1}{\alpha^2}\frac{e^{-\gamma}z^{-1}}{1-2e^{-\gamma}z^{-1}+e^{-2\gamma}z^{-2}} \tag{5.21}$$

deduciéndose a partir de aquí, que:

$$a_0=0\ ;\ a_1=\frac{1}{\alpha^2}e^{-\gamma}\ ;\ b_1=-2e^{-2\gamma}\ ;\ b_2=e^{-2\gamma} \tag{5.22}$$

En definitiva, aplicando la transformada inversa y los coeficientes obtenidos, se llega a la implementación recursiva temporal deseada.

$$y(n)=\frac{1}{\alpha^2}x(n-1)e^{-\gamma}+2e^{-\gamma}y(n-1)-e^{-2\gamma}y(n-2) \tag{5.23}$$

Esta expresión denota que la función aproximada puede ser generada a partir de la imagen de entrada y 2 fotogramas previos de salida. Se necesitan 3 multiplicaciones y 2 sumas, si se compara con la convolución temporal usual (ésta requiere el mismo número de multiplicaciones y sumas que el tamaño de la ventana, 23 fotogramas por defecto) este método incrementa el rendimiento en un factor 7.

5.2.3.5. Implementación *hardware* de los filtros temporales diseñados al efecto

A continuación se procede a mostrar como se implementa físicamente esta estructura, según se indica en la figura 5.3. Se observa un triple *buffer* circular donde se almacenan cada uno de los fotogramas según van llegando. En el instante, *t=n* (usando esta notación), se utilizan los fotogramas correspondientes a *t=n-1, n-2* cuyo resultado pasa a la posición alta.

En el siguiente instante *t=n+1*, se sobrescribe la posición ocupada por el fotograma en el instante *t=n-3* (posición baja), por último, en el instante *t=n+2*, se sobrescribe la posición alta. Este proceso se vuelve a repetir cada 3 fotogramas consecutivos, mostrándose la existencia de entradas para cada uno de los pesos ponderados que intervienen en el proceso de cálculo .

Posteriormente, es necesario implementar un módulo que se encargue de formar las derivadas temporales, (*v.* 3.2.2.1.) puesto que el diseñado anteriormente sólo crea la derivada de orden cero, para aprovechar recursos de la estructura presente diseñada, se idea una derivación numérica basada en simples operadores de gradiente, con una longitud asociada de dos y tres operaciones respectivamente.

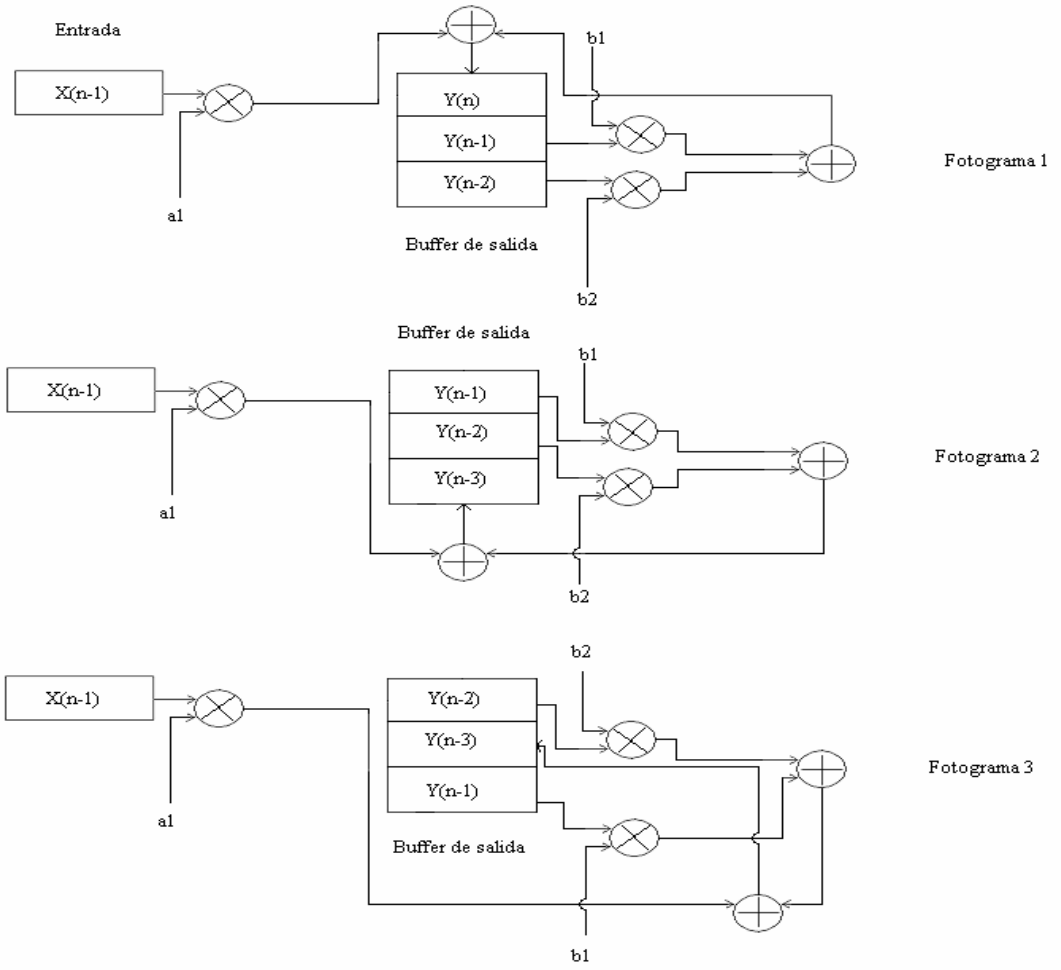


Figura 5.3. Implementación del filtro *IIR* y sus derivadas, mediante un triple buffer circular.

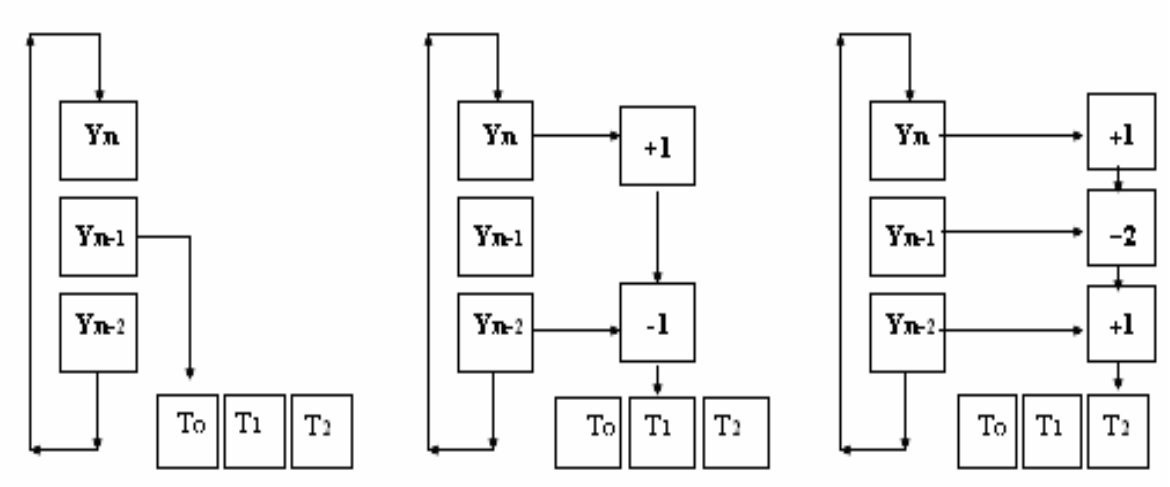


Figura 5.4. Obtención de las derivadas temporales, gracias al triple buffer recursivo. (Izquierda) , se tiene la derivada de orden cero. (Centro) primera derivada al aplicar el operador gradiente. (Derecha), Segunda derivada al aplicar el operador Laplaciano.

Es decir, se fabrica el operador gradiente de primer orden (+1,-1), volviendo a repetir esta derivada sobre sí misma, se obtiene (+1,-2,+1), donde se van almacenando en *T0*, *T1* y *T2* los resultados de la derivada cero, primera y segunda.

En la figura 5.4, se esquematiza todo este proceso descrito, que relaciona el triplete de salidas con las 3 derivadas temporales. Es posible implementar esta derivada introduciendo operadores más complejos [BUR07]. El problema de aproximar la derivada de esta forma sencilla es que se introducen errores acumulativos a medida que se iteran para conformar derivadas de órdenes superiores.

Otra consecuencia, es que a medida que las frecuencias crecen, sus errores se incrementan. [SPI71] hizo un estudio de la propagación del error diferencial cuadrático medio para una retícula de senos, concluyendo que para una longitud constante, los operadores tienen errores asociados que resultan inversamente proporcionales a la frecuencia de la retícula.

Otro aspecto importante a resaltar, es el efecto de la etapa 5.2.2 (filtro paso baja), considerando que inhibe el error de los operadores discretos en el cálculo del flujo óptico cuando se usan modelos de gradiente [CHR98].

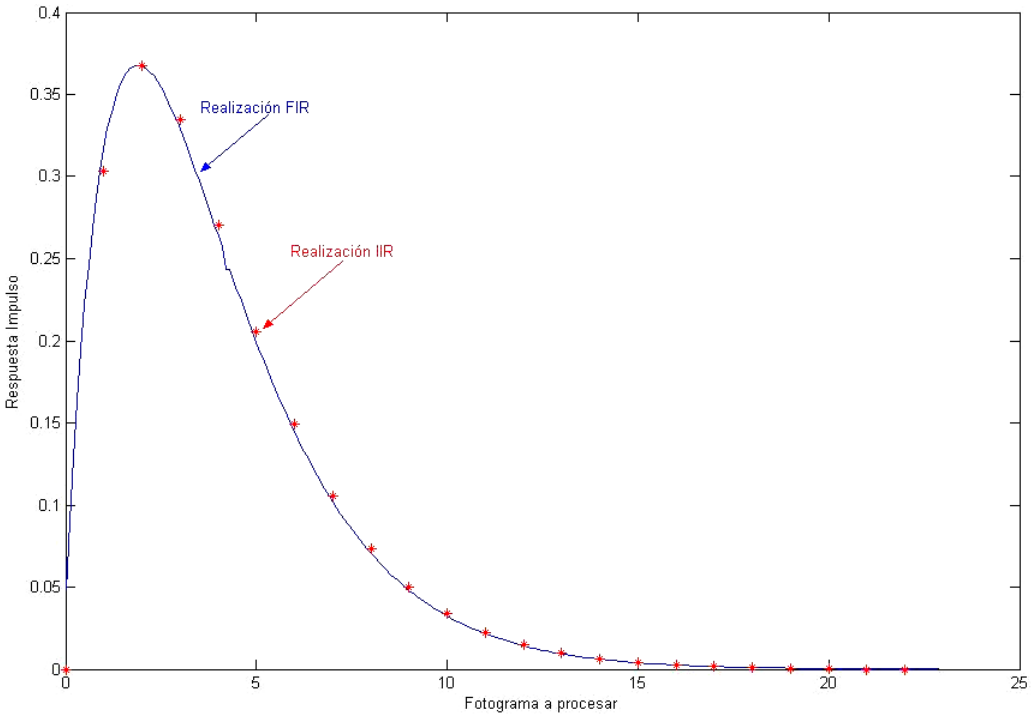


Figura 5.5. Respuesta a impulso de la realización FIR e *IIR* de los filtros correspondientes a la funión *Gelt* y la función aproximada. Nótese que el diseño de la realización *IIR* es estable.

En la figura 5.4 se representa el mecanismo implementado para la obtención de las derivadas temporales. En la figura 5.5 en cambio, se muestran sendas respuestas impulso, correspondientes a la función *Gelt* realizada a partir de un filtro FIR y la función aproximada según el método implementado. Como es posible apreciar, una vez que el filtro se estabiliza , los errores se hacen muy pequeños.

5.2.4. Pirámides de filtros espaciales

En este apartado se va a construir la estructura piramidal espacial, consistente en los diferentes filtros que se usarán en el diseño.

Es conveniente cuantificar previamente el número de términos que intervendrán en la pirámide, mostrando en la tabla siguiente su cálculo. Este número se puede modelar mediante la ecuación recursiva $T[n]=T[n\text{-}1]+n+1$, ($T[0]=1$).

TABLA 5.III Número de celdas en la pirámide según el orden de diferenciación.

Orden x	0	1	2	3	4
Términos	1	3	6	10	15

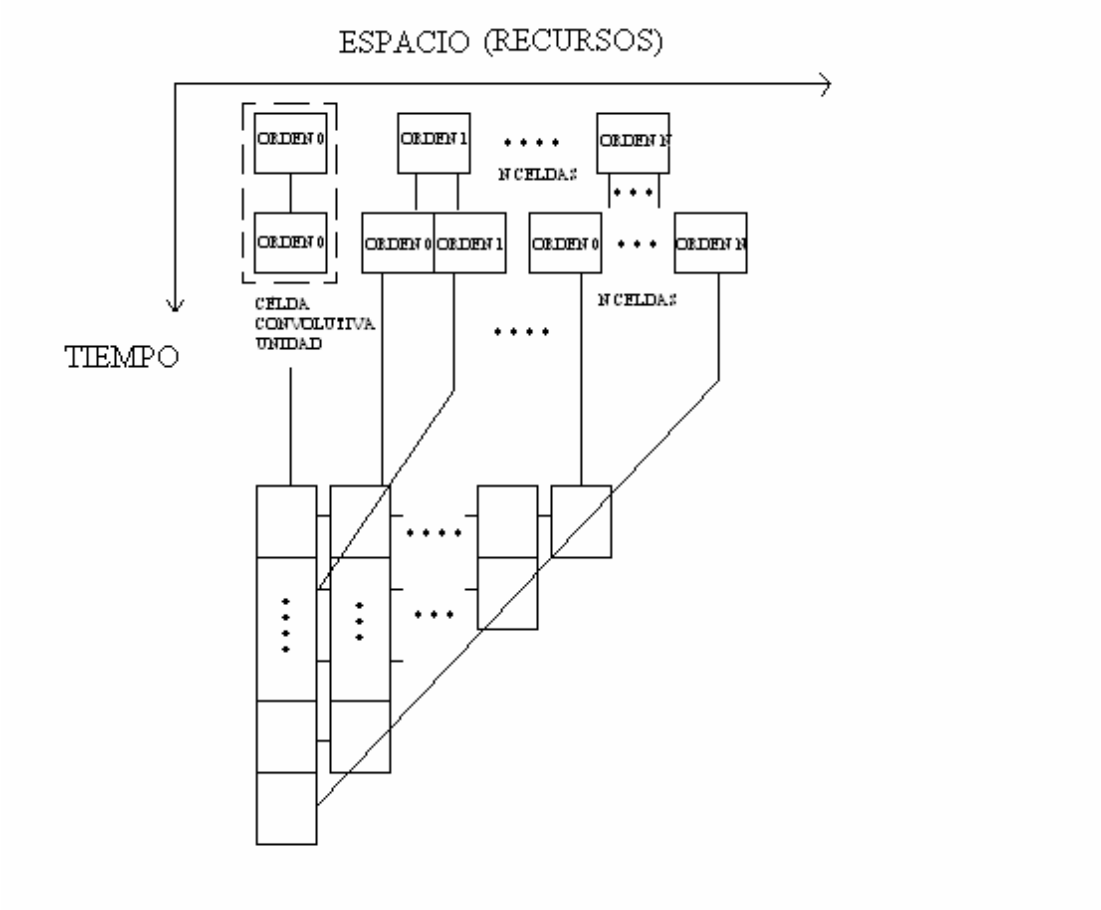


Figura 5.6. (Superior) Esquema de cálculo de las diferentes componentes de la pirámide, donde aparecen tantos términos en *x* como celdas se calculan por filas y como celdas aparecen en la convolución de mayor orden por columnas. (Inferior).

A continuación, se presenta el esquema de implementación para las derivadas espaciales. Para ello se toma como referencia la construcción de la figura 5.1, que a su vez se irá replicando en la estructura presente, según se muestra en la figura 5.6. Dentro del área punteada, se aprecian dos bloques conceptuales, siendo el superior (inferior) representativo de la convolución por filas (columnas). Se construye pues, una estructura, donde existen tantas celdas por filas como celdas tiene la última columna, esto se hace para confeccionar la pirámide de la parte inferior de la figura (obsérvese que si se avanza por frentes diagonales dentro de cada pirámide, se mantiene constante el orden de diferenciación).

Por otra parte, el diseño es totalmente escalable, respetando la jerarquía temporal de la célula convolutiva básica (primero se realizan las convoluciones por filas y después por columnas). El eje de espacio representa los recursos consumidos, RAMs, *slices* y el del tiempo da una idea del grado de paralelismo implementado en este módulo específico de cálculo. En este caso se calculan en paralelo *N* celdas (convolución por filas)y en el siguiente

evento (no necesariamente siguiente ciclo, aunque usualmente será así) se calcula la convolución por columnas mediante 1+2+…+*N* celdas adicionales.

5.2.5. Orientaciones. Etapa de *Steering*

El esquema de los filtros con múltiples orientaciones (*Steering*), se representa en la figura 5.7, volviéndose a usar el diagrama bidimensional con sendos ejes (tiempo y recursos), donde se expone la manera de implementar estos módulos de cálculo. Por cada una de las pirámides anteriores, se construyen otras tres estructuras que representarán los tres primeros órdenes ortogonales. Estas pirámides se diseñan de forma asimétrica, debido al truncamiento de componentes no necesarias en la estructura de cálculo para su consiguiente ahorro en recursos. Haciendo uso de la naturaleza paralela de los cálculos a realizar, se repite esta organización para cada conjunto de orientaciones (este conjunto por defecto es monocomponente y consta de sólo una orientación, aunque esto no tiene porqué cumplirse siempre, como se verá posteriormente) realizando el cálculo de las tres derivadas ortogonales según se explica en (3.2.2.4.), (usando una *Block* RAM cuyo número de términos se justifican en la tabla 5.IV).

En la figura 5.7, aparece una estructura modular donde se calcula cada uno de los

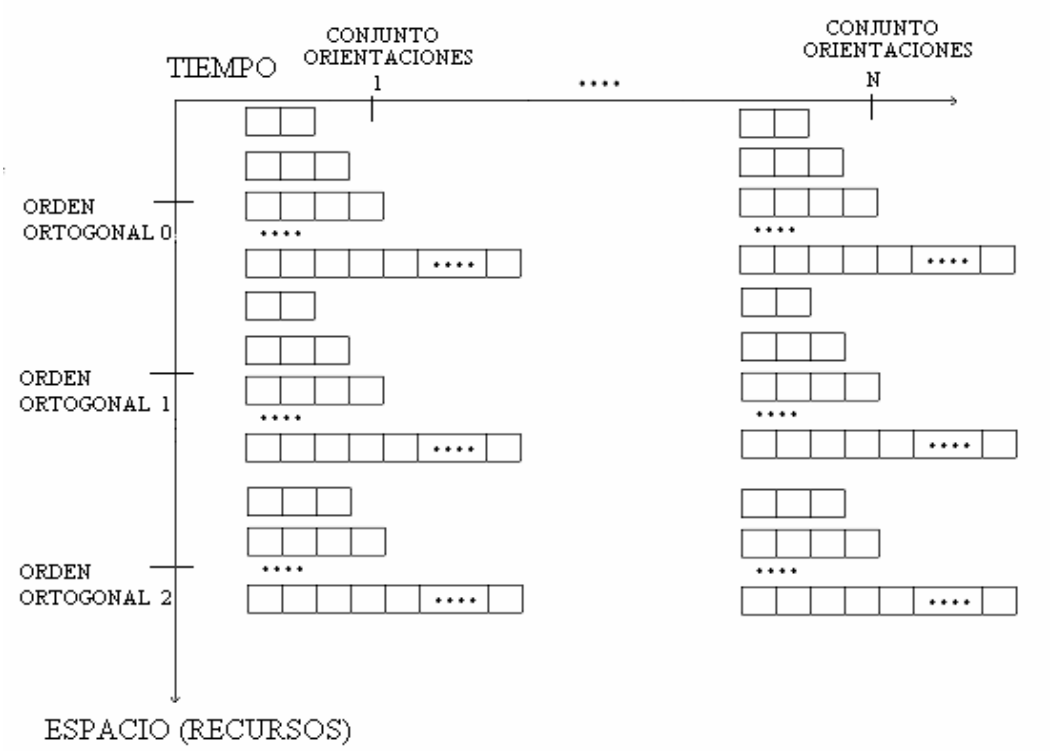


Figura 5.7. Esquema de diseño de la etapa de orientación, donde se elaboran las 3 derivadas ortogonales de forma paralela, se repite este proceso para cada ángulo. Nótese como el diseño es completamente escalable.

TABLA 5.IV Número de términos de orientación en función del orden de diferenciación x. Oo0, Oo1, Oo2 son los órdenes ortogonales, cero, primero y segundo, respectivamente, este número de términos coincide con el número de columnas de la *Block RAM* para guardar los coeficientes

Orden x	*Oo0*	*Oo1*	*Oo2*	*Términos*
1	2	2	0	4
2	3	3	3	9
3	4	4	4	12
n	*n*+1	*n*+1	*n*+1	3·(*n*+1)

términos necesarios para formar este volumen de datos, con una prioridad temporal en el cálculo de todos los órdenes ortogonales correspondientes a cada conjunto de orientaciones frente a los siguientes.

5.2.6. Desarrollo de Taylor y sus derivadas

En este apartado se pretende realizar el desarrollo de Taylor en 3 dimensiones, escalando el diseño al número de términos de la serie. En la figura 5.8, se han representado por colores unas estructuras que indican el número de términos relacionado con el orden máximo de la serie de Taylor (que coincide con el orden de derivación total, menos uno).

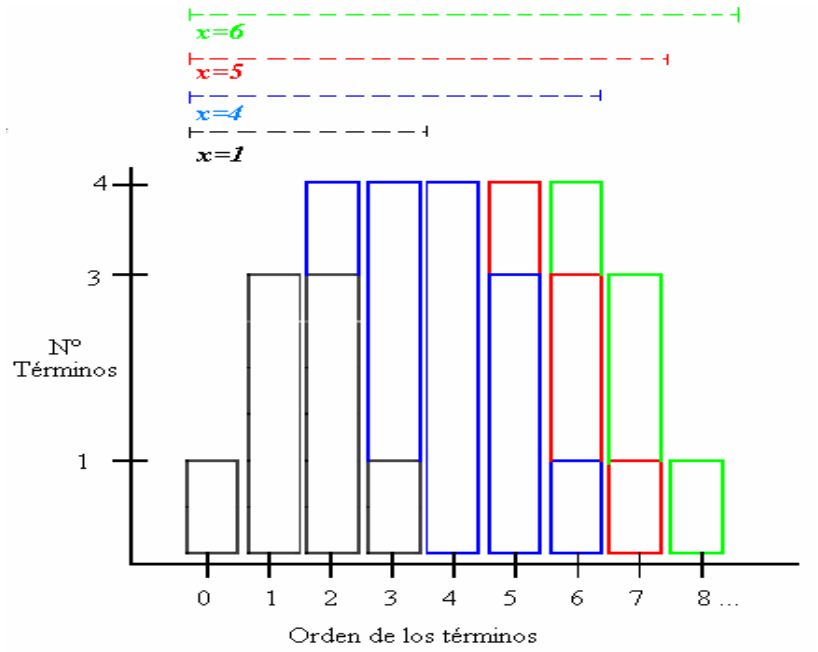


Figura 5.8. Número de términos para cada orden de la serie de Taylor. Representación con distintos colores para resaltar el incremento al añadir un orden adicional.(representado por x).Nótese que la estructura azul contiene a la negra, a su vez la roja contiene a la azul, y la verde a la roja, respectivamente.

Si se observa la estructura de colores de izquierda a derecha, se puede notar que cada una está autocontenida en su vecina inmediata. El número de términos es modular, escalable, y el optar por uno u otro valor depende de la precisión y exactitud que se requiera, además de los recursos de que se disponga. Por otra parte, como se ha indicado previamente en esta memoria, las derivadas de la Gaussiana no forman un conjunto netamente ortogonal, estando los órdenes superiores cada vez más correlacionados entre sí, aportando sólo información redundante.

En la figura 5.9 se esquematiza la forma en que se van construyendo las derivadas del desarrollo de Taylor *X, Y, T* con el número de términos variable desde 8 (correspondiente a un orden primero en *x*, orden segundo de derivada) hasta un número de términos que vienen dado por *4(l+1)*, siendo *l* el orden del desarrollo de Taylor en *x*.

Se calculan todas las derivadas *X, Y, T* de forma paralela, repitiéndose este proceso para cada orientación o conjunto de ellas, usándose registros para almacenar toda esta información. Nótese como los pesos del desarrollo no se anexan en esta etapa, esto es un detalle importante, que se justificará en el siguiente apartado.

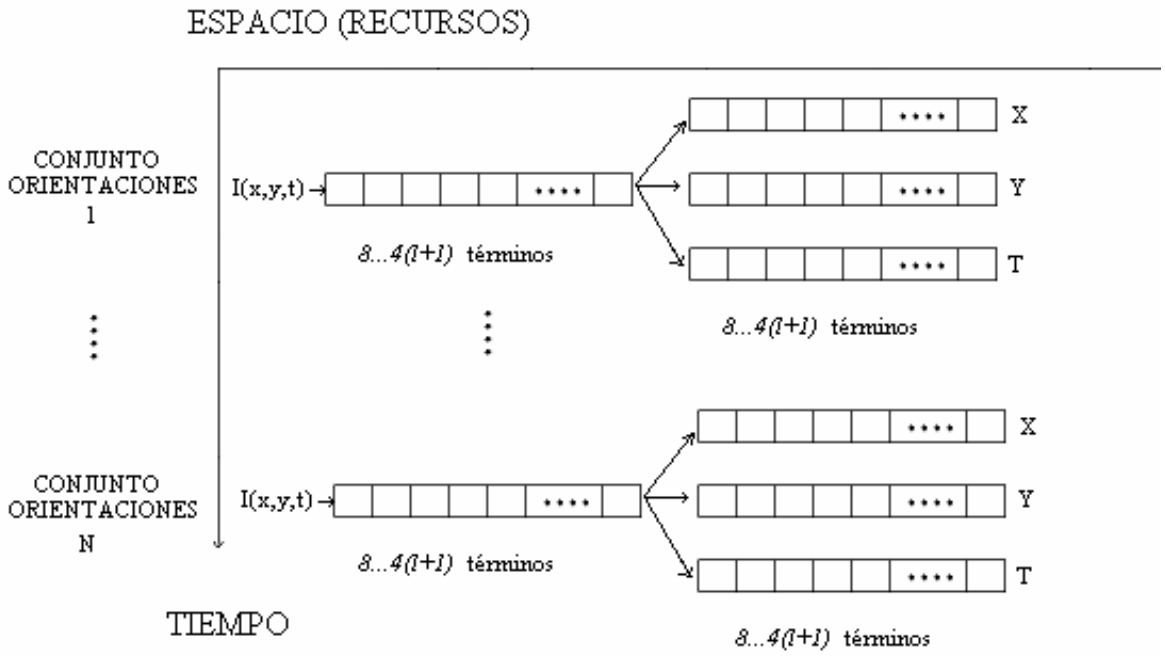


Figura 5.9 Estructura donde se calculan las derivadas de la serie de Taylor.
Para cada conjunto de orientaciones se vuelve a iterar la operación.

5.2.7. Productos de Taylor

En este apartado se trata de elaborar una estructura capaz de realizar los productos internos y su integral discreta a lo largo de la superficie elegida.

Según se indicó en (3.2.4), se escoge una zona simétrica de integración, tomando como origen el cero, esto conlleva (apéndice II y capítulo III) anular todos los términos cruzados no cuadráticos que aparecen en los productos de las derivadas de la serie de Taylor. Ahorrando en espacio y recursos, se aplica esta propiedad mediante un registro con $4\cdot(l+1)$ componentes donde se almacenarán sólo las contribuciones precalculadas que intervienen en la operación. (esta es la razón por la que se utilizan los pesos de la serie de Taylor sólo en este último paso, no siendo así en los modelos previos de *software* implementados).

Los productos de cada sexteto *XX,XY,XT,YY,YT,TT* se calculan en paralelo, empleándose tantas adiciones intermedias como términos hay menos uno, según se esquematiza en las figuras 5.10 y 5.11. Se han ensayado dos diseños, ambos modulares, en el primer diseño se almacena este sexteto mediante *Block* RAM*s* asociadas a la FPGA y en el segundo diseño se usan sólo puertas lógicas (*slices*). La opción de utilizar memoria distribuida (slices) mejora las prestaciones porque permite un mayor grado de paralelismo, aunque como contrapartida los recursos consumidos son mayores.

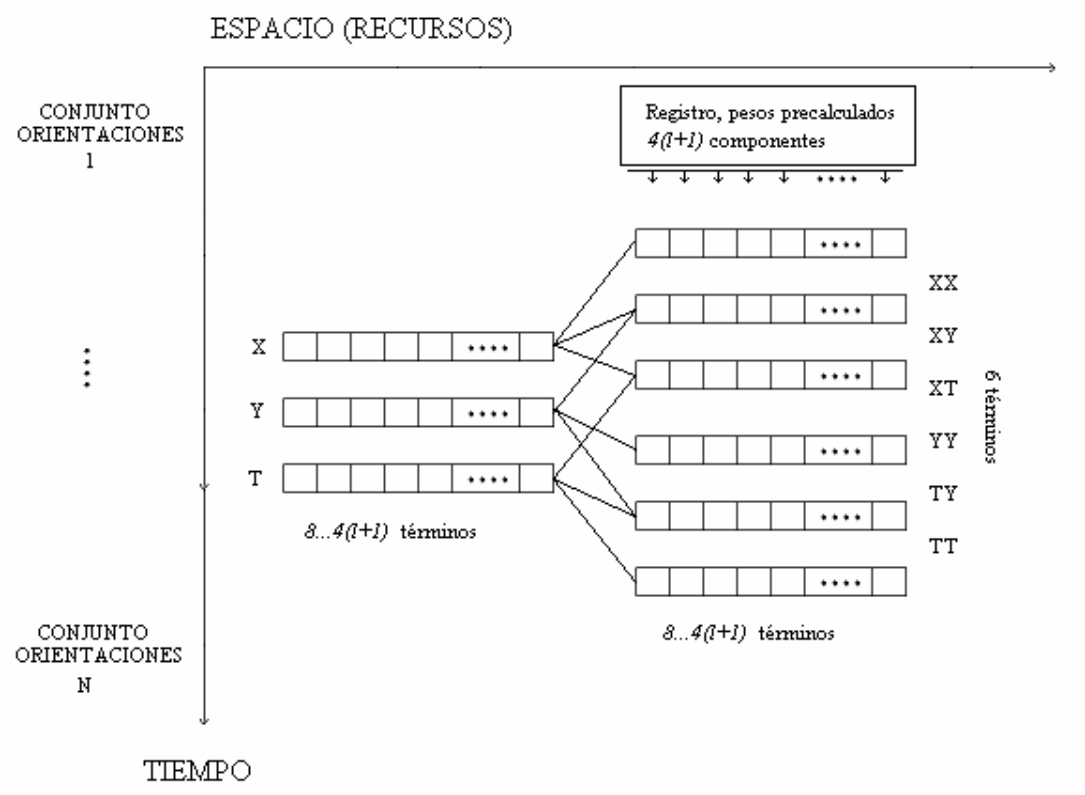


Figura 5.10. Fabricación de los productos de Taylor (coincide con el primer paso para desarrollar la integral discreta), heredándose la modularidad de los componentes.

Se ha optado por esta última solución a pesar del gasto que conlleva, priorizando reducir el número de ciclos total que consume esta etapa.

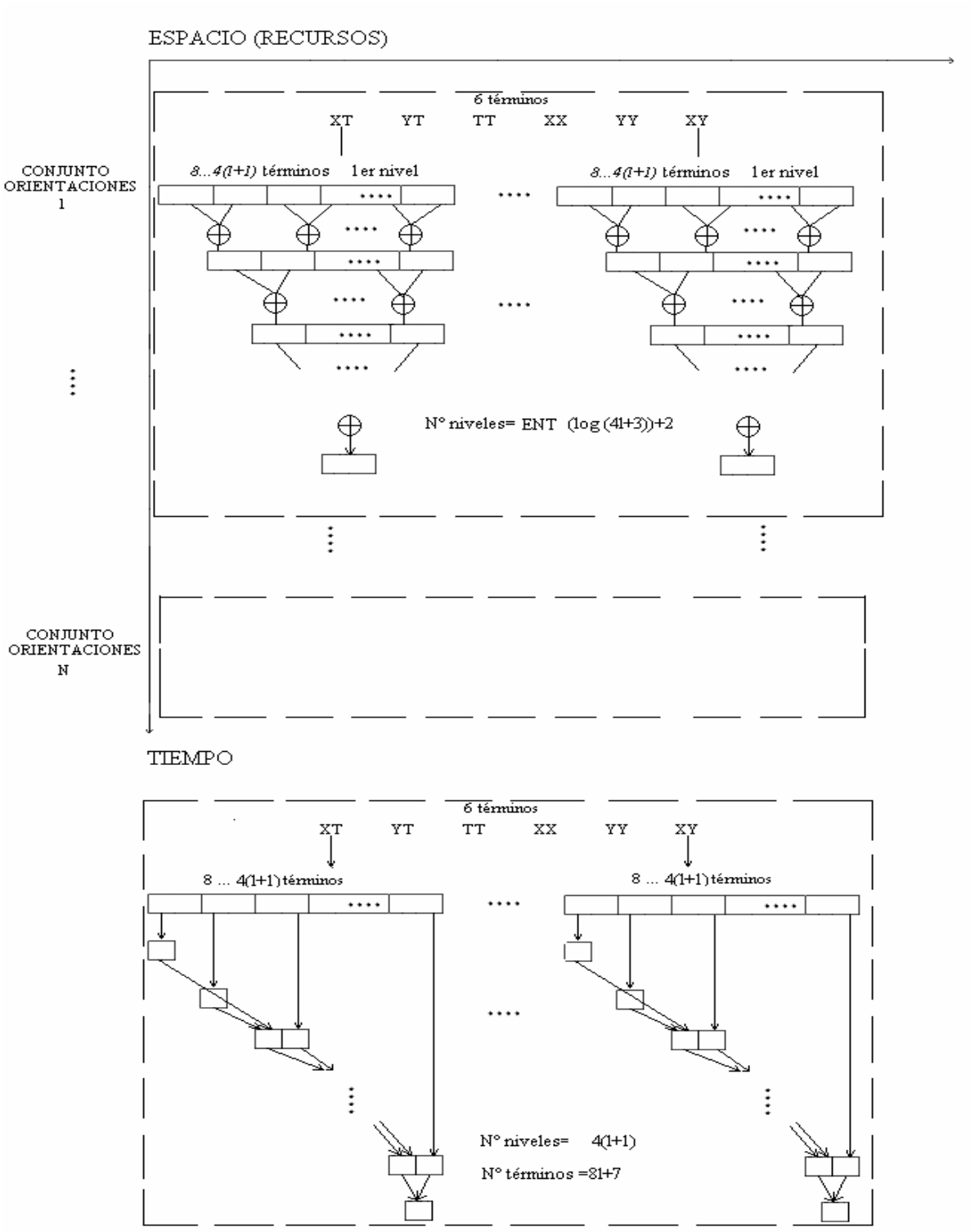


Figura 5.11 Suma de contribuciones de los productos de Taylor (coincide con el segundo y último paso para realizar la integral discreta sobre un volumen espacio temporal). Nótese que el diseño es completamente escalable. Se muestra las diferencias del proceso acumulativo en función de la naturaleza del registro usado, operación usando *slices* (superior), el mismo proceso si se utiliza memoria empotrada (*BlockRAM*) (figura punteada inferior).

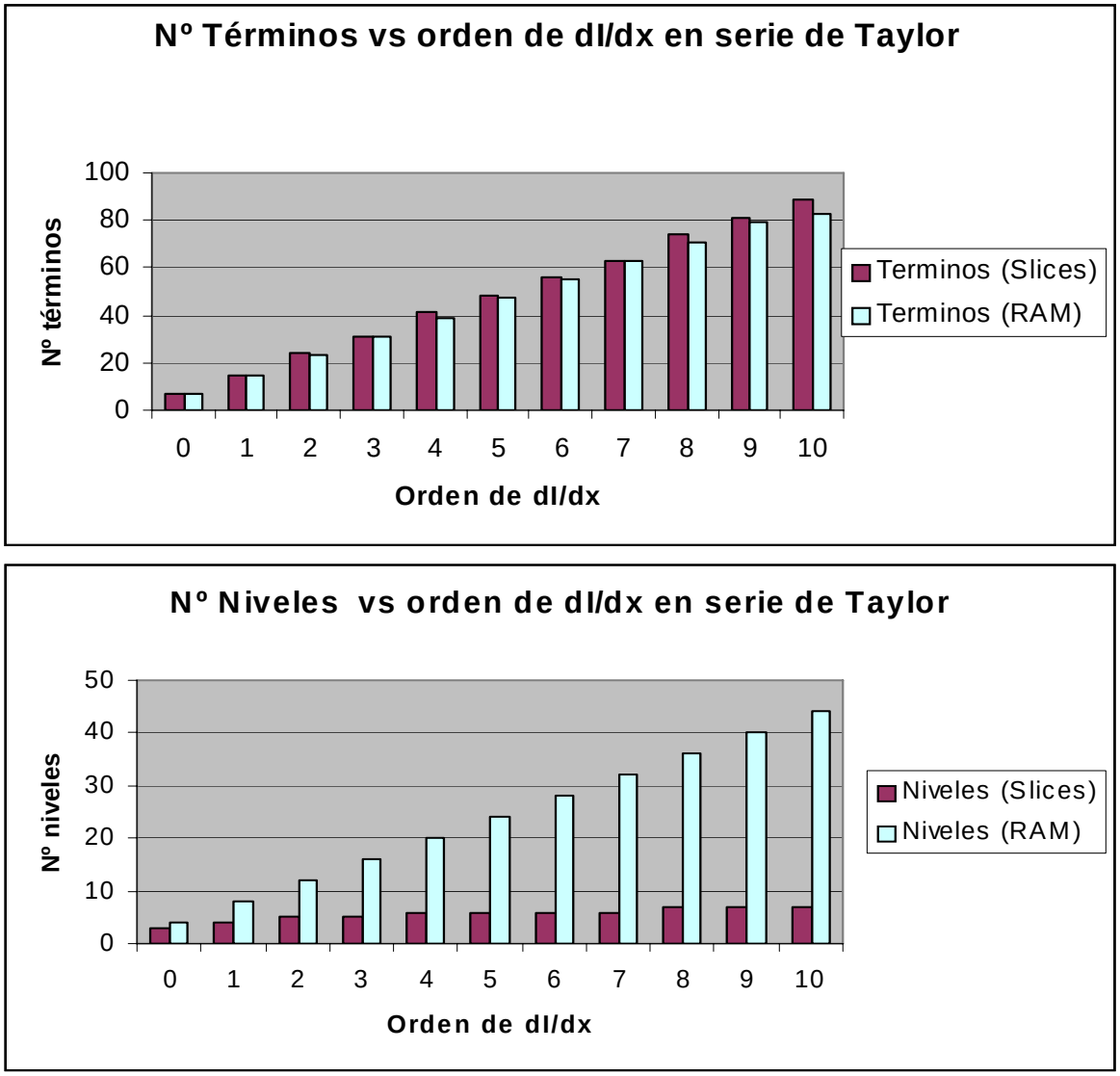


Figura 5.12(a) superior. 5.12(b) inferior. Fabricación de los productos de Taylor (coincide con el primer paso para desarrollar la integral discreta), heredándose la modularidad de los componentes.

En las figura 5.12(a), se muestra el número de términos utilizados en cada una de las dos estrategias, los resultados son similares aunque deben interpretarse correctamente, en el caso de usar *Block* RAM éstos términos se usan de forma puramente secuencial, pudiendo corresponder al mismo término físico reutilizado varias veces. En el caso de usar memoria distribuida, (*slices*), los términos se usan en paralelo con el consumo inherente asociado que conlleva. El número de términos en el caso de usar *slices*, viene caracterizado por una progresión geométrica de razón ½, cuyo valor total depende del orden de la primera derivada en *x* en el desarrollo de Taylor.

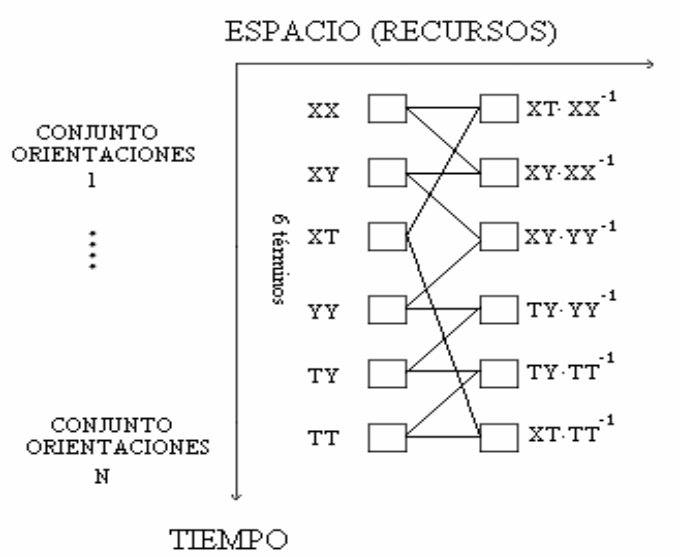


Figura 5.13. Cálculo del sexteto en la etapa de cocientes. Donde se sigue respetando la organización temporal con respecto al conjunto de orientaciones.

Con respecto al número de ciclos, se muestra en la figura 5.12(b) la evolución de éstos frente a las mismas abcisas que el análisis anterior. En este caso claramente el usar *slices*, disminuye el número de ciclos necesitados en esta parte del módulo. El numero de niveles viene dado por la ENT($\log_2(4l+3)$)+2 , siendo ENT la parte entera del número. En el caso de *Block* RAM, el número de niveles coincide con el número de términos ($4(l+1)$) del desarrollo de Taylor

5.2.8. Cocientes

Llegado a este punto, se procede a calcular 6 cocientes con aritmética de punto fijo a partir de los términos de la sección 3.2.4.3 del capítulo III. En este cálculo se usan masivamente *slices* (figura 5.13) debido a que hay que implementar las multiplicaciones y divisiones a medida, puesto que no se dispone de módulos aritméticos embebidos en la placa de prototipado. Se repite esta estrategia para cada conjunto de orientaciones, a partir de ahí se pasa al PC externo que se ocupará de continuar el cálculo.

5.3. Descripción del sistema adaptado al *hardware*: arquitecturas básica y extendida de procesamiento de movimiento

5.3.1. Introducción

En esta sección se desarrolla la arquitectura del diseño a otro nivel de abstracción conceptual, avanzando en la descripción realizada en el presente capítulo. Se muestran 2

arquitecturas denominadas *básica* y *extendida* respectivamente, cada una con sus funcionalidades y especificaciones.

5.3.2. Arquitectura básica

Se presenta a continuación el conjunto de parámetros definidos para conferir una especificidad a la arquitectura general propuesta anteriormente. La arquitectura básica consta del módulo de pre-procesamiento de filtrado espacial, el módulo de filtrado temporal, el de filtrado espacial piramidal, y el de orientación, con los parámetros señalados en la tabla 5.V.

TABLA 5.V Parámetros y características de la arquitectura básica.
Notación: L≡Longitud de la máscara; N_i≡Nº bits asociados a etapa *i*; R≡registro (*slices*), BR≡*Block RAM* (Memoria empotrada), O[z]≡orden del filtro z, O^O[z]≡orden ortogonal del filtro z.
(Tamaño de fotograma 128×96)

ETAPA	Nº BITS	Parámetros	
Suavizado	$N_{entrada}$=8 N_{filtro_previo}=6	$L_{mascara}$=7	R(filas)=[7] BR(columnas)=[7,128]
Filtro IIR	$N_{fIIRiltro}$=6 $N_{operación\_recursiva}$=9	L_{buffer}=3	
Filtro FIR	N_{filtro}=8 $N_{convolución\_filas}$=9 $N_{convolución\_columnas}$=10	O[x]=5 O[y],O[t]=2 $L_{máscara=}$21	21 términos/pirámide temporal 3 ·6 BR(columnas) de [21,128]
Orientación	$N_{pesos\_steering}$=6 $N_{steering}$=10	O^O[x,y]=2 Orientaciones=24	58 R(términos/orientación)

Cabe destacar que este diseño por si mismo, ya constituye un módulo de pre-procesamiento [BOT04][BOT06a][BOT06b] para la familia de algoritmos de flujo óptico basada en métodos diferenciales (véase capítulo II).

5.3.3. Arquitectura extendida

Como arquitectura extendida, se va a denominar el proceso de co-diseño completo, es decir la arquitectura básica, además de la creación de la serie de Taylor, sus derivadas y los cocientes que conducen al cálculo de velocidad directa e inversa.

TABLA 5.VI Parámetros y características de la arquitectura extendida.
Notación: L≡Longitud de la máscara; N_i≡Nº bits asociados a etapa *i*; R≡registro (*slices*),
BR≡*Block RAM* (Memoria empotrada), O[z]≡orden del filtro z, O^O[z]≡orden ortogonal del filtro z.
O^T[z]≡orden del desarrollo de Taylor en z. OF^T≡orden final de desarrollo.
(Tamaño de fotograma 128×96)

ETAPA	Nº BITS	Parámetros	
Suavizado	$N_{entrada}$=8 N_{filtro_previo}=6	$L_{mascara}$=7	R(filas)=[7] BR(columnas)=[7,128]
Filtro IIR	N_{filtro_IIR}=6 $N_{operación\_recursiva}$=9	L_{buffer} =3	
Filtro FIR	N_{filtro}=8 $N_{convolución\_filas}$=9 $N_{convolución\_columnas}$=10	O[x]=4 O[y],O[t]=2 $L_{máscara=}$17	15 términos/pirámide temporal 3 ·5 BR(colum) de [17,128]
Orientación	$N_{pesos\_steering}$=6 $N_{steering}$=10	O^O[x,y]=2 Orientaciones=8	41 R(términos/orientación)
Serie Taylor	$N_{desarrollo\_Taylor}$=12	O^T[x]=3	16 R(términos/orientación)
Derivada Taylor	N_{pesos_Taylor}=11 $N_{derivada\_Taylor}$=17	OF^T=6 O^T[x]=4	48 R(términos/orientación)
Cocientes	$N_{cocientes\_Taylor}$=12	6 (términos/orientación)	

El conjunto de parámetros varía respecto al usado en la arquitectura básica, ya que se requieren más etapas encadenadas en la placa de prototipado.

5.4. Interconexión de etapas: Estructuras de datos y comunicación

5.4.1. Segmentación de cauce asíncrona. *Micropipeline*

Se procede a ilustrar la segmentación de cauce de grano grueso diseñado al efecto, en la figura 5.14. Los bancos externos de la placa participan activamente en la operación de cálculo (representados en gris), escribiendo/leyendo sobre ellos en el proceso *Host* y en el procesado temporal IIR. El proceso *Host* merece una mención aparte en el siguiente apartado. Los datos parten del proceso *Host* y son transferidos por el bus PCI hasta banco externo 0, a

partir de ahí, el módulo *Envío/Recepción* se está ejecutando de forma paralela al conjunto de etapas que conforman el cauce.

Se parte del módulo *Filtrado Paso Baja* (Suavizado) donde se aplica el proceso visto en la sección 5.2.2, a continuación el módulo *Filtrado Temporal* IIR hace uso del banco externo 3, en el almacenamiento de los fotogramas necesarios para ir efectuando la recursión (5.2.3), posteriormente se pasa al módulo *Filtrado Espacial piramidal*, donde se emplea masivamente el paralelismo para construir todas las derivadas espacio-temporales que proporcionarán información a las etapas posteriores.(5.2.4). La siguiente etapa queda implementada con el *módulo Steering* (5.2.5), donde existe la posibilidad a la finalización de ésta para mandar los datos de nuevo al módulo *Envío/Recepción* para la recepción de los datos procesados (quedando definida la arquitectura básica aquí). Para la arquitectura extendida comienza de forma idéntica a la básica pero con los parámetros adjuntos en la tabla

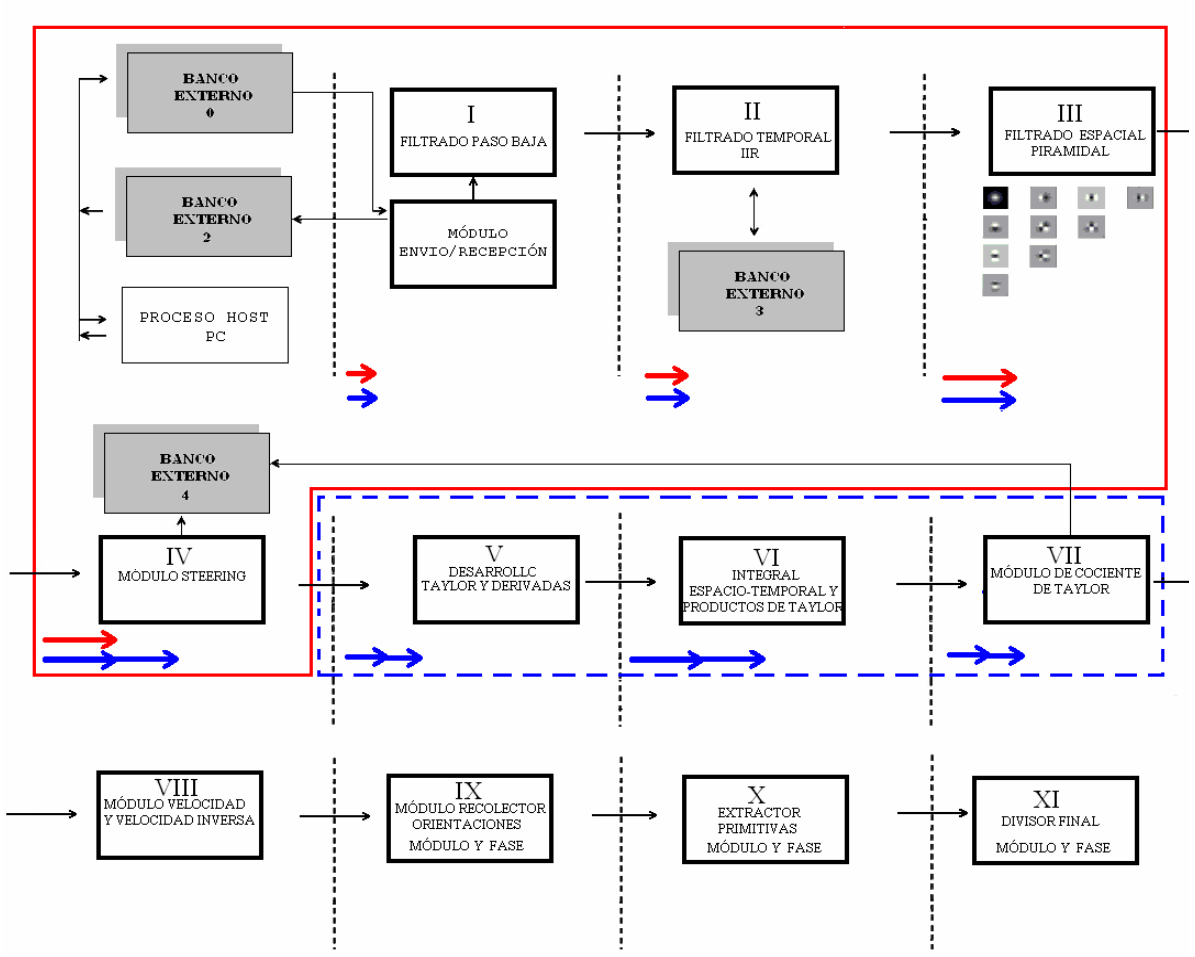


Figura 5.14 Esquema de la segmentación de cauce utilizada, se han utilizado 3 niveles para poder hacer uso de una representación total de los módulos necesarios. Las zonas sombreadas representan los bancos de memoria externos a la FPGA utilizados. Las zonas rojas esquematizan la arquitectura básica y la roja+azul, la arquitectura extendida, el resto de módulos se ejecutan en un microprocesador de propósito general.

5.VI, a partir de la cual se va hacia el *módulo desarrollo Taylor y derivadas* (5.2.6), *Integral y productos de Taylor* (5.2.7), terminando la arquitectura extendida con el *módulo de cocientes de Taylor* (5.2.8).

El significado de las flechas es el siguiente, se usa el color rojo para la arquitectura básica y el color azul para la extendida.

La longitud de cada línea (flecha de color) es proporcional al número de ciclos que tarda cada etapa dentro del cauce. (Una flecha formada por la prolongación de otras *n* indica que esa etapa se bloquea hasta ejecutarse *n* veces, formalmente es como si se tardase el número de ciclos de esa etapa multiplicado por *n*). Por eso cuando se usa la arquitectura básica es posible calcular todas las orientaciones masivamente, debido a que el cálculo *hardware* termina en esa etapa, sin embargo en la extendida, es necesario llamar a cada módulo otra vez más, secuenciando el paso de datos, para tener *slices* que permitan llegar a etapas más lejanas. Cabe destacar que el proceso es fundamentalmente expansivo en términos de recursos consumidos y volumen de datos transmitidos a la etapa siguiente, como se evaluará posteriormente.

5.4.2. Programa *Host*

Este proceso ha sido diseñado a modo de árbitro, siendo origen y destino de los datos a procesar, después de pasar por el cauce de la FPGA.(figura 5.15). Por medio de unas primitivas que controlan el canal DMA, este proceso se apropia de los bancos externos que se necesiten de la placa, transmite vía DMA los datos y libera de nuevo dichos bancos, usándose esta estrategia para la transmisión de los pesos correspondientes en la etapa de *Steering,* pesos d

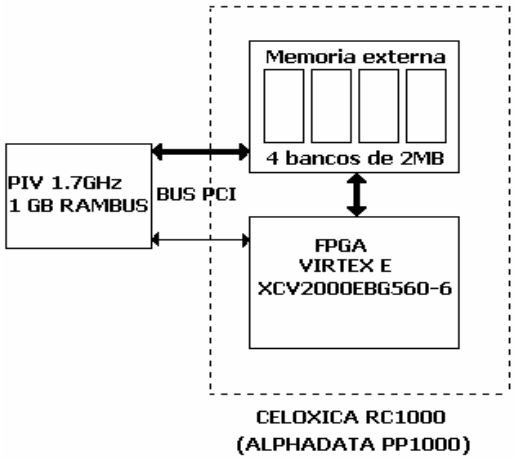


Figura 5.15 Esquema del PC de propósito general y su comunicación con la placa de *Celoxica* (área punteada) mediante el proceso *Host*, mediante DMA se establece un flujo de datos del PC a los bancos externos y a su vez de estos a la FPGA mediante primitivas adecuadas.

Existen también unas primitivas que actúan como semáforos, izando una bandera virtual en el programa *Host* que manda un mensaje hacía su homólogo en *Handel-C* y bloqueando el progreso en éste, hasta que no se reciba un mensaje de arrío de bandera desde la FPGA, desbloqueando el proceso, que a su vez bloquea el proceso en la FPGA. Con este sencillo protocolo tipo *handshaking*. se sincroniza la ejecución concurrente de programas entre el bus PCI y la FPGA.

Mediante una estrategia similar se realizan medidas de tiempo, si se extrae la diferencia entre el tiempo de transmisión, cálculo, recepción y el tiempo de transmisión, recepción, además esta medida se realiza para centenares de iteraciones de cálculo en la FPGA, independizando aún más, si cabe la influencia de la transmisión. El programa *Host* enlaza con el resto del algoritmo ejecutándose en el PC, cuando se interpreta el algoritmo completo como un sistema de co-diseño.

5.5. Coste asociado

Se presenta a continuación una medida de los recursos consumidos, en cada una de las arquitecturas propuestas anteriormente.

5.5.1. Coste *hardware* (Compilaciones parciales)

Se analiza la arquitectura básica, presentando en la tabla 5.VII los *slices* consumidos. Se recuerda que un *slice* equivale a la mitad de un *CLB* (celda lógica), y que el dispositivo sobre el que se ha implementado el diseño, tiene una matriz de (80×120 CLBs). Además, se muestra el número de ciclos de cada etapa y el uso de memoria RAM implícita en la FPGA.

En esta arquitectura básica, la etapa limitante es el filtro FIR, consumiendo 17 ciclos, cabe destacar cómo se realiza la etapa de orientación con un 3l % de gasto de *slices* debido a que se calculan las 24 orientaciones (cada 15º), por transformaciones geométricas (simetría) equivale sólo a calcular la mitad, luego el módulo procesa 24 orientaciones, aunque sólo tiene que calcular por giro *(steering)* 12 de ellas, calculando el resto con simetría (*v.*apéndice II).

También destaca aquí el uso masivo de la RAM en el módulo convolucionador espacial, debido al paralelismo a la hora de calcular todas las convoluciones. Si se examinan la figuras 5.1, 5.6 se constata cómo se define una RAM por cada orden espacial del filtro.

TABLA 5.VII Coste *hardware* de la arquitectura básica

ETAPA	*Slices (% Dispositivo)*	*Nº Ciclos*	*RAM (% Dispositivo)*
Suavizado	289 (2%)	4	(1%)
Filtro IIR	190 (1%)	9	(1%)
Filtro FIR	1307 (7%)	17	(36%)
Orientación	5961 (31%)	15	(2%)

Por último, se almacena la información de salida en uno de los bancos externos, copiando esta información mediante llamadas al DMA (a través el bus PCI) es posible guardar esta información de salida de vuelta al PC, para seguir procesando el flujo en *software*.

Con un 41% de uso de la placa en términos de *slices* y un 40% de la RAM implícita, se puede diseñar un coprocesador a medida, para un tamaño de fotograma de 128×96 (el número de filas del fotograma limita la longitud de la RAM embebida que tiene que ser usada). Se puede caracterizar este cauce segmentado con una latencia de 43 ciclos.

Con respecto a la arquitectura extendida, la etapa limitante es la correspondiente a *steering* debido a que se procesan 4 orientaciones (2 en paralelo, y 2 en serie), bloqueando el sistema global hasta tener todas calculadas. Las derivadas de Taylor y los cocientes son los módulos que más consumen en términos de *slices*, debido a que tienen que hacer frente a multiplicaciones y divisiones enteras. Con respecto al uso de RAM resalta la etapa FIR, que al igual que en la arquitectura básica, demanda el uso de una RAM por cada orden del filtro piramidal y la etapa de los cocientes que se encarga de dividir y recolectar los diferentes términos orientados para conformar un vector que entrará al módulo de cálculo de mínimos cuadrados. Para las medidas de parámetros en esta arquitectura, se sigue conservando el tamaño del fotograma anterior (128×96) y la latencia del presente cauce es de 131 ciclos.

TABLA 5.VIII Coste hardware de la arquitectura extendida.

ETAPA	*Slices (% Dispositivo)*	*Nº Ciclos*	*RAM (% Dispositivo)*
Suavizado	289 (2%)	4	(1%)
Filtro IIR	190 (1%)	9	(1%)
Filtro FIR	1233 (6%)	17	(36%)
Orientación	2012 (10%)	30	(2%)
Serie Taylor	2082 (10%)	22	(2%)
Derivada Taylor	5170 (27%)	29	(9%)
Cocientes	8299 (43%)	20	(13%)

5.5.2. Velocidad (Mpuntos/seg)

Se muestra a continuación la eficiencia de cálculo expresada en puntos/segundo para cada una de las arquitecturas consideradas.

En la figura 5.16 se representa la ganancia para ambas arquitecturas en función del número de ejecuciones efectuadas, también se puede caracterizar el cauce por su eficiencia (la ganancia dividida por el número de etapas), que resulta entorno a 0.6, según la figura 5.17, para ambos sistemas, lo cual da una idea de que el escalado del cauce se ha diseñado de forma coherente y satisfactoria.

Como consideraciones previas, se hace notar que los resultados del analizador de tiempos de la herramienta ISE (en sus 3 versiones diferentes), son bastantes moderados respecto a los valores que se pueden llegar a obtener, ajustando manualmente un valor de

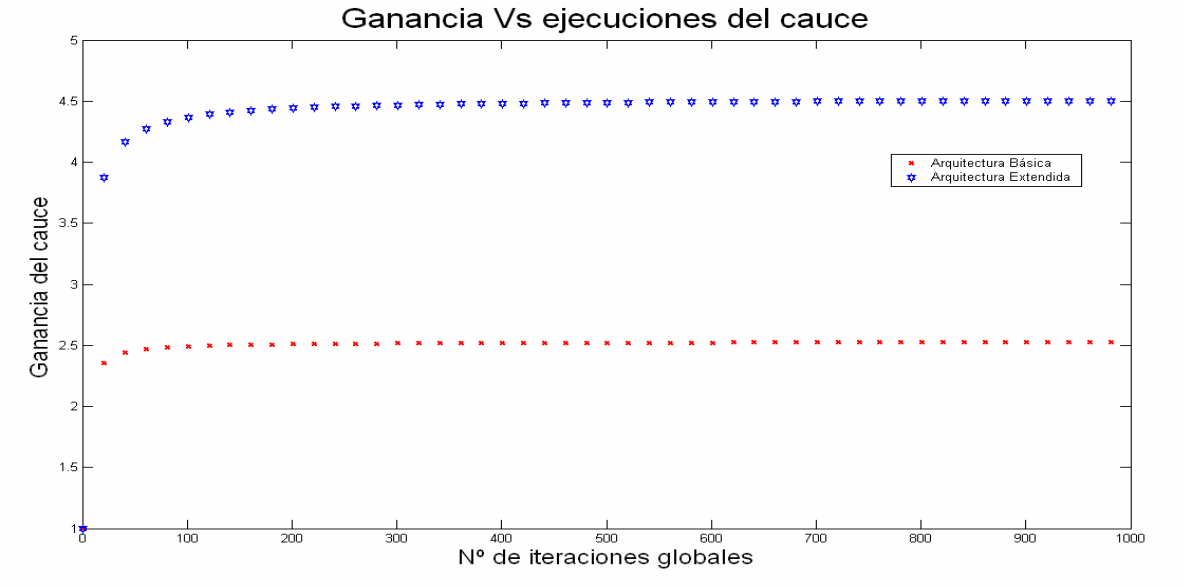


Figura 5.16 Ganancia del cauce vs nº de ejecuciones, según la arquitectura básica y la extendida, hasta converger a 2.52 y 4.51 respectivamente.

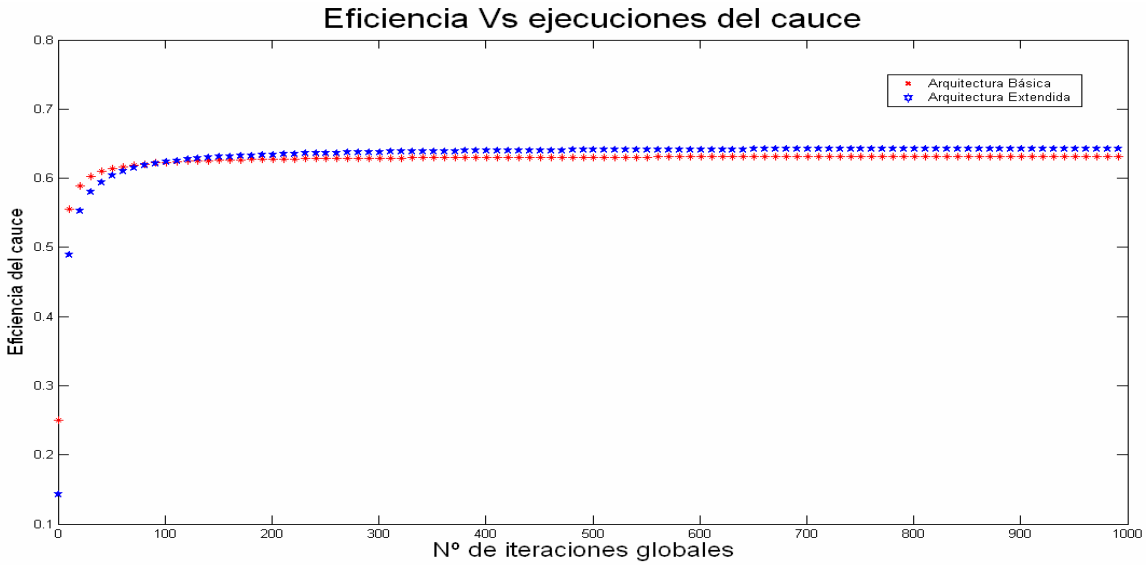


Figura 5.17 Eficiencia del cauce vs nº de ejecuciones, según la arquitectura básica y la extendida, donde se aprecia una convergencia de 0.6 en ambos casos.

frecuencia superior a la que proporciona ISE, que a su vez sea compatible con la ausencia de errores determinados de forma experimental.

Por otra parte, se han realizado experimentos diseñados para hacer medidas de tiempos de forma que se independice el cálculo de la FPGA de la transmisión y recepción de datos desde/hacia el PCI lo máximo posible. Este proceso se ha realizado mediante multiples iteraciones en el cálculo del flujo en la FPGA pero con sólo o una transmisión y una recepción, concluyendo que el tiempo de transmisión oscila aproximadamente entre un 25% y

un 35% del medido de forma experimental, mostrando en las tablas siguientes, los resultados de las medidas sin tener en cuenta este tiempo.

En la tablas 5.X y 5.XI se presentan las medidas obtenidas en: Kilopuntos/seg, frecuencia máxima teórica, la programada en el circuito (referida como aquella que no proporciona valores erróneos como puntos espúreos, resultados distorsionados, etc. Esta frecuencia como se ha comentado previamente, se ajusta por el diseñador de forma empírica formando parte del conjunto de parámetros que inician el circuito), y el número de fotogramas/seg para dos resoluciones diferentes en cada uno del conjunto de etapas encadenadas.

Se constata, que si desea procesar en tiempo real (definiendo éste, como aquel que proporciona una medida mayor de 30 fotogramas/seg, haciendo un símil con las proyecciones cinematográficas usuales), éste se podría alcanzar en *hardware* (correspondiente a la arquitectura básica) con la resolución de 128×96, respecto a la arquitectura extendida se podría procesar idénticamente en *hardware* hasta la etapa VII, (de nuevo a 128×96). Sin embargo, teniendo en cuenta que el uso de esta arquitectura se efectúa para posteriormente realimentar al PC mediante la transferencia de datos, y comprobando que esta migración de datos junto con su cálculo asociado en el PC retarda considerablemente la finalización del proceso, no se obtiene tiempo real en ningún caso para el sistema de co-diseño completo, siendo necesario explorar nuevas alternativas de diseño.

TABLA 5.X Velocidad de la arquitectura básica (puntos de salida).

ETAPA	*Kpps (*ISE*)*	*Máx frec (*ISE*)*	*MHz (experim)*	*Fotogramas/seg 384x288 (128x96)*
I	12500	37.1	50	113 (1017)
I+II	5333	33.8	48	48.22 (433.98)
I+II+III	2352	29.2	40	21.26 (191.4)
I+II+III+IV	2176	27.6	37	19.67 (177.08)

TABLA 5.XI Velocidad de la arquitectura extendida (puntos de salida).

ETAPA	*Kpps (*ISE*)*	*Máx frec (*ISE*)*	*MHz (experim)*	*Fotogramas/seg 384x288 (128x96)*
I	12500	37.1	50	113 (1017)
I+II	5333	33.8	48	48.22 (433.98)
I+II+III	2352	31.0	40	21.26 (191.4)
I+II+III+IV	1066	29.1	32	9.63 (86.67)
I+II+III+IV+V	1000	22.6	30	9.04 (81.36)
I+II+III+IV+V+VI	700	12.6	21	6.32 (56.96)
I+II+III+IV+V+VI+VII	466	8.1	14	4.21 (37.9)

5.5.3. Calidad

Se realiza finalmente un análisis que queda reflejado en la tabla 5.XII, donde se muestra para cada uno de los estímulos usados en el presente trabajo de investigación (Compendio de I,II,II, Diverging Tree, Translating Tree, como se vió en el capítulo IV) la calidad de resultados obtenidos según cada una de las distintas métricas (módulo, fase, Galvin, Galvin ⊥, Barron) en cada una de las arquitecturas diseñadas, sistema *semihardware* además de la versión *software* diseñada en el capítulo III de la presente memoria.

En la plataforma *software* se han usado filtros FIR, 24 orientaciones espaciales y un desarrollo de Taylor truncado completo por otra parte en la plataforma *semihardware* se siguen usando los filtros FIR y 24 orientaciones aunque esta vez con una precisión controlada y un desarrollo de Taylor no completo (es decir menos términos que en la plataforma *software*).

Como se puede apreciar, los errores aumentan a partir de la arquitectura básica, esto es debido principalmente al uso del filtro FIR en el sistema *software* y el sistema IIR en todas las arquitecturas realizadas en *hardware*.

Debido a esta diferencia en el cálculo de filtros, también aparece otra diferencia asociada a la hora de calcular las derivadas temporales de los mismos, ya que se han usado operadores numéricos sencillos en lugar de realizar 3 convoluciones recursivas.

TABLA 5.XII Errores según diferentes, métricas para la versión *software* y *hardware*,con densidad 100%. Los estímulos I, II y III y las secuencias de test *Div. Tree* y *Trans. Tree* se describen en el capítulo IV
Nota: Los valores después del símbolo ± no indica el error sino la varianza asociada al cálculo de cada medida.

	***SOFTWARE* PURO**				
	Barron	**Galvin**	**Galvin ⊥**	**Fase**	**Mod**
Estímulos (I+II+III)	1.61±1.00	0.02±0.06	0.017±0.002	1.56±1.30	0.04±0.03
Div.Tree	12.46±8.00	0.29±0.15	0.18±0.10	13.9±11.0	0.93±0.21
Trans.Tree	18.01±15.00	0.69±0.40	0.13±0.03	9±7	1.52±0.40
	SEMI HARDWARE				
	Barron	**Galvin**	**Galvin ⊥**	**Fase**	**Mod**
Estímulos (I+II+III)	11.39±6	0.32±0.09	0.04±0.03	2.78±2.03	0.70±0.12
Div.Tree	17.8±11	1.2±0.20	0.98±0.07	15.1±12.03	2.1±0.42
Trans.Tree	21.73±18	2.9±0.49	0.56±0.08	13.76±10.00	3.5±0.72
	HARDWARE básica				
	Barron	**Galvin**	**Galvin ⊥**	**Fase**	**Mod**
Estímulos (I+II+III)	15.2±7	1.5±0.30	0.09±0.1	3.9±4.1	1.7±0.2
Div.Tree	29.8±15	2.79±0.53	1.22±1.3	18.01±12.03	3.7±0.92
Trans.Tree	33.13±23	3.1±0.62	1.03±0.11	15.76±12.2	4.3±1.11
	HARDWARE extendida				
	Barron	**Galvin**	**Galvin ⊥**	**Fase**	**Mod**
Estímulos (I+II+III)	17.9±8	1.92±0.30	0.13±0.1	4.6±4.5	2.6±0.4
Div.Tree	31.8±17	3.2±0.89	1.31±1.42	27.3±17.03	5.1±1.46
Trans.Tree	38.29±29	3.8±1.2	1.05±0.04	21.1±13.1	7.1±2.12

Otras diferencias se pueden justificar debido a los coeficientes de los filtros que se han guardado siguiendo un escalado basado en potencias de 2, así mismo las orientaciones espaciales tampoco se han escogido siempre de forma simétrica, escogiendo ángulos cuyos coeficientes del *steering* sean números cuya representación en punto flotante sea sencilla.

Cabe destacar también una disminución del error de la métrica de Galvin perpendicular para el *Translating Tree* con respecto al *Diverging Tree*, esto es debido a la naturaleza de la métrica perpendicular que indica la proyección del vector diferencia sobre uno perpendicular a la medida experimental, es decir esta métrica requiere una interpretación distinta, prácticamente hace referencia a una medida de la orientación del error y puede evolucionar de forma no acorde a otras métricas, como de hecho sucede.

Por último en la figura 5.18, se muestra el comportamiento de los errores al pasar de un tipo de implementación a otro según diferentes las diferentes métricas consideradas.

En la parte superior de la figura 5.18, se puede apreciar como la variación de

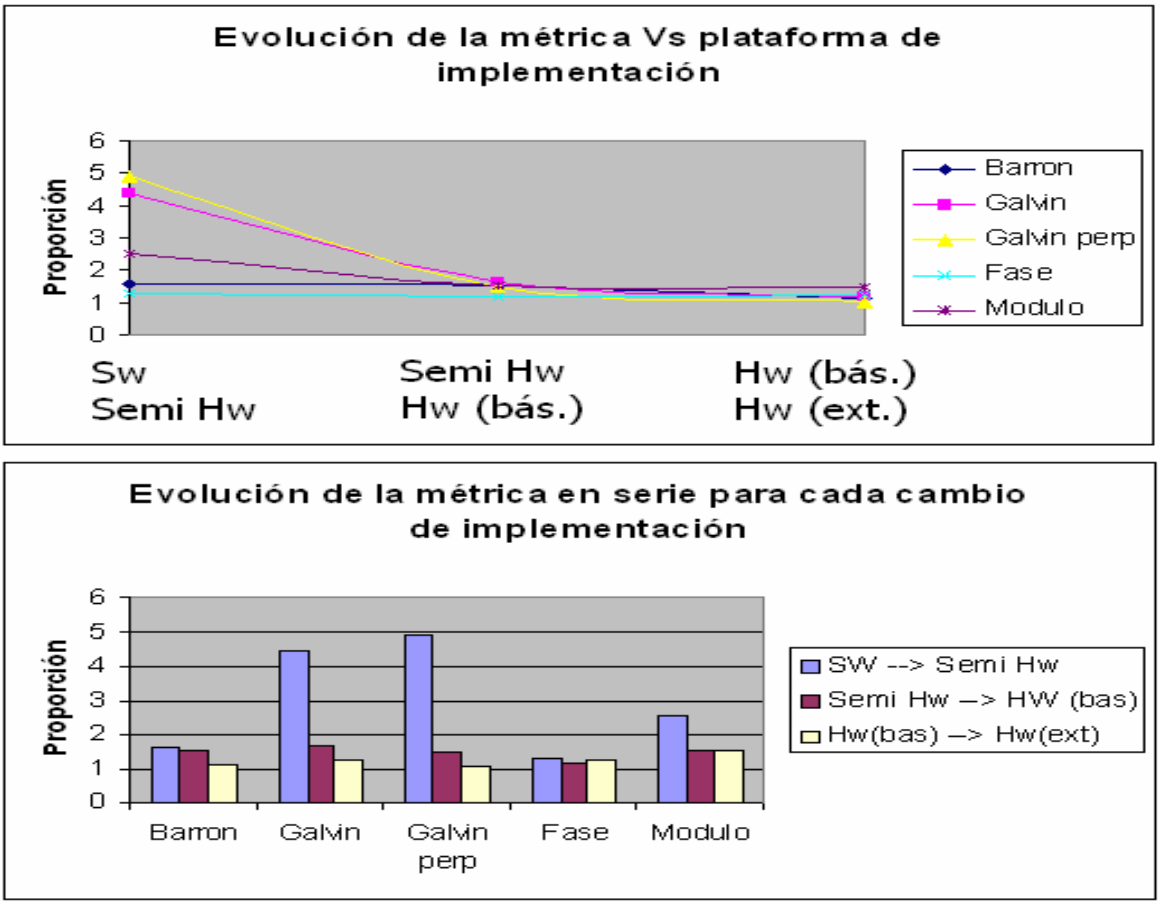


Figura 5.18. Evolución de la métrica de error, para diferentes cambios de implementaciones (arriba). Evolución de cada métrica en serie (abajo).

implementación que más errores acarrea es de *software* a *semihardware,* y dentro de ella, el uso de las métricas de Galvin y Galvin perpendicular, a partir de esta implementación, las demás realizadas en *hardware* propiamente dicho, tienen un comportamiento estable en cuanto al tipo de métricas. Es decir, el cambio que experimentan al pasar de *hardware* básico a *hardware* extendido es aproximadamente similar en todas ellas. Se representa en la parte inferior de la figura 5.18 una información complementaria, aunque esta vez se representan en el eje de abcisas el conjunto de todas las métricas para su mejor comprensión, repitiéndose esta representación para cada cambio de implementación.

Hay que reseñar por último, que si bien los errores obtenidos en *hardware* no son para nada despreciables, estos han sido obtenidos con una densidad de cálculo en torno al 100% .

5.6. Resumen y conclusiones

Se han presentado un conjunto de transformaciones e implementaciones en *hardware* de sendas arquitecturas, una común para modelos de flujo óptico generales basados en gradiente y una especializada para el modelo del que es objeto este trabajo de investigación. Se muestra como un circuito de estimación de flujo óptico puede ser implementado, gracias a un dispositivo reconfigurable como una FPGA, con una arquitectura basada en un cauce segmentado inelástico dirigido por eventos.

Este sistema se integra en un entorno de co-diseño funcional y pone en relieve una metodología de implementación de modelos de flujo óptico en *hardware* reconfigurable, que sienta las bases para futuras implementaciones.

Cada una de la arquitecturas consideradas tienen la propiedad de la escalabilidad, modularidad y adaptabilidad, de forma que es relativamente fácil cambiar cada parámetro para obtener distintas soluciones de compromiso entre velocidad de cómputo y recursos *hardware*.

Por último, se han evaluado los recursos consumidos, concluyendo que la etapa donde se realizan los productos de los diferentes desarrollos de Taylor debido al uso masivo de multiplicaciones y la etapa donde se realizan los cocientes, debido al uso de divisiones, son las que más recursos consumen dentro de la arquitectura extendida, aunque en este caso se gastan *slices* debido a que el diseño de la etapa de filtros orientados permite suficientes libres (con la contrapartida de tener un diseño que calcula menos orientaciones que en la

arquitectura básica y además ocupa un número de ciclos superior, a la hora de ejecutar el *steering* completo).

En la arquitectura básica sin embargo se emplean un conjunto de orientaciones consistentes en 24 canales sobre todo el espacio angular (integrando información cada 15º), pudiendo dedicar más recursos en esta etapa ya que el procesamiento *hardware* finaliza en este punto.

La velocidad de proceso presentada es independiente del tiempo de transmisión/recepción como se ha explicado, para la arquitectura básica se obtienen valores en torno a 2200 kpps y para la extendida 466 kpps, para ir a tiempo real con esta tasa de valores se pueden procesar fotogramas de 280×256 y 128×120 para las arquitecturas básica y extendida , respectivamente.

Con respecto a la exactitud con que operan las diferentes implementaciones hasta llegar al coprocesador final diseñado y su relación con los estímulos de test, se puede concluir que el compendio de los 3 estímulos considerados (Estímulos I,II,III) es el que evidencia un mejor comportamiento, incrementándose éstos desde un error medio en módulo (fase) de 0.04 (1.56º) a unos valores de 1.7 (3.9º) en arq. básica 2.6 (4.6º) para arq.extendida, estando el error final razonablemente acotado. Con respecto a los otros dos estímulos usados los errores crecen de forma apreciable obteniendo errores medios en torno a 26º en fase.

Después de estudiar atentamente las distintas métricas para el cálculo de error se concluye que las métricas de Barron, Fase y Módulo son las más robustas frente a los cambios de *software-hardware*, sin embargo las métricas de Galvin varían notablemente además de proporcionar (Galvin $\perp$) una disminución de su valor al pasar del estímulo *Diverging Tree* al *Translating Tree* (es la única métrica con este comportamiento) justificando estos valores en una medida de la afectación del problema de apertura propio de cada estímulo y su relación con el algoritmo total.

La plataforma *software* tiene todos los parámetros “teóricos” del algoritmo completo, evolucionando el sistema total a medida que se van restringiendo funcionalidades (orientaciones, términos del desarrollo de Taylor, precisión en cada etapa, etc.), así como cambiando la naturaleza de los filtros, (un análisis exhaustivo sobre la estabilidad de los filtros IIR cuya topología sea similar a la construida en este trabajo y de la exactitud de los operadores derivativos que se le aplican vía métodos numéricos es una tarea importante a tener en cuenta).

En el apéndice IV se presentarán diversos ejemplos de aplicaciones reales del sistema diseñado, como secuencias de cámara estática, y secuencias dinámicas de adelantamientos de coches, obtenidas en el marco del proyecto europeo ECOVISION, que también será explicado en el capítulo siguiente.

Hay que destacar que debido a que actualmente existen FPGAS comerciales (*Virtex 5*) [XIL07] donde se tiene *hardware* optimizado para convolución espacial dentro de la propia FPGA, esta arquitectura podría ser extendida, ampliada e incluso modificada ,de forma que se podrían aplicar filtros y en general requerimientos de procesamiento, en cuanto a cantidad y precisión, mucho más elevados que los materializados en este trabajo debido a la disponibilidad de los recursos especializados.

CAPÍTULO VI

Conclusiones

En este último capítulo se presenta el marco científico en el que se ha contextuado este trabajo, las conclusiones finales, indicando las principales aportaciones efectuadas y planteando futuras líneas de investigación.

6.1. Marco científico de este trabajo: Proyectos ECOVISION y DEPROVI

Este trabajo se ha desarrollado en el marco de dos proyectos de investigación, uno europeo y uno nacional:

- ECOVISION [ECO03] *"Artificial Vision Systems based on early cognitive cortical processing"* (IST-2001-32114), desde marzo de 2002 hasta enero de 2005.
- DEPROVI [DEP04] *"Diseño de Sistemas Empotrados para procesamiento de visión en tiempo real. Aplicaciones en medicina, vehículos y robots*" financiado por CICYT, desde enero de 2004 hasta diciembre de 2007. Siendo en ambos investigador responsable el Dr. D.Eduardo Ros Vidal.

En el marco de estos proyectos, se requiere tener prototipos funcionales en un tiempo definido (debido a las revisiones científicas asociadas). Consecuentemente se apuesta por un esquema de diseño a un nivel superior a RTL. Si se utiliza una alternativa a lenguajes algorítmicos de alto nivel orientados a *hardware*, se ahorran recursos, aunque el tiempo de diseño e implementación se incrementa considerablemente

En función de estas condiciones particulares, se opta por la herramienta *DK Design Suite* de *Celoxica* [CEL06b], que permite usar varios lenguajes, escogiendo a su vez *HandelC*, cuya principal característica es la posibilidad de realizar descripciones relativamente rápidas, (comparadas con métodos RTL) desde un grado de abstracción elevado.

El sistema genera también diferentes tipos de salida, utilizando ficheros *netlist* (EDIF), que se integran perfectamente con el entorno del fabricante de FPGAS, *Xilinx* [XIL06], generando estos últimos el fichero final de programación del dispositivo.

6.1.1. ECOVISION

ECOVISION es un consorcio europeo formado por 7 universidades y una empresa, con una heterogeneidad relativa a sus diversos campos de especialización, como diseño VLSI, visión artificial, neurociencia, automovilismo, etc., cuyo objetivo final es formar una arquitectura de procesamiento distribuida para análisis visual adaptativo, usando información precognitiva. Esto es así porque los dispositivos ideados hasta la fecha están lejos de conseguir la eficiencia y exactitud en el análisis de escenas visuales encontradas en vertebrados.

Durante este proyecto se ha realizado una estancia en el University College of London financiada con una beca Marie Curie, dando como resultado una fructífera colaboración entre las universidades de Granada y UCL, donde se han adquirido interesantes conocimientos determinantes para el buen desarrollo de este trabajo.

El aporte de este trabajo dentro de ECOVISION es hacer un estudio completo de la viabilidad de la implementación en *hardware* reconfigurable de un algoritmo de estimación de movimiento fuertemente bioinspirado. El proyecto tiene como campo de aplicación primario la asistencia al conductor, es decir el mundo automovilístico, donde se espera un dispositivo válido de asistencia a diversas maniobras dentro de la conducción en los próximos años.

6.1.2. DEPROVI

Actualmente los dispositivos de tipo FPGA tienen la suficiente complejidad como para realizar el procesamiento de imágenes en tiempo real, pudiéndose definir DSPs personalizados en ellos. El procesamiento en tiempo real de movimiento, estéreo y colores con circuitos específicos hace posible su utilización en diversos campos de aplicación, siendo el procesamiento de movimiento particularmente costoso, por los recursos mínimos que necesita.

Por otro lado, la computación de visión en estéreo requiere alta resolución espacial, ya que la estimación de profundidad se basa en la correlación entre dos imágenes de dos cámaras situadas a cierta distancia para calcular la disparidad espacial entre ambas. La integración de

estos dos tipos de información (movimiento y estéreo) puede ser de mucha utilidad en diversos campos de aplicación, ya que, basándose en estas características se pueden acometer tareas como segmentación de objetos, estimación de distancias, cálculo de movimiento en profundidad, entre otras.

Los objetivos del proyecto se centran en la implementación de circuitos de procesamiento de imágenes en tiempo real y su utilización en diversos campos de aplicación como medicina (plataforma de procesamiento de imágenes bio-inspirado para implantes neuro-corticales en sistemas de ayuda a la visión para deficientes visuales), vehículos (sistemas de asistencia a la conducción) y robots (sistemas empotrados de procesamiento de imágenes en tiempo real como dispositivos de procesamiento inteligente o *smart front-ends*).

El aporte de este trabajo dentro de DEPROVI [DEP04] ha sido la implementación de sendas arquitecturas bioinspiradas de procesamiento de movimiento.

6.2. Discusión

Se presenta un trabajo organizado en 5 capítulos y 4 apéndices donde se expone una metodología de implementación de un modelo de flujo óptico sobre una plataforma reconfigurable de propósito general.

En el comienzo de este trabajo, se han introducido conceptos necesarios para la comprensión y alcance de la naturaleza del problema a tratar, así como las herramientas de las que disponemos para abordarlo. Se ha hablado también del comportamiento cambiante de las disciplinas relacionadas con la visión artificial, siendo necesario interpretar el problema y proponer soluciones desde diversos puntos de vista.

Con estos conceptos desarrollados, se ha realizado una revisión de los principios generales de la estimación de movimiento, estudiando el estado del arte actual, concluyendo que los métodos de gradiente son buenos extractores de movimiento, están basados en modelos biológicos y se prestan de forma directa a implementaciones paralelas. Estos métodos diferenciales proporcionan el mejor rendimiento en exactitud y densidad, aunque con la contrapartida de tener que distinguir entre flujo normal y perpendicular necesitando en las etapas finales métodos de resolución basados en mínimos cuadrados. En general, los algoritmos que combinan resultados sobre el tiempo (varias estimaciones) son más robustos que aquellos que tratan cada fotograma de forma aislada; también es importante reseñar que

como requisito se necesita una tasa de muestreo suficientemente alta para respetar una coherencia temporal.

De forma posterior, se ha explicado, justificado e implementado en un lenguaje de alto nivel, un algoritmo de estimación de flujo óptico robusto (frente a ruido, patrón estático y variaciones de contraste), entre otras consideraciones, fruto de continuas mejoras desde 1992 (McGM [JOH92] [JOH99]), siendo este modelo muy costoso a nivel computacional.

El modelo procesa velocidad direccional directa e inversa a partir de las derivadas de la estructura espacio-temporal de la imagen (representada por un desarrollo de Taylor), combinando medidas del desarrollo para dar un cociente de determinantes. Todas las divisiones usadas para la extracción de resultados están autonormalizadas respecto al contraste, evitando diseñar un mecanismo que controle la ganancia para estabilizar el sistema. La velocidad final se traduce en un cociente de funciones sincronizadas entre sí, que confieren una gran robustez al sistema. Cuando la velocidad directa es pequeña, la velocidad inversa es grande y viceversa, siendo una consecuencia inmediata obviar estimaciones no representativas de la tendencia general, que desvirtúan la estimación de mínimos cuadrados final. El modelo se ha diseñado en sus orígenes siempre atendiendo a las evidencias científicas encontradas en los vertebrados, fundamentalmente mamíferos.

Una vez que se tiene el algoritmo desarrollado e implementado en *software*, se ha diseñado una plataforma que controla eficazmente todos sus parámetros, comportándose a modo de puente entre *software* y *hardware*, donde se persigue obtener un conjunto de parámetros que conformen una configuración suficientemente amplia y descriptiva para el diseño de un coprocesador a medida en la placa de desarrollo. Esta plataforma se ha construido avanzando por etapas, con la ejecución de múltiples simulaciones para el estudio del espacio de parámetros en precisión controlada. Todo esto se valida para un conjunto de estímulos y métricas usadas, representativas todas ellas de diversas tendencias existentes en la medida de error de flujo óptico, para independizar el conjunto de parámetros usados de la naturaleza de la entradas a considerar (secuencias de test). Adicionalmente en esta parte se ha realizado un análisis del error en la estimación del flujo frente a la densidad tomada, concluyendo que éste disminuye a medida que la medida de confianza aumenta, independientemente de la métrica programada, teniendo como contrapartida la disminución de densidad. Aún así, se ha intentado fijar los parámetros que gobiernan estos filtros en el camino de datos, de forma que la densidad siempre esté próxima al 100%. Por último, se ha

realizado una comparación cuantitativa en la que se examina la solución obtenida de modo que se pueda estimar el coste en precisión al pasar a punto fijo todas las etapas.

Partiendo de este estudio de viabilidad, se han diseñado varios módulos sobre *hardware* reconfigurable que se concretan en dos arquitecturas: una común a los modelos de gradiente, que funciona como un procesador convolutivo espacio-temporal múltiplemente orientado y otra más específica, perteneciente al algoritmo objeto de estudio en este trabajo. Ambas arquitecturas son escalables y modulares, extensivas a un dispositivo tipo FPGA que disponga de más recursos que el dispositivo Virtex 2000E usado. Se han evaluado los recursos consumidos, la velocidad de proceso obtenida y la exactitud con que opera el coprocesador diseñado. Es realmente sugestivo, apreciar como el presente trabajo ilustra el proceso de implementación de un modelo de estimación de flujo óptico sobre un dispositivo reconfigurable como una FPGA, gracias a una metodología de diseño apta para extender futuras etapas funcionando en tiempo real.

Por último se han mostrado las principales aportaciones y las líneas futuras que plantea el trabajo realizado, anexando también de forma breve algunos ejemplos de situaciones importantes (como la pérdida de precisión) a partir de secuencias reales del sistema diseñado, secuencias de cámara estática, y secuencias dinámicas de adelantamientos de coches, obtenidas en el marco del mencionado proyecto europeo ECOVISION [ECO06].

Así mismo, también se agrega un diagrama de flujo del algoritmo para su correcto seguimiento y comprensión, además de diversas justificaciones relativas a su robustez y simetría, que luego son utilizadas en la implementación tanto en *software*, *semihardware*, y *hardware*. Se hace una breve revisión de la taxonomía de los dispositivos reconfigurables y la bioinspiración del modelo tratado, que creemos puede ser muy útil a especialistas en otras áreas que estudien este trabajo, debido a la interdisciplinaridad inherente al campo de la ingeniería neuromórfica.

6.3. Principales aportaciones

El trabajo presentando en esta memoria se ha traducido en una serie de aportaciones concretas:

1. Se ha implementado en *software* un modelo de estimación de flujo óptico, cuya principal característica es su bioinspiración y robustez frente a contraste, patrones estáticos y ruido.

Se ha modificado la estructura previa del mismo, optimizando su código y acelerándolo mediante transformaciones geométricas. Como ejemplo se ha aprovechado la simetría espacial en el cálculo de múltiples orientaciones y el precálculo de pesos cuadrados contributivos en las derivadas de Taylor. Además, se han modificado las propiedades de mínimos cuadrados finales, eliminando el cálculo de todas las primitivas de velocidad asociadas a puntos que desvirtúan la estimación final del flujo.

2. Se ha diseñado e implementado una plataforma que tiene como característica el estudio y control de precisión de cada módulo o etapa de procesamiento. Con esta herramienta se pueden realizar simulaciones del modelo con precisión limitada en sus operaciones y se utiliza como paso previo antes de la implementación *hardware* del sistema. Cabe destacar que es posible insertar los módulos *hardware* en cualquier posición (sustituyendo módulos *software*) del camino de datos, independientemente de su posición relativa entre ellos.

3. Se ha comparado la degradación del sistema para distintas precisiones en etapas de procesamiento, añadiendo adicionalmente dos métricas de error propias y demostrando que la medida de confianza de cada estimación (en nuestro caso, el valor absoluto de la suma de la primera y segunda derivadas temporales al cuadrado) evoluciona de forma satisfactoria con respecto a la densidad de flujo estimado.

4. Se ha diseñado y desarrollado una arquitectura configurable, ampliable, masivamente paralela y escalable, compuesta por una segmentación de cauce de 4 módulos: un filtrado espacial paso baja, un filtrado IIR temporal de orden 2 con sólo 3 fotogramas de latencia, una máquina superconvolutiva y un módulo de orientación espacial de cada una de las convoluciones previas. Estos cuatro módulos representan el principal cuello de botella de procesamiento de la mayoría de modelos de estimación de flujo óptico y estéreo basados en gradiente.

5. Se ha ampliado la arquitectura anterior para el cálculo de 7 etapas conceptuales del presente modelo de estimación de movimiento, McGM, con parámetros biológicos y

matemáticos, ajustados de forma heurística y utilizando la plataforma referida en la segunda aportación. Esta nueva arquitectura sigue siendo escalable.

6. Se ha llevado a cabo la caracterización de las arquitecturas anteriores, parametrizando las mismas y evaluando distintas funciones de coste (fundamentalmente consumo de recursos y velocidad de computación). Así mismo, su escalabilidad facilita su posterior viabilidad en una plataforma *hardware* con más recursos de los utilizados en este trabajo.

6.4. Líneas de investigación futuras

El trabajo realizado abre un abanico de interesantes líneas futuras a considerar:

1. Diseñar todo el algoritmo funcionando en tiempo real como un sistema independiente. Es decir, convertir sistema de co-diseño en un *sistema–en-chip* (SOC). Para ello se hace necesario la utilización de *hardware* actualizado con suficientes recursos, (10 millones de puertas, aritmética empotrada, un sistema de adquisición de imágenes donde sea posible capturar los datos directamente de una cámara, etc.).

2. Diseñar un cuantizador no lineal combinado con un predictor adaptativo hacia detrás**.** Se trata de modelar un predictor hacia detrás, que transmite hacía etapas previas la estimación del número de bits necesarios, para ello se dota al sistema de una histéresis temporal ajustable, según una heurística dependiendo del entorno. Con ella es posible ajustar la precisión mínima requerida de forma que el error (según un conjunto de métricas representativas) esté controlado dentro de un margen también ajustable. En combinación con esta reflexión preliminar, es conveniente ensayar otros tipos de cuantización no lineales para optimizar el sistema, al igual que se usa en telefonía, por ejemplo en PCM.

3. Cálculo de profundidad (estéreo) y distinción de color. Se busca extender el cálculo del flujo óptico a la disparidad binocular, midiendo profundidad y ayudar a la estimación de flujo óptico a través de una capa de reconocimiento de color. Esto se puede llevar a cabo

mediante métodos dinámicos de seguimiento de la distribución espectral, que resulten robustos incluso para cámaras infrarrojas o aplicaciones médicas donde las características del espacio colorimétrico son restringidas, como los estudiados por [STO03].

4. Uso de recursión en cada uno de los filtros diseñados. Se busca mejorar la implementación de filtros recursivos, y extender el uso de filtros IIR no sólo a los temporales sino también a los espaciales, permitiendo estudiar su efecto en el sistema global.

5. Aceleración del algoritmo implementado en *software* con instrucciones multimedia (MMX, 3DNOW, SSE2). Se intenta acelerar los cuellos de botella críticos de procesamiento mediante instrucciones multimedia y técnicas de optimización que reduzcan tiempo de cómputo, como desenrollado de bucles, ejecución predictiva y uso intensivo de instrucciones MAC, entre otras.

6. Extender el algoritmo a un cálculo multiescala en función de la información a priori que se conozca. El desarrollo de Taylor establece una interesante analogía geométrica entre la información espacio-temporal y la información multiescala. A partir de un punto dado, se puede explotar esta propiedad para extrapolar el modelo a diferentes escalas usando la misma arquitectura *hardware* que la desarrollada para calcular la serie truncada de Taylor.

7. Añadir una capa de regularización (tanto en *software* como en *hardware*). Una red neuronal de Hopfield [HOP86][HOP06] puede homogeneizar el resultado final, relajando los puntos cuya estimación difiera suficientemente de su vecindad.

8. Ajuste de los parámetros biológicos de McGM mediante computación evolutiva. Al tener una cantidad tan elevada de parámetros que definen el modelo y a su vez, un rango amplio de variabilidad en cada uno de ellos, se hace difícil un ajuste de forma óptima, hasta ahora realizado mediante heurísticas. Se va a intentar relacionar las métricas de error con un ajuste mediante computación evolutiva que obtenga un abanico de parámetros óptimos (los que produzcan un menor gasto de recursos para una determinada aplicación).

9. Relacionar los parámetros de la implementación con el consumo de potencia de la plataforma *hardware* donde va implementada.

6.5. Conclusiones finales

Se ha presentado un trabajo que recoge el desafío de implementar en *hardware* reconfigurable un algoritmo de flujo óptico bioinspirado con unas características y complejidad muy superior a los de su clase (modelos de gradiente). Este modelo se ha ido construyendo por capas desde hace más de 15 años, con más de 60 parámetros (cuya dependencia cuantitativa entre ellos no está estudiada con una profundidad suficiente), y cuyos recursos *hardware* requeridos superan a cualquier sistema de flujo óptico publicado hasta la presente.

Todos los modelos se han diseñado haciendo uso de arquitecturas segmentadas asíncronas (*micropipeline*). Probablemente la eclosión de lenguajes tipo C/C++ y herramientas para diseño de *hardware* (Handel-C [CEL06c], System C [SYS06], CodeSim [REI00] lleve en un futuro no muy lejano a la utilización de este tipo de estructuras y metodologías por parte de los diseñadores que requieren arquitecturas eficientes de procesamiento visual en tiempo real.

La implementación en *hardware* reconfigurable de modelos de procesado digital de señal, particularmente de estimación de flujo óptico, requiere no sólo unos resultados aceptables en cuanto a la precisión y a la velocidad de cómputo de forma independiente, sino además un fuerte compromiso entre ellos dos, siendo éste último, la principal limitación a la que se enfrenta el diseñador. Por lo tanto pueden considerarse como modelos si no diferentes, complementarios a sus versiones homólogas en *software* y con una entidad propia.

Este trabajo abre la puerta a la simulación e implementación en tiempo real de sistemas bioinspirados complejos. A modo de ejemplo, el paradigma de la precisión asociada a cada orientación espacial, frente al número total de ellas, establece una interesante analogía con los sistemas densos biológicos (existencia de muchas neuronas con muy poca precisión frente a pocos canales con capacidad de información ingente), materializando en *hardware* características inherentes de la ingeniería neuromórfica.

CAPÍTULO VI

Conclusions

This last chapter summarizes the scientific framework where this work has been developed, as well as the final conclusions including its main contributions and future research lines.

6.1. Scientific Framework: ECOVISION and DEPROVI Projects

This work has been developed within a European and a national research projects:

- ECOVISION [ECO03] *"Artificial Vision Systems based on early cognitive cortical processing"* (IST-2001-32114), from March 2002 to January 2005.
- DEPROVI [DEP04] *"Design of Embedded Systems for Real-time Vision Processing. Applications to Medicine, Vehicles and Robots*" funded by CICYT, from January 2004 to December 2007.

  Dr. Eduardo Ros Vidal was the project leader for both projects.

Both projects require functional prototypes to be ready within concrete deadlines, due to associated scientific evaluations. Consequently, a design scheme at an abstraction level higher than RTL is required. The use of other alternatives but high-level algorithmic languages oriented to hardware may result in resource optimization, although design and implementation times increment largely.

With these premises, the selected design tool was *DK Design Suite* from *Celoxica* [CEL06b], which allows the use of several languages. Thus, the selected language is *HandelC*, whose main characteristic is the possibility of doing fast descriptions, compared to RTL-based methods, from a high abstraction level. The system also generates different kinds

of outputs, through EDIF netlists, which are integrated within the FPGA vendor tools that generate the final programming files.

6.1.1. ECOVISION

ECOVISION is a European consortium formed by 7 universities and 1 company with relative heterogeneity between their specialization fields, which include VLSI design, artificial vision, neuroscience, automotive applications, etc. The final goal is to develop a distributed processing architecture for adaptive visual analysis using precognitive information. This is due to the fact that the existing devices up to date are far from achieving the accuracy and efficiency in visual scene analysis that are found in vertebrates.

During this project, the candidate was granted a Marie Curie fellowship at the University College of London, which resulted in a very productive collaboration between both UCL and the University of Granada and provided the candidate with the necessary background for the development of this work.

The contribution of this work within ECOVISION is a complete viability study of the implementation with reconfigurable hardware of a highly bio-inspired motion estimation algorithm. The primary application field of the project is driver assistance, with a valid assistance device to different maneuvers expected in the next years.

6.1.2. DEPROVI

FPGA devices are actually complex enough to host real-time image processing, being possible to define custom DSPs for this technology. Real-time stereo processing of movement and colors with specific circuits makes possible applications in different fields, movement processing being especially costly due to required resources.

On the other hand, stereo vision computation requires high spatial resolution, since depth estimation is based on the correlation between images from two separated cameras through their spatial disparity. The integration of stereo and movement processing may be quite useful in diverse application fields, since using both of them it is possible to perform tasks as object segmentation, distance estimation, depth movement computation and others.

The objectives of the project focus on the implementation of circuits for real-time image processing and their use in diverse application fields such as medicine (bio-inspired image processing platform for neuro-cortical implants in assistance systems for the visually

impaired) , vehicles (systems for driving assistance) and robotics (embedded real-time image processing systems as smart front-ends).

The main contribution of this work within DEPROVI has been the implementation of two bio-inspired architectures for movement processing.

6.2. Discussion

This work is organized in 5 chapters and 4 appendixes that show a methodology for the implementation of an optical flow model on a general-purpose reconfigurable platform.

At the beginning o this work, the required concepts for the understanding of the problem were introduced, as well as the available tools. It has been also discussed the changing behavior of the related disciplines, being necessary a proper interpretation of the problem and the proposal of solutions from different points of view.

With these premises, a revision of the general principles of movement estimation has been carried out, including a study of the state of the art. It has been concluded that gradient methods are adequate movement extractors while being based on biological models and being suitable for parallel implementations. These differential methods provide the best performance in density and accuracy, although with the disadvantage of the necessity to distinguish between normal and perpendicular flows that require resolution methods based on least squares for the final stages. In general, algorithms that combine results over time (several estimations) are more robust than those processing each frame separately; it is also important to note that a sampling rate high enough is required in order to maintain temporal coherency.

It has been shown, justified and implemented using a high-level language an algorithm for robust (against noise, static patterns and contrast variations) optical flow estimation; this model, very costly from a computational point of view, has been under continuous development from 1992 (McGM [JOH92] [JOH99]).

The model processes direct and inverse directional speed from the derivatives of the spatial-temporal structure of the image (represented with a Taylor series), combining measurements of the series to obtain a determinant quotient. All the divisions used for the result extraction are normalized with respect to contrast, thus avoiding the design of a mechanism of gain control to stabilize the system. The final speed results in quotient of functions that are synchronized and provide great robustness to the system. When direct speed

is small, inverse speed is great and vice versa, being an immediate consequence the oblivion of non representative estimations of the general tendency that distort the final least squares estimation. The model has been designed from its origin taking account of scientific evidence found in vertebrates, especially mammals.

Once the algorithm has been developed and implemented in software, a platform that controls all its parameters has been designed. This platform acts as bridge between software and hardware, trying to obtain a set of parameters that form a wide and descriptive enough configuration for the design of a custom coprocessor on the development board. This platform has been built in several stages, carrying out multiple simulations for the study of the parameter space in controlled precision. This has been validated for the set of stimuli and metrics used, these representing different tendencies in the measurement of error in optical flow, in order to make independent the set of parameters from the different inputs (test sequences). Moreover, an analysis of the error in the estimation of optical flow against chosen density has been carried out, resulting in the conclusion that this error decrements as the reliability increments, for any metric, although density decrements. Even though, the parameters controlling these filters in the data path have been adjusted in order to keep density always close to 100%. Finally, a quantitative comparison that examines the obtained solution has been carried out, so it is possible to estimate the cost in precision when all the stages are move to fixed point.

From this viability study, several modules have been designed on reconfigurable hardware, including two architectures: one of them is common to the gradient models and works as a convolutive spatial-temporal processor with multiple orientations, while the other one is more specific and belongs to the class in which the algorithm object of this work is. Both architectures are scalable and modular and can be extended to an FPGA device with more resources than the available Virtex2000E. The used resources, the resulting throughput and the obtained accuracy have been evaluated for the designed coprocessor, being really suggestive the fact that this work shows the implementation process of an optical flow model over a reconfigurable device, thanks to a design methodology able to be extended to future stages while working real-time.

Finally, the main contributions and future research lines have been shown, also illustrating briefly different examples of important issues, as precision losses, using real sequences from the designed system, sequences from static cameras and dynamic sequences from passing maneuvers in cars, all obtained with the European Project ECOVISION

[ECO03]. Also, an annex includes a flowchart for the algorithm in order to follow and understand it, as well as several issues about its robustness and symmetry that are later used for its implementations. A brief revision of the reconfigurable devices structure and the bio-inspiration of the model is also included, which may be useful for specialists in other areas studying this work due to the multiple fields involved in neuromorphic engineering.

6.3. Main Contributions

The work presented in this thesis has resulted in the following contributions:

1. A software model for estimation of optical flow has been implemented, whose main characteristics are its bio-inspiration and robustness against contrast, static patterns and noise. Its previous structure has been modified, optimizing the code and speeding it up through geometric transformations. As an example, spatial symmetry in the computation of multiple orientations and the precomputation of contributive square weights in Taylor derivatives has been used. Moreover, the final least squares properties have been modified, suppressing from the computation all the speed primitives associated to points that degrade the final estimation of flow.

2. A platform for the study and control of precision in each processing stage and module has been designed and implemented. With this tool, simulations of the model with limited precision can be carried out, and it may be used as a previous step to the hardware implementation of the system. It must be noted that it is possible to insert hardware modules, replacing software modules, in any stage of the data path, independently of their relative positions.

3. Degradation of the system for several precisions in processing stages has been compared, including two new error metrics and showing that the reliability measurement of each estimation (in this case, the absolute value of the square of the sum of the first and second time derivatives) evolves satisfactorily regarding estimated flow density.

4. A reconfigurable, extensible, massively parallel and scalable architecture has been developed. It is composed of 4 pipelined modules: spatial low-pass filtering, 2nd order IIR time filtering with only 3 frame latency, a superconvolutive machine and a spatial orientation module for each previous convolution. These four modules represent the main processing bottleneck for most optical flow and stereo estimation models based on gradients.

5. The previous architecture has been extended for the computation of 7 conceptual stages of the present model of motion estimation, McGM, with biological and mathematical parameters that are heuristically set with the platform referred in the second contribution. This new architecture is still scalable.

6. The previous architectures have been characterized through parameterization and the evaluation of several cost functions (basically resource usage and computation throughput). In the same way, scalability allows its viability within a hardware platform with more resources available than those in this work.

6.4. Future Research Lines

The work that has been carried out leaves several interesting research topics to be considered:

1. The design of the algorithm as an independent real-time system, i.e., the transformation of this co-design system into a System-on-Chip (SoC). For this, it is required the use of updated hardware (10 million gates, embedded arithmetic, direct image acquisition from a camera, etc.).

2. The design on a non-linear quantizer combined with a feedback adaptive predictor**.** For the modelization of the feedback predictor, which transmits to previous stages the estimation of the required number of bits, the system has to include temporal adjustable hysteresis depending on the environment. With this, it is possible to adjust the minimum required precision, so the error (with a representative set of metrics) is controlled within

an also adjustable margin. Combined with this preliminary reflection, it is convenient to try other types of non-linear quantizations for system optimization, as it is usual in telephony, with PCM as example.

3. Color distinction and depth computation (stereo). The objective is to extend the computation of optical flow to the binocular disparity, measuring depth and helping the estimation of optical flow through a color recognition layer. This can be done with dynamic methods of spectral distribution monitoring that are robust even for infrared cameras or medical applications, where the characteristics of the colorimetric space are restricted, as studied by [STO03].

4. Use of recursion for each designed filter**.** This may improve the implementation of recursive filters, extending the use of IIR filter not only to the time domain but also for spatial filters, allowing the study of their effect to the global system.

5. Acceleration of the software algorithm with multimedia instructions (MMX, 3DNOW, SSE2). This may speed up critic processing bottlenecks with multimedia instructions and optimization techniques that reduce computation time, such as loop unfolding, predictive execution and intensive use of MAC instructions, among others.

6. The extension of the algorithm to a multiscale computation as a function of *a priori* information. The Taylor series establishes an interesting geometric analogy between spatial-temporal information and multiscale information. From a given point, this property may be exploited for extrapolating the model to different scales using the same hardware architecture than this developed for the computation of the truncated Taylor series.

7. Addition of a regularization layer (both in hardware and software). A Hopfield neural network [HOP86][HOP06] may homogenize the final result, relaxing those points whose estimation differs significantly from its vicinity.

8. Adjustment of the biological parameters of McGM through evolutive computation. With such a high amount of parameters defining the model and, at the same time, a wide range of variability for each one, it is difficult to achieve an optimum optimization, which has been made heuristically up to date. One of the possible options is to relate the error metrics to an optimization with evolutive computation that obtains a set of optimum parameters (those resulting in the lower cost for a given application).

9. To relate the parameters of the implementation with power consumption of the hardware platform.

6.5. Final Conclusions

This works approaches the implementation using reconfigurable hardware of a bio-inspired optical flow algorithm, whose characteristics and complexity are far larger than those of other gradient models algorithms. This model has been developed layer by layer during the last 15 years, with more than 60 parameters (whose quantitative dependencies has not been yet studied in the proper depth) and requiring an amount of hardware resources that are beyond any other optical flow system up to date.

All the models have been designed making used of asynchronous pipelined architectures (micropipelines). Probably, the advent of C/C++-like languages and hardware design tools (Handel-C [CEL06b], System C [SYS06], CodeSim [REI00]) will lead in a not distant future to the use of this kind of structures and methodologies by designers requiring efficient architectures for real-time visual processing.

The implementation in reconfigurable hardware of digital signal processing models, particularly those for optical flow estimation, requires not only acceptable results regarding precision and computation time independently, while also an strong compromise between these two parameters, which is the main limitation that designers confront. Thus, these models may be considered as complementary, if not different, to their software homologous.
This work leaves doors open to simulation and real-time implementation of complex bio-inspired systems. As an example, the paradigm of precision associated to each spatial orientation against the total number of orientations establishes and interesting analogy with dense biological systems (large number of neurons with very little precision for a few

channels with huge information capacity), which materializes in hardware inherent characteristics of neuromorphic engineering.

APÉNDICE I

Sistemas reconfigurables

AI.1. Introducción

Tradicionalmente se han explorado dos alternativas computacionales para la ejecución de algoritmos.

La primera consiste en el uso de tecnología cableada, mediante un circuito integrado de aplicación específica (ASIC) para realizar las operaciones sobre el *hardware*. Los dispositivos ASICs se diseñan para ejecutar una operación determinada y por lo tanto, son bastante rápidos y eficientes cuando ejecutan la operación precisa para la que fueron diseñados. Sin embargo, el circuito no puede ser modificado después de su fabricación. Si se desea modificar alguna parte del circuito, éste debe ser rediseñado y fabricado de nuevo, lo cual puede llegar a ser un proceso bastante costoso, en especial cuando el circuito tiene que ser reemplazado en un gran número de sistemas.

La segunda alternativa consiste en el uso de procesadores programables por *software*. El procesador se encarga de ejecutar un conjunto de instrucciones que realizan una operación determinada, donde la funcionalidad de un sistema se puede modificar cambiando las instrucciones *software* sin tener que alterar el *hardware*. Sin embargo, la desventaja de esta flexibilidad es que el rendimiento siempre será inferior al de un ASIC para la misma aplicación. El procesador debe leer las instrucciones de memoria, interpretar su significado y después ejecutarlas. Por lo tanto, se incurre en un coste adicional en la ejecución de cada instrucción. Además, el conjunto de instrucciones que puede ser empleado en un programa se define antes de la fabricación del procesador. Cualquier operación que se quiera implementar debe expresarse a partir de las instrucciones existentes.

AI.2. Sistemas Reconfigurables.

La computación reconfigurable (Figura A1.1) surge como una alternativa intermedia entre el rendimiento de las implementaciones en *hardware* de funcionalidad fija y la flexibilidad de las soluciones programables por *software*. Como los ASICs, los sistemas reconfigurables destacan por su capacidad de implementar circuitos especializados directamente en *hardware*.

Además, como sucede con los procesadores programables, los sistemas reconfigurables poseen recursos funcionales que pueden modificarse como respuesta a cambios en los parámetros operacionales o los conjuntos de datos.

En resumen, para la implementación de una aplicación, los sistemas reconfigurables ofrecen un mejor rendimiento que una implementación *software* sobre un procesador de propósito general, y un nivel de flexibilidad mayor que una implementación fija en *hardware*. Una clasificación de los sistemas reconfigurables puede verse en la Figura A1.2.

Los dispositivos reconfigurables, [MAX06] contienen un conjunto de elementos computacionales cuya funcionalidad se define mediante múltiples bits programables de configuración. Estos elementos, denominados “bloques lógicos”, se unen mediante recursos de interconexión que también son programables. De esta manera, un circuito digital puede implementarse en *hardware* reconfigurable realizando las funciones lógicas del circuito en los bloques lógicos y usando la red de interconexión programable para unir los bloques entre sí. A modo de ejemplo, se puede apreciar la estructura de una FPGA en la figura A1.3.

Los dispositivos reconfigurables normalmente están conformados por unidades lógicas reconfigurables y un procesador de propósito general.

Figura A1.1. Alternativas de implementación. Programabilidad frente a especialización para diferentes sistemas *hardware*.

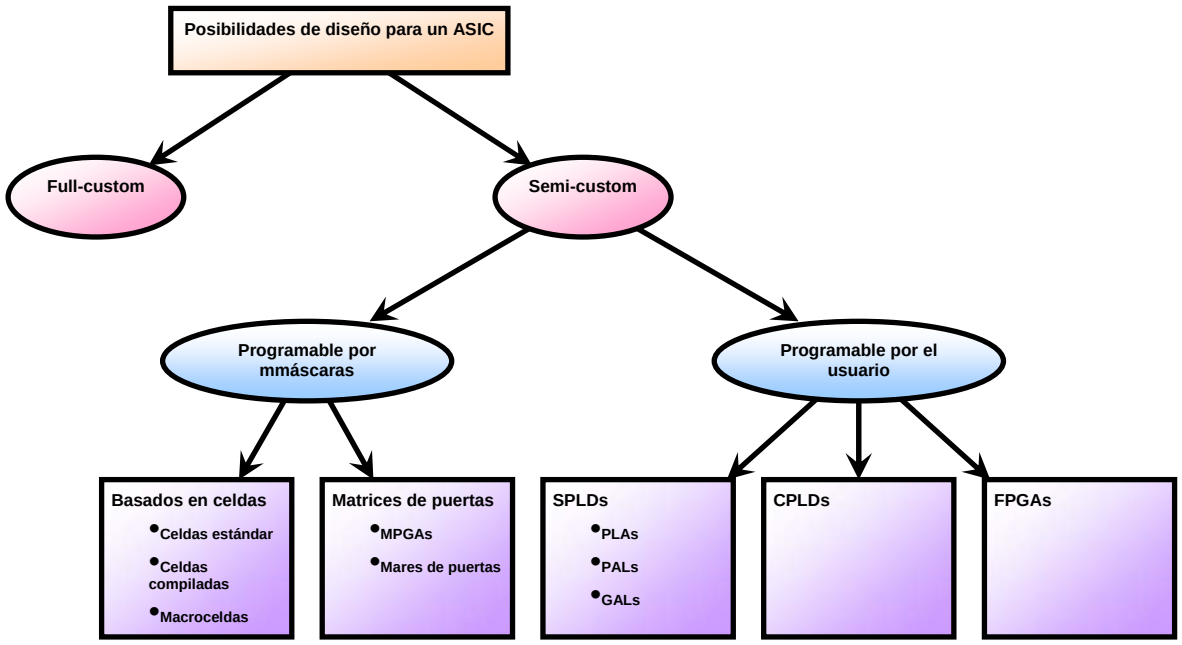


Figura A1.2. Diferentes tipos de dispositivos reconfigurables.

El procesador se encarga de las operaciones que no se pueden ejecutar eficientemente en la lógica reconfigurable como son los accesos a memoria o el control de flujo de datos, mientras que el *hardware* se encarga del núcleo computacional de la aplicación.

Para caracterizar un sistema reconfigurable se usan los siguientes parámetros [CHO02]:

- Granularidad: se refiere al tamaño de los datos usados en la unidad reconfigurable.
- Profundidad de programación: describe el número de programas de configuración o contextos almacenados en la unidad reconfigurable. Un sistema reconfigurable puede ser monocontextual o relativo a múltiples contextos.
- Reconfigurabilidad: se refiere al tipo de reconfiguración que soporta el sistema, a su vez esta propiedad se puede clasificar en :
  - Estática, cuando la reconfiguración interrumpe la ejecución.
  - Dinámica, cuando la reconfiguración se hace en paralelo con la ejecución.
- Interfaz: hace referencia a la presencia de un procesador principal que controla el sistema reconfigurable. Cuando el procesador principal y la unidad reconfigurable están en el mismo chip, se dice que ésta tiene una interfaz local, mientras que cuando no están en el mismo chip se dice que la unidad reconfigurable tiene una interfaz remota.

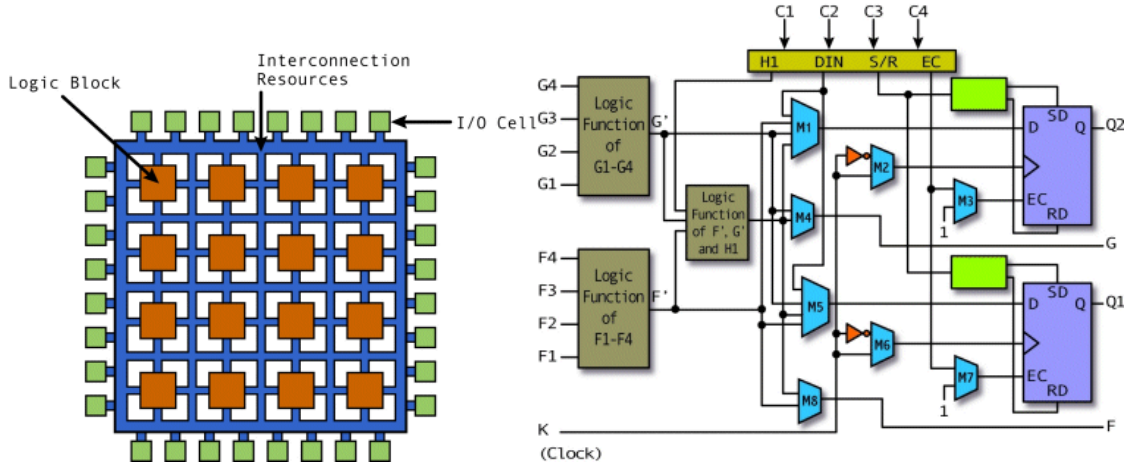


Figura A1.3. Arquitectura típica de una FPGA.

- Modelo de computación: se refiere [HMO03] al estilo que posee la unidad reconfigurable de ejecutar las operaciones. Este modelo puede ser *SIMD*, *MIMD*, *VLIW*, entre otros.

  A continuación se profundiza un poco más en dos de estos criterios: la granularidad y la reconfigurabilidad.

AI.2.1. Granularidad.

Uno de los conceptos que se emplea para describir un sistema reconfigurable es su granularidad donde se ha reseñado previamente que se hace referencia al tamaño de los datos involucrados en las operaciones de la componente reconfigurable.

En los sistemas de grano fino, los elementos de procesamiento de la unidad reconfigurable son normalmente puertas lógicas, *flip-flops* y LUTs *(look-up tables)*; estos sistemas operan a nivel de bit, implementando funciones lógicas. Por otra parte, en los sistemas de grano grueso los elementos de procesamiento de la componente reconfigurable son unidades funcionales completas como, por ejemplo, ALUs y/o multiplicadores que operan sobre palabras de varios bits. Un sistema que combine ambos tipos de granularidad se denomina de grano mixto.

Comúnmente el diseño, en el caso de usar FPGAs, se realiza a nivel de puertas (*flip-flops*, unidades de memoria), haciéndolas bastante útiles para la implementación de funciones complejas a nivel de bit.

Algunos ejemplos de sistemas reconfigurables basados en FPGAs son Splash [GHK91], Chimaera [HFH97] y DISC [WHA95]. Sin embargo, una de las desventajas de las FPGAs es

su bajo rendimiento en operaciones que emplean rutas de datos de granularidad fina como palabras de 8 bits o más. Por lo tanto, varios investigadores han propuesto otros modelos de sistemas reconfigurables de grano grueso enfocados en diferentes aplicaciones, entre los que cabe citar a MorphoSys [MOR00], MATRIX [MIR96], RaPiD [EBE97], REMARC [MIY98] y CHAMELEON/MONTIUM [SHS02]. Otros prototipos de investigación de grano fino, no basados en FPGAs, son DPGA [TCE95] y GARP [HAW97]. Todos estos sistemas poseen características comunes como el uso de celdas idénticas que contienen unidades funcionales similares a la ruta de datos de un procesador, ampliamente conectadas entre sí, además de interfaces de memoria de alta velocidad.

AI.2.2. Reconfiguración dinámica

El sistema reconfigurable reestructurado varias veces para ejecutar diferentes aplicaciones. La reconfiguración es el proceso de recarga de programas de configuración, denominados contextos. Este proceso puede ser estático (cuando la reconfiguración interrumpe la ejecución) o dinámico (cuando la reconfiguración se hace en paralelo con la ejecución).

Los primeros dispositivos reconfigurables eran de un solo contexto y reconfiguración estática, posteriormente, la reconfiguración dinámica hizo relevantes los sistemas de múltiples contextos. Un sistema multi-contexto posee varios planos de configuración, donde uno de estos planos puede activarse en un momento dado, pudiendo el dispositivo conmutar rápidamente entre diferentes planos que se encuentren almacenados en él.

Para reducir el tiempo de latencia de la reconfiguración, estos dispositivos disponen de una memoria interna donde se almacena un conjunto de contextos. De esta manera un dispositivo multi-contexto puede considerarse como un conjunto multiplexado de dispositivos monocontextuales, proporcionando dos planos independientes de configuración.

En un momento dado, uno de estos planos está controlando la ejecución de la unidad reconfigurable, mientras que el otro plano está disponible para que se realice la carga de una configuración diferente. Diferentes arquitecturas reconfigurables de grano grueso son a su vez de múltiples contextos, como es el caso de MorphoSys [MOR00].

AI.3. Aplicaciones para visión artificial, procesado de señal y multimedia:

A lo largo de la evolución de la computación, las aplicaciones de procesamiento digital de señal (DSP) han propiciado el desarrollo de la capacidad de cálculo, especialmente en términos del procesamiento en tiempo real. El paralelismo inherente que existe en muchas funciones DSP las ha convertido en candidatas ideales para ser implementadas en *hardware*.

El creciente interés que han suscitado las aplicaciones multimedia ha desencadenado un gran número de investigaciones en búsqueda de nuevas implementaciones de algoritmos DSP. Así, por ejemplo, muchos procesadores de propósito general ofrecen extensiones especialmente concebidas para procesamiento de audio y vídeo (VIS para Sparc, MVI para Alpha, MMX para IA32, Altivez para PowerP).

También se han diseñado procesadores multimedia y circuitos de aplicación específica para ejecutar diferentes algoritmos DSP y sistemas completos de multimedia (TriMedia de Philips (RS98), Emotion Engine de Sony). Desde la aparición de los sistemas reconfigurables, los problemas DSP han servido como aplicaciones de prueba en el desarrollo de la arquitectura y el *software*. A medida que la capacidad de los sistemas reconfigurables aumentaba, la diversidad y complejidad de sus aplicaciones crecían también. Si en un comienzo las FPGAs se utilizaron especialmente para el filtrado de señales, la aparición de arquitecturas reconfigurables multi-contexto permitió la ejecución de aplicaciones de procesamiento intensivo de datos. Gran parte del trabajo se dirigió al uso efectivo de los abundantes recursos paralelos de las arquitecturas, considerando el elevado paralelismo inherente de datos que poseen las aplicaciones multimedia.

Las aplicaciones objetivo de los sistemas reconfigurables son el procesamiento de imágenes, vídeo, audio y voz, reconocimiento de objetivos y codificación de información. Todas estas aplicaciones se caracterizan por el alto volumen de datos que deben procesar, así como por sus restricciones de tiempo real, que pueden llegar a ser bastante críticas en el caso de aplicaciones interactivas. Por lo tanto, para ocultar la latencia de la memoria se debe realizar una gestión óptima de los datos y las configuraciones.

En términos generales, los núcleos de las aplicaciones DSP y multimedia suelen describirse como secuencias condicionales de macro-tareas o *kernels*. En el caso de las aplicaciones estáticas, éstas tienen la propiedad de que su comportamiento se conoce en

tiempo de compilación (se tiene certeza sobre los *kernels* que se deben ejecutar en cada momento y los datos que deben procesar; en resumen, se conoce de antemano el flujo de datos y control), facilitando la gestión de datos y contextos cuando se compilan para una arquitectura reconfigurable. El hecho de conocer en cada momento qué operaciones se debe ejecutar y sobre cuáles datos, hace que el proceso de compilación obtenga un código bastante eficiente para la aplicación.

Recientemente, el abanico de aplicaciones objetivo de los sistemas reconfigurables se ha extendido a las aplicaciones interactivas. En este caso, el volumen de datos a procesar no es solamente muy elevado sino también variable. De acuerdo con las acciones del usuario, un determinado volumen de datos tendrá que ser procesado en un tiempo que estará acotado estrictamente por el requisito de interactividad.

En las aplicaciones dinámicas interactivas muchos *kernels* se ejecutan condicionalmente, dependiendo de ciertos aspectos que están determinados por acciones externas o por el flujo de datos. La presencia de bloques de decisión entre los *kernels* hace que se desconozca previamente el flujo de datos y control.

Durante la ejecución de una aplicación dinámica, los datos que se van a procesar deben estar disponibles en la memoria interna del circuito tan pronto como sean requeridos para evitar paradas en el procesamiento. Sin embargo, las aplicaciones dinámicas no son deterministas y son eventos externos (acciones del usuario o los mismos datos de entrada) los que determinan el flujo de datos y control de la aplicación. Este grado de incertidumbre hace de la gestión de datos y contextos un problema importante por resolver.

Para evitar las paradas de procesamiento, y por lo tanto, el incremento del tiempo de ejecución, se recomienda el uso de técnicas de precarga (*prefetch*) tanto de datos como de contextos. Por ser las aplicaciones dinámicas muy impredecibles, las técnicas de precarga estarán basadas en estudios estadísticos de la ejecución de la aplicación (*profiling*) en una gran variedad de escenarios.

Mediante el *profiling* se debe extraer información relevante como, por ejemplo, la probabilidad de los saltos condicionales, regularidades de los accesos a memoria, máximo tamaño posible de los datos de entrada de los *kernels*. El objetivo de las técnicas de precarga es suministrar datos la unidad reconfigurable de procesamiento antes de que estos sean solicitados por ella.

De esta manera se oculta la latencia de las transferencias de información desde la memoria externa que puedan impedir satisfacer las restricciones de tiempo real e interactividad de la aplicación. La calidad de la técnica de precarga se medirá por el número de aciertos en la precarga de datos. Así mismo, el volumen de datos que se decida precargar, (combinado con la tasa de aciertos), determinará un costo adicional en energía que tendrá que ser considerado. Las aplicaciones dinámicas pueden hacer parte de sistemas en los que el presupuesto de energía está restringido (el caso de dispositivos móviles, por ejemplo) y, por lo tanto, el diseño se debe enfrentar a partir de un compromiso tiempo frente a energía.

Hasta ahora se han realizado investigaciones importantes en la planificación de datos y contextos para arquitecturas reconfigurables multicontexto en aplicaciones estáticas relacionadas con visión por artificial, concretamente con el mundo de la animación 3D (*Ray tracing* o renderizado de objetos) como los trabajos de [RIV06], [SAN06]. Éstos sirven como punto de partida para lograr la ejecución eficiente de aplicaciones dinámicas, sobre arquitecturas reconfigurables multicontexto desde el punto de vista del tiempo de ejecución enfatizando en mantener el consumo de energía dentro de un margen razonable.

APÉNDICE II

Estudio y mejoras del Algoritmo Multicanal de Gradiente

AII.1. Breve Resumen del Algoritmo

AII.1.1. Introducción.

Este apéndice contiene una descripción del modelo empleado en esta memoria, conviene aclarar que tanto en esta sección, como en el diagrama de flujo perteneciente al siguiente apartado, se reescriben conscientemente algunas ecuaciones importantes correspondientes a capítulos anteriores para evitar volver a buscarlas en la presente memoria y facilitar la comprensión de los conceptos a considerar.

Se piensa inicialmente en un modelo de descrito en dos dimensiones (una dimensión espacial y una temporal).

El modelo se obtiene de la aproximación del gradiente en la que la velocidad se calcula dividiendo la derivada temporal y espacial del brillo de la imagen. Dado que diferenciar parcialmente una imagen filtrada, es equivalente a aplicar un filtro diferenciado parcialmente a la imagen, se puede procesar un modelo de gradiente aplicando pares de filtros a la imagen, siendo uno la derivada temporal y otro la derivada espacial del mismo núcleo de filtros. La velocidad local después se calcula haciendo el cociente de las salidas de los dos filtros.

Llegado a este punto, el principal problema a considerar es que cuando la salida del filtro espacial se anula, la velocidad está indefinida. Sin embargo, es posible añadir medidas adicionales para condicionar el cálculo del promedio de velocidad.

En vez de considerar un sólo núcleo de filtros, se puede extrapolar este concepto a un vector de núcleos de filtros, obteniendo un vector de filtros espacial al hacer la derivada

espacial de cada componente del vector de núcleos de filtros, (idem temporal). Se podrá calcular posteriormente, una estimación de mínimos cuadrados de la velocidad como el cociente de los productos escalares de los vectores X y T (expresión (A2.1)), siendo X el vector de salidas de filtros diferenciados espacialmente y T (idem temporalmente).

$$Velocidad = \frac{X \cdot T}{X \cdot X} \tag{A2.1}$$

En este caso para que el denominador se anule todas las medidas en X deben ser cero también, en cualquier caso si se incrementa el número de medidas, la probabilidad de que se anule el vector X se ve reducida.

No obstante, se plantea el problema de cómo construir el vector de núcleos de filtros antes de la derivación, qué tipos de filtros deben usarse, y cómo deben calibrarse los unos respecto a otros. Una manera de solucionar este problema consiste en utilizar la propuesta planteada por Koenderink y van Doorn [KOE87] en la que la estructura local de la imagen se representa como una expansión truncada de series de Taylor. En esta propuesta, la estructura de la imagen en un área local se representa mediante los datos de salida de una serie de filtros aplicados a un punto de la imagen.

Estos filtros se generan a partir de un solo núcleo de filtros denominado *Blur Kernel* que consiste en una función Gaussiana en espacio y en el logaritmo del tiempo incrementando progresivamente el orden de los operadores diferenciales espaciales y temporales que se aplican a este núcleo.

A partir del desarrollo en serie de Taylor, se puede estimar el brillo de la imagen a cierta distancia con respecto al punto desde el que se toman las medidas. Los pesos adjuntos a las diversas derivadas dependen de la dirección y longitud del vector que se aproxima al punto en el que se toman las medidas y el punto en el que se desea estimar el brillo de la imagen. Por tanto, para cualquier punto de la imagen será posible construir un vector de funciones de filtros calibrados que servirá como el vector inicial del núcleo de filtros. Consiguiéndose de esta manera una representación de la estructura espacio-temporal de la secuencia, el cálculo de la velocidad integrará los puntos sobre una región en la que se aplica el banco de filtros.

Se generaliza a partir de ahora, todo el proceso anterior para generar un modelo tridimensional y se continúa los pasos asumiendo este último paso. Para generar el vector de núcleos de filtro a partir de un solo núcleo, se ha diferenciado el núcleo en orden creciente de

x,*y*,*t*. Igual que en la versión de dos dimensiones del modelo, el núcleo de filtro es una Gaussiana en el espacio y en el logaritmo del tiempo, según la ecuación A2.2.

$$K(r,t) = \frac{1}{4\pi\sigma} e^{-\frac{r^2}{4\sigma}} \frac{1}{\sqrt{\pi\tau\alpha}\ e^{r^2/4}} e^{-\left(\frac{\ln\left(\frac{t}{\alpha}\right)}{\tau}\right)_2} \tag{A2.2}$$

En la versión de tres dimensiones del modelo, el fotograma de referencia (los ejes *x* e *y* definen la dirección local de la diferenciación espacial) se rota a través de un número de orientaciones con respecto a la imagen de entrada. Usualmente se emplean 24 orientaciones (barriendo el espacio cada 15º). Para cada orientación se crean 3 vectores de filtros, mediante derivadas del núcleo del vector de filtros con respecto *x*, *y*, *t*. conduciendo este proceso a tener una población de filtros sintonizados para distintas orientaciones y frecuencias espaciales que pueden mostrar propiedades temporales tanto transitorias como estacionarias.

A partir de las medidas resultantes de aplicar los tres vectores de funciones de filtro a la imagen, se calculan las siguientes cuatro medidas relacionadas con la velocidad: velocidad, velocidad ortogonal, velocidad inversa y velocidad inversa ortogonal, definidas según las expresiones A2.3 a A2.6.

$$velocidad = \frac{X \cdot T}{X \cdot X} \cos^2\theta = \frac{X \cdot T}{X \cdot X}\left(1 + \left(\frac{X \cdot Y}{X \cdot X}\right)^2\right)^{-1} \tag{A2.3}$$

$$velocidad\ ortog. = \frac{Y \cdot T}{Y \cdot Y} sen^2\theta = \frac{Y \cdot T}{Y \cdot Y}\left(1 + \left(\frac{X \cdot Y}{Y \cdot Y}\right)^2\right)^{-1} \tag{A2.4}$$

$$velocidad\ inversa = \frac{X \cdot T}{T \cdot T} \tag{A2.5}$$

$$velocidad\ inversa\ ortog. = \frac{Y \cdot T}{T \cdot T} \tag{A2.6}$$

Donde X es el vector de salidas del filtro de los filtros diferenciados espacialmente, (análogamente con Y y T).

Las medidas ·de velocidad *(X T/X X)* y de velocidad ortogonal *(Y T/Y Y)* no son válidas cuando no hay variación en *x* e *y*, respectivamente. Para evitar que esto altere la estimación final de la velocidad, tanto la velocidad como la velocidad ortogonal están condicionadas mediante medidas del ángulo existente entre la estructura de la imagen y el fotograma de referencia θ.

De la ecuación (A2.3) se desprende que mientras *X X* disminuye, el término correspondiente a $(X\ Y/(X\ X)^2$ aumenta más rápidamente que la componente de velocidad, ocasionando, por tanto que el producto sea cero a medida que (*X T/X X)* se aproxima al infinito. Aún así, las medidas de velocidad inversa tienden al infinito cuando la derivada temporal tiende a cero.

Dado que las cuatro medidas relacionadas con la velocidad se calculan a partir de distintas combinaciones de datos de salida de los filtros, se puede considerar que todas ellas son estimaciones distintas relacionadas con la velocidad local. La inclusión de medidas adicionales de la imagen con el propósito de aumentar la robustez es uno de los criterios fundamentales de diseño del modelo.

Utilizando cada una de las velocidades definidas en las ecuaciones A2.3 a A2.6 y ordenándolas según las orientaciones del fotograma de referencia, se pueden definir los vectores velocidad directa (a su vez formada por la velocidad directa paralela y directa ortogonal) y velocidad inversa (a su vez formada por la velocidad inversa paralela e inversa ortogonal), según las expresiones A2.7 a A2.10.

$$\hat{s}(\theta) = (\hat{s}_{\amalg}, \hat{s}_{\perp}) \tag{A2.7}$$

$$\hat{s}(\theta) = \sqrt{\frac{2}{m}}\left[\frac{x_\theta \cdot t_\theta}{x_\theta \cdot x_\theta}\left(1+\left(\frac{x_\theta \cdot y_\theta}{x_\theta \cdot x_\theta}\right)^2\right)^{-1}, \frac{y_\theta \cdot t_\theta}{y_\theta \cdot y_\theta}\left(1+\left(\frac{x_\theta \cdot y_\theta}{y_\theta \cdot y_\theta}\right)^2\right)^{-1}\right] \tag{A2.8}$$

$$\breve{s}(\theta) = (\breve{s}_{\amalg}, \breve{s}_{\perp}) \tag{A2.9}$$

$$\breve{s}(\theta) = \sqrt{\frac{2}{m}}\left[\frac{x_\theta \cdot t_\theta}{t_\theta \cdot t_\theta}, \frac{y_\theta \cdot t_\theta}{t_\theta \cdot t_\theta}\right] \tag{A2.10}$$

La estimación del módulo de la velocidad final, se calcula como la raíz cuadrada del determinante definido en la ecuación A2.11.

$$S^2 = \begin{bmatrix} \widehat{s}_{\text{II}} F_{\text{II}} & \widehat{s}_{\text{II}} F_{\perp} \\ \widehat{s}_{\perp} F_{\text{II}} & \widehat{s}_{\perp} F_{\perp} \\ \hline \widehat{s}_{\text{II}} \breve{s}_{\text{II}} & \widehat{s}_{\text{II}} \breve{s}_{\text{II}} \\ \widehat{s}_{\perp} \breve{s}_{\perp} & \widehat{s}_{\perp} \breve{s}_{\perp} \end{bmatrix} \tag{A2.11}$$

Donde $\bar{F}(\theta) = ((F_{\|}(\theta), F_{\perp}(\theta)) = \sqrt{2/m}\left[\cos(\theta), \sin(\theta)\right]$ es el vector que contiene los cosenos y senos de los ángulos de rotación del fotograma de referencia.

En el caso del movimiento de un diseño simple como, por ejemplo una rejilla sinusoidal en movimiento, el denominador de la ecuación A2.11 adquiere el valor de la unidad. Las medidas de la velocidad y de la velocidad ortogonal varían sinusoidalmente con el ángulo del fotograma de referencia, con la velocidad ortogonal retrasada un cuarto de ciclo respecto a la velocidad. El numerador en la ecuación A2.11 opera tomando medidas de la amplitud de la distribución de las medidas de velocidad, la cuales se combinan con la velocidad y con la velocidad ortogonal. El denominador (y sus medidas de velocidad inversa) se incluyen para estabilizar la estimación final de velocidad. Esto por ejemplo, es particularmente importante cuando el movimiento ocurre en presencia de patrones estáticos.

Por otra parte, se calcula la dirección de movimiento final a través de la medida de la fase que se combina de forma cruzada con las 4 funciones de velocidad, según la ecuación A2.12.

$$\textit{Dirección} = \arctan\left(\frac{(\breve{s}_{\text{II}} + \widehat{s}_{\text{II}})F_{\perp} + (\breve{s}_{\perp} + \widehat{s}_{\perp})F_{\text{II}}}{(\breve{s}_{\text{II}} + \widehat{s}_{\text{II}})F_{\text{II}} - (\breve{s}_{\perp} + \widehat{s}_{\perp})F_{\perp}}\right) \tag{A2.12}$$

Las medidas de velocidad y velocidad inversa son complementarias y antagónicas, proporcionando al algoritmo una robustez necesaria en situaciones donde las señales son débiles y existe una gran interacción con ruido de cualquier tipo.

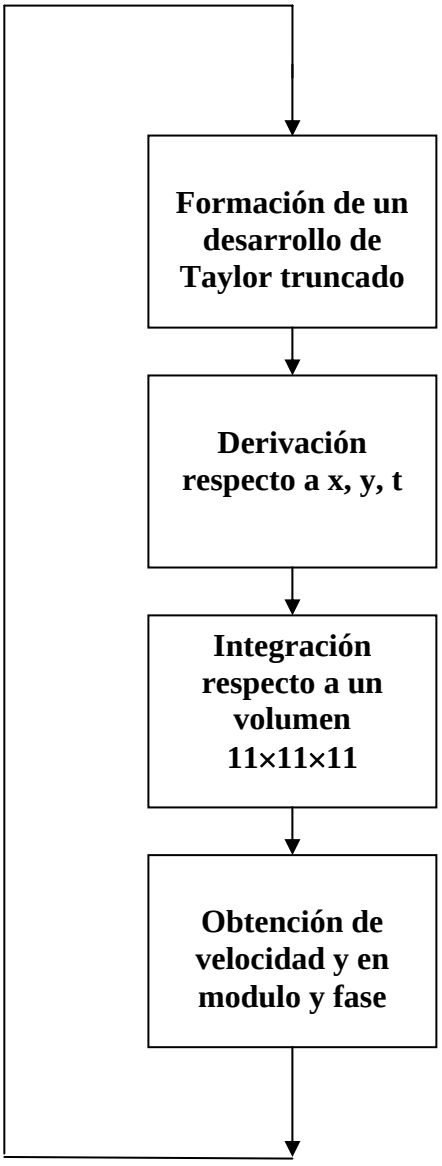
AII.2. Diagrama de flujo del Algoritmo.

A continuación se muestra un organigrama que clasifica jerárquicamente el algoritmo del modelo, con vista a su implementación:

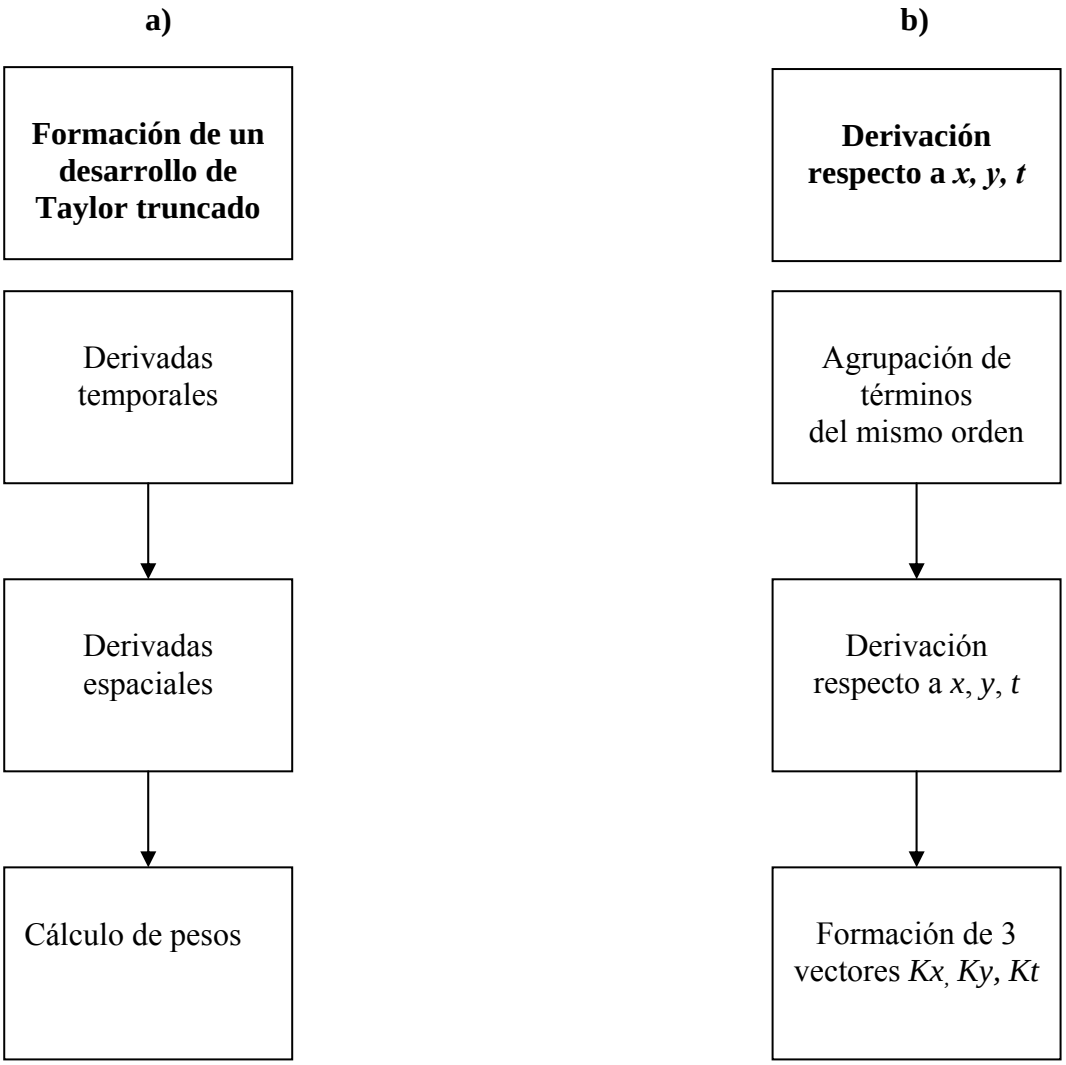
AII.2.1. Nivel I.

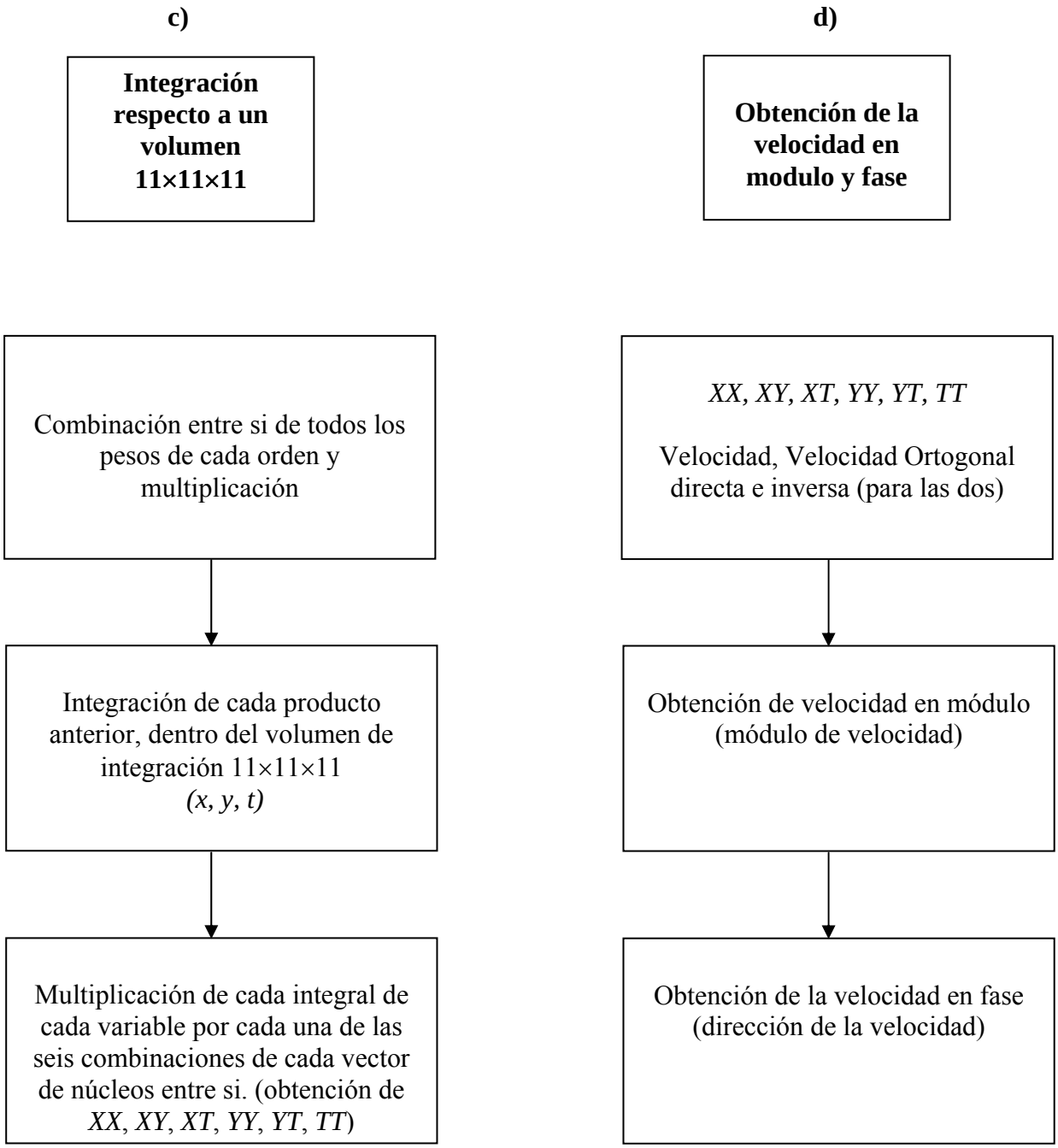
Para cada fotograma de la secuencia.

Para cada orientación



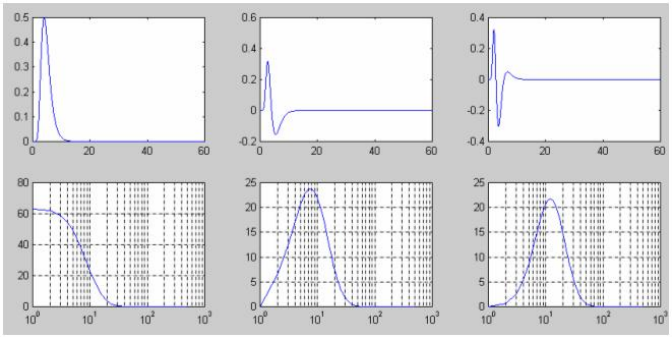
AII.2.2. Nivel II.




c)
Integración respecto a un volumen 11×11×11
Combinación entre si de todos los pesos de cada orden y multiplicación
Integración de cada producto anterior, dentro del volumen de integración 11×11×11
(x, y, t)
Multiplicación de cada integral de cada variable por cada una de las seis combinaciones de cada vector de núcleos entre si. (obtención de XX, XY, XT, YY, YT, TT)
d)
Obtención de la velocidad en modulo y fase
XX, XY, XT, YY, YT, TT
Velocidad, Velocidad Ortogonal directa e inversa (para las dos)
Obtención de velocidad en módulo (módulo de velocidad)
Obtención de la velocidad en fase (dirección de la velocidad)

AII.2.3. Nivel III.

(a.1)
Derivadas
temporales



Para cada uno de los 3 filtros temporales

Convolución del fotograma a lo largo del eje temporal.

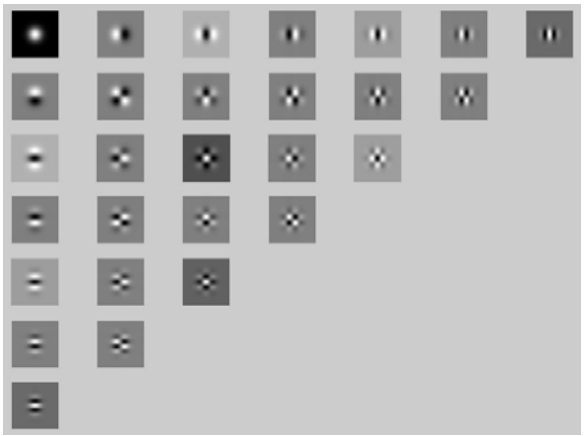
(a.2)
Derivadas
espaciales

(cada operador es separable)

Para cada uno de los 3 fotog. derivados temporalmente

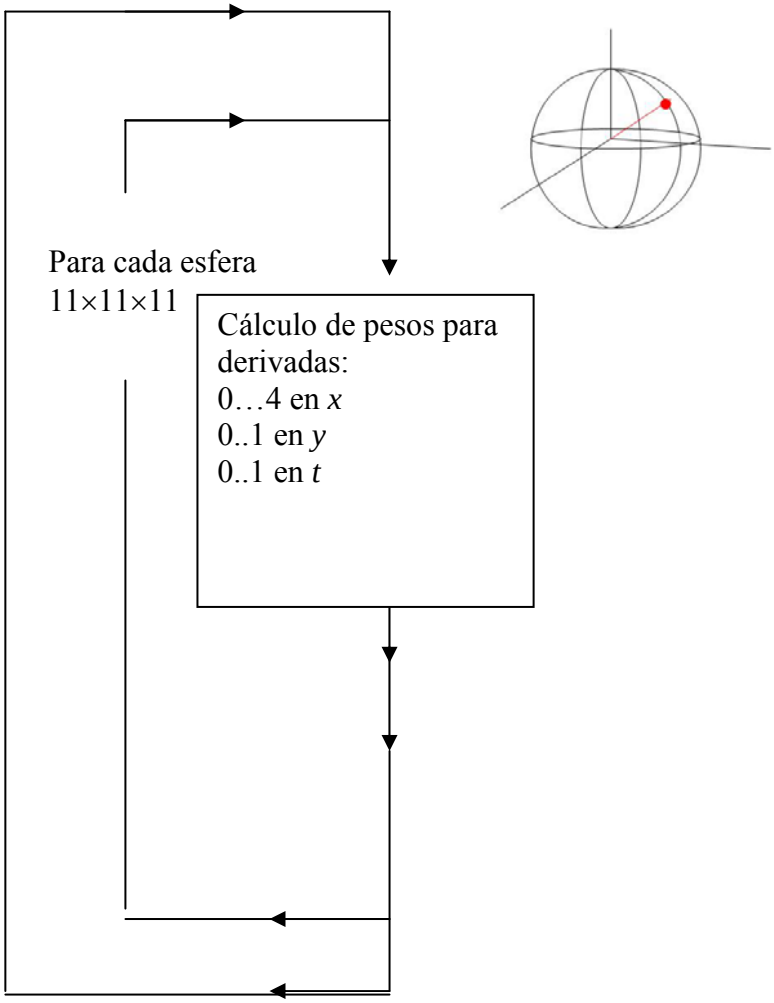
Convolución por columnas

Convolución por filas



(a.3)
Pesos

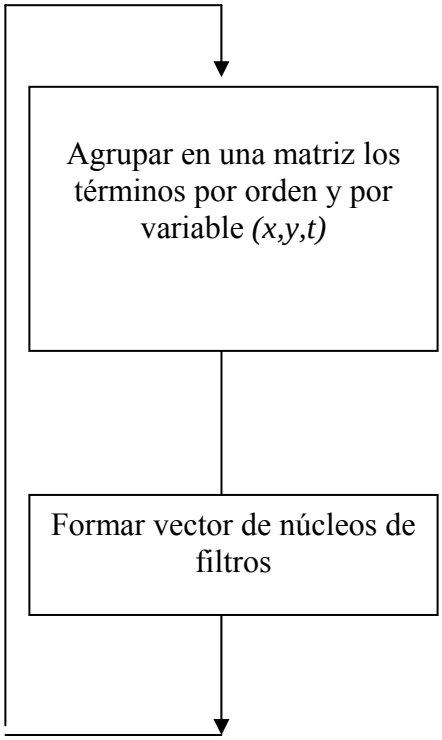
Para cada uno de los 3×28 fotogramas derivados espacio-temp.



$$I(x+p,y+q,t+r)=\sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n}\frac{p^i q^i r^k}{i!\,j!k!}\frac{\partial^n}{\partial x^i \partial y^j \partial t^k}I(x,y,t)$$

(b.1)
Agrupación de términos del mismo orden

Para cada
Desarrollo truncado
De Taylor



$$= \begin{bmatrix} \frac{\partial\{Orden_0(x,y,t)\}}{\partial x} & \frac{\partial\{Orden_0(x,y,t)\}}{\partial y} & \frac{\partial\{Orden_0(x,y,t)\}}{\partial t} \\ \frac{\partial\{Orden_1(x,y,t)\}}{\partial x} & \frac{\partial\{Orden_1(x,y,t)\}}{\partial y} & \frac{\partial\{Orden_1(x,y,t)\}}{\partial t} \\ \frac{\partial\{Orden_2(x,y,t)\}}{\partial x} & \frac{\partial\{Orden_2(x,y,t)\}}{\partial y} & \frac{\partial\{Orden_2(x,y,t)\}}{\partial t} \\ \frac{\partial\{Orden_3(x,y,t)\}}{\partial x} & \frac{\partial\{Orden_3(x,y,t)\}}{\partial y} & \frac{\partial\{Orden_3(x,y,t)\}}{\partial t} \\ \frac{\partial\{Orden_4(x,y,t)\}}{\partial x} & \frac{\partial\{Orden_4(x,y,t)\}}{\partial y} & \frac{\partial\{Orden_4(x,y,t)\}}{\partial t} \\ \frac{\partial\{Orden_3(x,y,t)\}}{\partial x} & \frac{\partial\{Orden_3(x,y,t)\}}{\partial y} & \frac{\partial\{Orden_3(x,y,t)\}}{\partial t} \\ \frac{\partial\{Orden_6(x,y,t)\}}{\partial x} & \frac{\partial\{Orden_6(x,y,t)\}}{\partial y} & \frac{\partial\{Orden_6(x,y,t)\}}{\partial t} \end{bmatrix}$$

$$Vector_{nucleo\_de\_filtros}(x,y,t) = \{I(x,y,t),...,,...,\frac{\partial^6(I(x,y,t))}{\partial y \partial t \partial x^4},\}$$

(b.2)
Derivación
respecto a *x*, *y*, *t*.

Para cada
vector de
núcleos de
de filtro

x 0…4
y 0…1
t 0…1

Uso de derivadas:
x 0…5
y 0…2
t 0…2

(b.3)
Formación de 3
vectores núcleos
de filtros con esas
3 derivadas
Kx, Ky, Kt

Para cada
derivada siguiente:
x 0…5
y 0…2
t 0…2

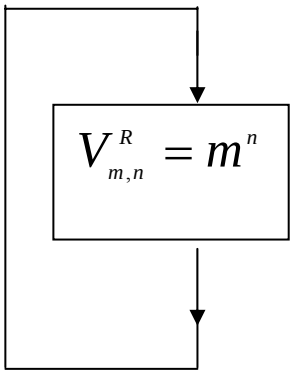
Orden de esas 3 derivadas en 3
vectores
Kx, Ky, Kt

$$Kx = \frac{\partial\{\sum_{i=0}^{6} Orden_i(x,y,t)\}}{\partial x}$$

$$Ky = \frac{\partial\{\sum_{i=0}^{6} Orden_i(x,y,t)\}}{\partial y}$$

$$Kt = \frac{\partial\{\sum_{i=0}^{6} Orden_i(x,y,t)\}}{\partial t}$$

(c.1)
Combinación r entre sí de todos los pesos de cada orden y multiplicarlos

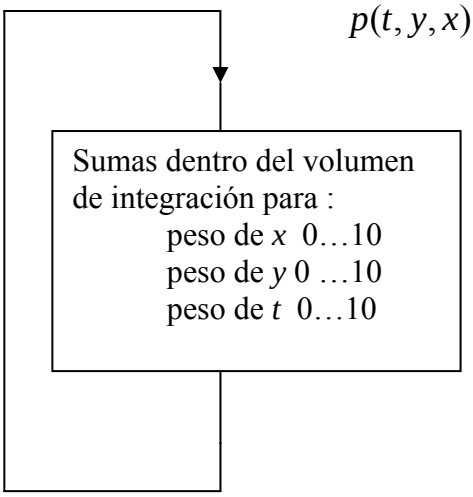


Para cada pesos calculado antes:

(c.2)
Integración de cada producto anterior, dentro del volumen de integración 11×11×11 *(x, y, t)*

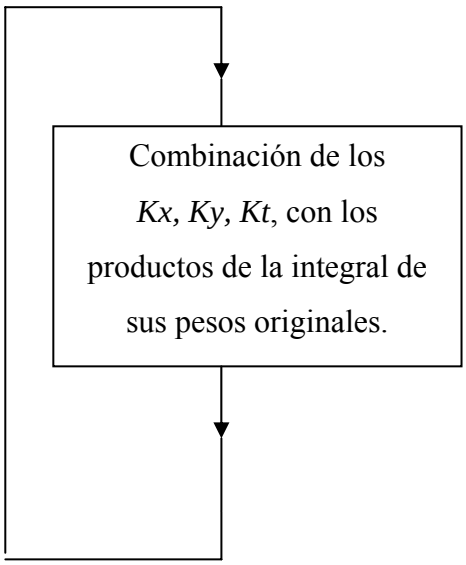
$$p(t,y,x) = \sum_{r=0}^{10} \sum_{q=0}^{10} \sum_{p=0}^{10} pesos(p,q,r,t,y,x)$$

Para cada uno de los 68 términos combinados



(c.3)
Multiplicación de cada integral de cada variable por cada una de las seis combinaciones de cada vector de núcleos entre si. (obtención de *XX*, *XY*, *XT*, *YY*, *YT*, *TT*)

Para cada uno de los términos integrados



$$XT = \sum_{k=o}^{orden} \sum_{i=0}^{m^n} \sum_{j=0}^{m^n} Kx(i)Kt(j)matriz(k,i,j)$$

$$YT = \sum_{k=o}^{orden} \sum_{i=0}^{m^n} \sum_{j=0}^{m^n} Ky(i)Kt(j)matriz(k,i,j)$$

$$TT = \sum_{k=o}^{orden} \sum_{i=0}^{m^n} \sum_{j=0}^{m^n} Kt(i)Kt(j)matriz(k,i,j)$$

$$XX = \sum_{k=o}^{orden} \sum_{i=0}^{m^n} \sum_{j=0}^{m^n} Kx(i)Kx(j)matriz(k,i,j)$$

$$YY = \sum_{k=o}^{orden} \sum_{i=0}^{m^n} \sum_{j=0}^{m^n} Ky(i)Ky(j)matriz(k,i,j)$$

$$XY = \sum_{k=o}^{orden} \sum_{i=0}^{m^n} \sum_{j=0}^{m^n} Kx(i)Ky(j)matriz(k,i,j)$$

(d.1)
XX, XY, XT, YY, YT,TT
Velocidad, Velocidad Ortogonal directa e inversa

$$\hat{s}(\theta)=\sqrt{\frac{2}{m}}\left[\frac{x_\theta\cdot t_\theta}{x_\theta\cdot x_\theta}\left(1+\left(\frac{x_\theta\cdot y_\theta}{x_\theta\cdot x_\theta}\right)^2\right)^{-1},\frac{y_\theta\cdot t_\theta}{y_\theta\cdot y_\theta}\left(1+\left(\frac{x_\theta\cdot y_\theta}{y_\theta\cdot y_\theta}\right)^2\right)^{-1}\right]$$

$$\hat{s}(\theta)=(\hat{s}_{\text{II}},\hat{s}_{\perp})$$
$$\breve{s}(\theta)=(\breve{s}_{\text{II}},\breve{s}_{\perp})$$

$$\breve{s}(\theta)=\sqrt{\frac{2}{m}}\left[\frac{x_\theta\cdot t_\theta}{t_\theta\cdot t_\theta},\frac{y_\theta\cdot t_\theta}{t_\theta\cdot t_\theta}\right]$$

(d.2)
Obtención de la velocidad en módulo (módulo de velocidad)

$$\breve{s}(\theta)=(\breve{s}_{\text{II}},\breve{s}_{\perp})$$
$$\hat{s}(\theta)=(\hat{s}_{\text{II}},\hat{s}_{\perp})$$

S

$$S=\begin{bmatrix}\hat{s}_{\text{II}}F_{\text{II}} & \hat{s}_{\text{II}}F_{\perp}\\ \hat{s}_{\perp}F_{\text{II}} & \hat{s}_{\perp}F_{\perp}\\ \hline \hat{s}_{\text{II}}\breve{s}_{\text{II}} & \hat{s}_{\text{II}}\breve{s}_{\text{II}}\\ \hat{s}_{\perp}\breve{s}_{\perp} & \hat{s}_{\perp}\breve{s}_{\perp}\end{bmatrix}^{1/2}$$

$$\bar{F}(\theta)=((F_{\|}(\theta),F_{\perp}(\theta))=\sqrt{2/m}\left[\cos(\theta),\sin(\theta)\right]$$

(d.3)
Obtención de la velocidad en fase (dirección de velocidad)

$$\bar{F}(\theta)=((F_{\|}(\theta),F_{\perp}(\theta))=\sqrt{2/m}\left[\cos(\theta),\sin(\theta)\right]$$
$$\breve{s}(\theta)=(\breve{s}_{\text{II}},\breve{s}_{\perp})$$
$$\hat{s}(\theta)=(\hat{s}_{\text{II}},\hat{s}_{\perp})$$

Dirección

$$\textit{Dirección}=\arctan\left(\frac{(\breve{s}_{\text{II}}+\hat{s}_{\text{II}})F_{\perp}+(\breve{s}_{\perp}+\hat{s}_{\perp})F_{\text{II}}}{(\breve{s}_{\text{II}}+\hat{s}_{\text{II}})F_{\text{II}}-(\breve{s}_{\perp}+\hat{s}_{\perp})F_{\perp}}\right)$$

AII.3. Examen del modelo con justificación de su robustez y mejoras del algoritmo

En este apartado se realiza un examen del modelo multicanal de gradiente (McGM) presentado en esta memoria, profundizándose en:

- Análisis de los parámetros teóricos más importantes de que consta el modelo.
- Estudio teórico de la robustez del modelo frente a ruido estático.
- Estudio de una posible degradación del modelo, para degenerar en un modelo de gradiente convencional.
- Optimización del algoritmo atendiendo a la simetría de éste.

AII.3.1. Examen del modelo.

Se va a proceder a un análisis de los parámetros más importantes de que consta el modelo, seguido de un estudio teórico de la robustez del mismo frente a ruido estático, para finalizar con un estudio de una posible degradación de éste, mediante diversas simplificaciones que lo que daría lugar a llegar a convertir el modelo en un modelo de gradiente convencional.

AII.3.1.1. Parámetros del modelo:

Se requieren tres parámetros espacio-temporales del *blur-kernel*, el cual se deriva para generar todos los filtros lineales que se usan según la ecuación A2.2. Por lo tanto la relación entre los filtros tiene una restricción bastante alta, reflejándose este hecho en la forma de los filtros temporales medidos psicofísicamente según los trabajos de Hess y Snowden [HES92a][HES92b]. Estos tres parámetros del núcleo *(*denominados parámetros ***primero (σ)***, ***segundo (α)*** y ***tercero (τ))*** realizan un mapa de la escala espacio-temporal del modelo a través de la escala física de espacio-tiempo. Han sido fijados por Johnston [JOH95] a los valores $\sigma = 1.5$, $\alpha = 10$ y $\tau = 0.2$, mostrando una forma de calibrar el modelo, para proporcionar medidas de velocidad en grados/seg (1 º = 128 pixels , 1 seg = 128 pixels).

Un parámetro adicional ***(cuarto)*** asigna el número de derivadas espaciales en la dirección primaria (en este caso son cinco). Se podría usar otro número diferente pero no se ganaría mucha más información a partir de un determinado punto ya que las derivadas de las Gaussianas no forman un conjunto ortogonal y las respuestas a órdenes superiores estarían

cada vez más correlacionadas. Aun más, una objeción a usar derivadas de ordenes superiores estaría en que se incrementaría la sensibilidad del modelo al ruido (las derivadas actúan como filtros de paso alta y se ajustan progresivamente a frecuencias más altas a medida que el orden de diferenciación aumenta). Sin embargo, cuando se considera flujo óptico, también se realiza un filtrado temporal, cuando se considera el efecto de un filtro paso baja temporal, se reduce el posible ruido de forma considerable.

El ***quinto*** parámetro del modelo define el número de direcciones de orientación muestreadas, en este caso 24. El máximo número de orientaciones estaría ilimitado y el mínimo serían 2 (no necesariamente ortogonales).

El ***sexto*** parámetro define la extensión de esas zonas de integración, en este caso 11×11×11. Estas zonas deben ser como máximo el tamaño de los filtros espaciales. Si se supera éste, el desarrollo de Taylor podría ser muy costoso o irrealizable. La elección de esta zona también tiene el efecto de alterar la importancia relativa de las medidas de las derivadas, un hecho que se puede usar a conveniencia para ajustar el modelo al ruido presente en el escenario de la medida. (todos estos parámetros vienen representados en el capítulo IV, 4.2.3).

AII.3.1.2. Análisis de la invarianza frente al patrón estático:

Sea $k(x, y, t) = f(x, y, t) + g(x, y, t)$, siendo $f(x, y, t)$ un patrón dinámico y $g(x, y, t)$ un patrón estático. En este caso, dado que la diferenciación y el producto escalar son operaciones lineales, partiendo de la expresión (3.47) (*v*.3.2.3) se tendría que:

$$J^T J = \begin{bmatrix} K_x K_x & K_x K_y & K_x K_t \\ K_y K_x & K_y K_y & K_y K_t \\ K_t K_x & K_t K_y & K_t K_t \end{bmatrix} =$$

$$= \begin{bmatrix} f_x f_x & f_x f_y & f_x f_t \\ f_y f_x & f_y f_y & f_y f_t \\ f_t f_x & f_t f_y & f_t f_t \end{bmatrix} + \begin{bmatrix} g_x f_x & g_x f_y & g_x f_t \\ g_y f_x & g_y f_y & g_y f_t \\ 0 & 0 & 0 \end{bmatrix} + \tag{A2.13}$$

$$+ \begin{bmatrix} f_x g_x & f_x g_y & 0 \\ f_y g_x & f_y g_y & 0 \\ f_t g_x & f_t g_y & 0 \end{bmatrix} + \begin{bmatrix} g_x g_x & g_x g_y & 0 \\ g_y g_x & g_y g_y & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

donde f_x, por ejemplo, es la derivada parcial con respecto a x del vector derivado del desarrollo de de Taylor de la función $f(x, y, t)$ y $g_t = 0$ ya que g es un patrón estático.

Como la integración es una operación lineal, se puede hacer la descomposición expuesta en la ecuación A2.13:

La integral de los términos de la primera matriz de la ecuación A2.13 (procesada sumando los índices *p*, *q*, y *r* como se ha explicado en esta memoria) es igual a la suma de integrales de las 4 matrices. Esta descomposición muestra que los valores de velocidad inversa (*v*.A2.10) dependen casi en su totalidad del patrón en movimiento, *f(x, y, t)*. Los dos términos de la 2ª matriz descompuesta (f_t, g_x, f_t, g_y) se añaden a los numeradores de los cocientes de velocidad inversa. Para que esos términos se anulen, los vectores no deberían estar correlacionados o estar correlacionados de forma positiva y negativa con la misma probabilidad sobre la zona espacio-temporal de integración.

En vista de que los productos con signo opuesto se cancelarán en la integración, la suma de esos términos sobre el volumen espacio-temporal será pequeña. El denominador $k_t\, k_t$ de los cocientes de la ecuación es totalmente independiente del patrón estático. De esta forma, la velocidad inversa que se calcula aquí, proporciona una medida robusta incluso en presencia de un patrón estático.

La velocidad directa también tiene productos internos cruzados donde intervienen funciones estáticas y dinámicas. Estos productos serán probablemente pequeños, pero les afectan los términos (del patrón estático) de la cuarta matriz. Sustituyendo en la cuarta matriz, se obtiene la ecuación A2.14:

$$\widehat{s} = \frac{\int_R (f_x f_t + g_x f_t)}{\int_R (f_x f_x + 2 g_x f_x + g_x g_x)} \left(1 + \left(\frac{\int_R k_x k_y}{\int_R k_x k_x} \right)^2 \right)^{-1} \qquad \text{(A2.14)}$$

El término $g_x g_x$ reduce bruscamente la velocidad directa en una cantidad que es proporcional al contraste estático. Este análisis ayuda a comprender porqué el modelo es insensible al patrón estático, ya que en este modelo la velocidad se calcula como un cociente de determinantes. Ambos determinantes pueden ser interpretados como los productos de las áreas abarcadas por 2 pares de vectores, tal y como ilustra la figura (A.2.1).

Los vectores son *m*-dimensionales (siendo *m* el número de orientaciones). Dado que los vectores de velocidad directa $\widehat{s}(\theta) = (\widehat{s}_{\amalg}, \widehat{s}_{\perp})$ contribuyen tanto al numerador, como al denominador, la medida se determina de forma primaria por la velocidad inversa que es relativamente invariante con respecto al patrón estático.

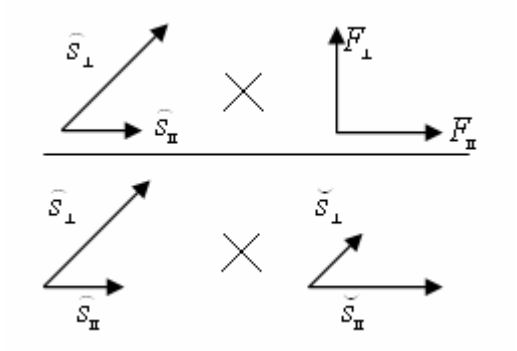


Figura A2.1. Cocientes de productos vectoriales de las áreas de vectores ortogonales y paralelos.

El cálculo de la velocidad inversa es muy importante para la estimación final de la velocidad (la velocidad inversa es invariable respecto a la adición del patrón estático), sin embargo, esta invarianza del patrón estático depende de la integración sobre el volumen espacio-temporal. En el caso del movimiento rígido de un patrón que varíe de forma suave, el denominador de la ecuación A2.11 se iguala a la unidad y la velocidad viene dada por el numerador.

Se ha demostrado que es posible quitar la influencia del patrón estático en la última etapa de procesado de movimiento. Las dos etapas adicionales a la combinación de filtros lineales, la integración sobre un volumen espacio temporal y la combinación de funciones de velocidad y velocidad inversa, proporcionan una forma de reducir la influencia del patrón estático en el análisis del movimiento. Una degradación de esos procesos dará lugar a un decremento del rendimiento del modelo en presencia de ruido estático.

AII.3.1.3. Degradado del modelo:

El modelo puede ser degradado para hacer el algoritmo equivalente a un modelo estándar de gradiente espacio-temporal, que puede adoptar a su vez la forma de un modelo de energía o de correlación de Reichardt [REI61].

Existen múltiples formas de degradar el modelo, según lo que se cambie se obtendrá unos resultados u otros. Estudiando el efecto de la supresión de la etapa de integración y del denominador de la ecuación (A2.11) manteniendo el resto de la estructura constante.

Si se elimina la etapa de integración, el modelo se comporta igual que un modelo de gradiente espacio-temporal en el cual las diferenciales de varios órdenes se combinan con el mismo peso.

Esto hace que la velocidad final de la estimación sea sensible al patrón estático, incluso usando los mismo filtros espacio temporales que se han usado en toda esta memoria.

En otro orden de degradación, si se usan sólo los resultados del numerador en el cálculo del módulo de velocidad, se pierde independencia frente al patrón estático aunque menos que si se elimina la etapa de integración. Otra manera de degradar el modelo consiste en decrementar el número de direcciones muestreadas aunque esta degradación depende de la naturaleza del estímulo, no siendo por tanto fundamental.[JOH98]

Se puede concluir que las etapas críticas para tener independencia frente al patrón estático son las de integración y el cociente.

AII.3.2. Algunas optimizaciones del algoritmo

En este apartado se muestran algunos procedimientos para optimizar el algoritmo y hacerlo más eficiente, sin perder ningún tipo de rendimiento, en función de la simetría de las operaciones.

- Como todos los filtros empleados en el algoritmo son simétricos o antisimétricos, sólo se necesita filtrar en un rango de orientaciones de 0º a 180° antes de repetir los resultados, pero con un factor negativo. Las medidas de velocidad son proyectadas sobre una onda seno para extraer la dirección, pero gracias a la simetría se puede usar la mitad de las orientaciones proyectando sobre la mitad del sinusoide. Esto requeriría generar sólo la mitad de respuestas orientadas, obteniéndose la mitad de las orientaciones.
- Incluso más, otra optimización posible es calcular los filtros orientados sólo en un rango de 0º a 90º, pues los otros 3 cuadrantes pueden ser generados mediante reflexiones y/o inversiones de las respuestas de los filtros, para el caso usual de 24 orientaciones, se tendrían que calcular 6 solamente. (de 15º a 90º).pero es necesario plantear adecuadamente estas operaciones geométricas
- Otra simplificación se puede realizar sobre el cálculo de esta integral de la expresión (A2.15). La zona simétrica de integración tiene un efecto crucial en la simplificación del cálculo de los productos de términos cruzados.

$$M = \int_e^f \int_c^d \int_a^b J^T J dp dq dr = \begin{bmatrix} XX & XY & XT \\ YX & YY & YT \\ TX & TY & TT \end{bmatrix} \tag{A2.15}$$

$$\Omega = \{a \le p \le b, c \le q \le d, e \le r \le f\}$$

$$\forall \quad a,b,c,d,e,f \in R$$

Considerando los primeros términos de los desarrollos de Taylor diferenciados, éstos corresponden a las ecuaciones (A2.16):

$$\begin{aligned} k_{1,y} &= pI_{xy}(x,y,t) + qI_{2y}(x,y,t) + rI_{ty}(x,y,t) \\ k_{1,x} &= pI_{2x}(x,y,t) + qI_{yx}(x,y,t) + rI_{tx}(x,y,t) \end{aligned} \tag{A2.16}$$

La siguiente etapa del algoritmo requiere que los vectores k_x y k_y sean multiplicados entre si. Para este ejemplo concreto, se tendrían que hacer nueve productos cruzados, según la ecuación (A2.17)

$$V_{m,n}^R = m^n \tag{A2.17}$$

Hay que recordar que m era el número de elementos del orden considerado en el desarrollo de Taylor (en este caso 3) y n el número de elementos que tenía cada término producto (en este caso siempre 2, porque se están multiplicando escalarmente dos vectores). Sin embargo si las zonas de integración son simétricas alrededor de cero (como ocurre en este caso), todos los términos cruzados (como por ejemplo $pqI_{2x}I_{2y}$) se hacen nulos al integrar y sólo se tienen tres términos diferentes de cero:

$$k_{1,x}k_{1,y} = p^2 I_{2x}I_{xy} + q^2 I_{yx}I_{2y} + r^2 I_{tx}I_{ty} \tag{A2.18}$$

La integración sobre la expresión (A2.16) se realiza sobre una esfera de ejes x, y, t y de diámetro 11 puntos. Por lo tanto una integración a través de una zona simétrica hace posible precalcular los cuadrados de los pesos y reducir el número de términos de los productos, ya que sólo los términos elevados al cuadrado proporcionan valores diferentes de cero.

AII.1. Conclusiones

Se ha mostrado un resumen de los fundamentos del algoritmo usando diferentes niveles de abstracción (primeramente una explicación descriptiva, así como sus parámetros teóricos más relevantes), después con un diagrama de flujo organizado jerárquicamente.

Posteriormente se ha mostrado un organigrama que clasifica jerárquicamente el algoritmo puramente *software* con vista a su implementación.

En la ultima parte se han examinado como degradar el modelo, sus efectos y como ahorrar diversas operaciones aprovechando las simetrías evidentes encontradas en las ecuaciones teóricas.

APÉNDICE III

Justificación biológica y psicofísica visual, del modelo de gradiente multicanal

AIII.1. La naturaleza como modelo

La naturaleza a través del proceso de evolución, ha desarrollado mecanismos para el procesamiento de la información visual. Esos algoritmos han sido diseñados de forma que maximizan la supervivencia en el mundo real, siendo las técnicas abordadas en este trabajo de investigación, eficientes, robustas y prácticas para su implementación en la arquitectura paralela del cerebro.

La percepción visual humana y del primate son las más desarrolladas, con un alto porcentaje de tejido cortical destinado a interpretar la señal visual. Se van a examinar las evidencias neurofisiológicas y psicofísicas encontradas para imitar los algoritmos usados por la biología, y posteriormente construir sistemas que incorporen algunos de los tratamientos de visión natural.

La percepción de movimiento es fundamental para sobrevivir, no es sorprendente, pués, que haya áreas en el córtex visual cuya única función sea detectar movimiento. [HUB65]. Este procesamiento de movimiento, requiere comprender procesamiento espacial y temporal de la imagen. Consecuencia de que las neuronas en el sistema visual sumen información sobre el tiempo además de sobre el espacio.

Relativo al córtex visual, hay que decir que se refiere tanto al córtex visual primario (conocido como córtex estriado o V1) como a las áreas extra-estriadas corticales llamadas V2,V3,V4,V5. El córtex visual primario es anatómicamente equivalente al área 17 de Brodmann (BA17), según el mapa histológico del cerebro humano.

Estas áreas han sido examinadas anatómicamente en primates y visualizadas en el hombre (mediante técnicas como PET) [ZEK93]. Concluyendo que el área cortical V5/MT (córtex temporal medio), contiene células selectivas respecto a las dirección para un estímulo

de movimiento. El resultado de combinar filtros espaciales y temporales que operan sobre la retina es conformar señales que dependan de la velocidad. Donde hay fuertes sospechas a favor de la ubicación de estos filtros en las áreas V1y V2 (áreas visuales primitivas), concretamente en la periferia del “Sulcus Calcanei” en el lóbulo occipital.

La forma en que el sistema visual combina la información espacio-temporal para extraer velocidad es un tema de investigación actualmente muy activo, existiendo numerosos problemas abiertos, como por ejemplo cómo se modelan los requerimientos de multiplexación tan elevados para poder transmitir todas las propiedades del mapa de saliencia ya que la retina del primate contiene 130 millones de fotorreceptores, sin embargo sólo hay alrededor de un millón de células ganglionares retinianas que forman el nervio óptico.

La activación de las células corticales dependen de un numero de estímulos, asumiendo éstos parámetros independientes como brillo, contraste o longitud de onda tener una influencia multiplicativa y separable sobre la actividad neuronal, se puede conseguir separar selectivamente estas señales a través de un cociente. Este cociente permitirá al sistema visual llegar a ser selectivo sólo para movimiento, un requisito indispensable para la constancia de percepción de movimiento.

El algoritmo presentado en este trabajo de investigación está basado en los modelos de gradientes que calculan movimiento mediante un cociente de derivadas de la imagen de entrada respecto al espacio y el tiempo. Se ha indicado a lo largo de esta memoria que este cociente puede estar mal condicionado de forma que se obtengan divisiones por cero, para resolver este problema se acude a las evidencias proporcionadas por el procesamiento visual del primate, en particular la forma del banco espacio-temporal de filtros que forma la etapa inicial del procesamiento visual, como se indica en el siguiente apartado.

AI.2. Evidencia biológica: Tipos de filtros encontrados

Es importante que el modelo de movimiento desarrollado recoja conocimientos de la biología y la fisiología cortical. Este modelo está basado en una estructura de operadores diferenciales espaciales y temporales, pero, ¿hay evidencia de que el sistema visual tiene acceso a estas operaciones?.

AIII.2.1 Filtros espaciales.

Dentro del dominio espacial, la forma de los campos receptivos pertenecientes a la células del córtex visual, puede modelarse con funciones Gaussianas [YOU86]. A medida que el orden de diferenciación se incrementa, estas funciones llegan a estar sintonizadas a frecuencias espaciales más altas dando un rango de canales espaciales independientes que han sido verificados experimentalmente.(No se comenta más sobre las Gaussianas y sus derivadas porque han sido explicadas y utilizadas activamente desde el capítulo III en el presente trabajo).

Se pasa a estudiar una nueva alternativa para el modelado de filtros espaciales, las funciones de Gabor, sigue habiendo una gran controversia entre las bondades de estos filtros y las Gaussianas, como ejemplo Young presenta evidencia de que la distribución de cruces por cero encontrada en campos receptivos de gatos y primates se describe mejor por derivadas de Gaussianas que por las funciones de Gabor. [YOU86].)

Las funciones de Gabor son funciones de variable compleja resultantes del producto de una función Gaussiana bidimensional por una modulación sinusoidal.

En la figura ApIII.1 se observa el perfil unidimensional de la función de Gabor bidimensional, la sección positiva (la que sobresale hacia arriba) se corresponde con la porción ON del campo receptivo, mientras que la sección negativa (la que está por debajo de la línea media) se corresponde con la porción OFF. (En la figura ApIII.2 se tiene los perfiles de la función compleja de Gabor)

$$g_{0,0,f_0,\theta_0}(x,y) = e^{-\pi a^2(x^2+T^2y^2)}e^{i2\pi f_0 x}e^{i\phi}$$

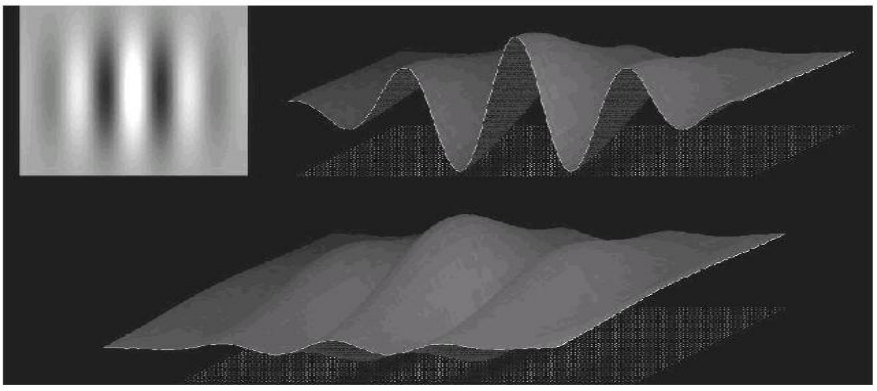


Figura ApIII.1. Esquema donde podemos apreciar la función de Gabor, tenemos la expresión algebraica (arriba) y sendas representaciones en perspectiva y en 2D, se ve claramente el efecto modulador de la sinusoide sobre la Gaussiana..

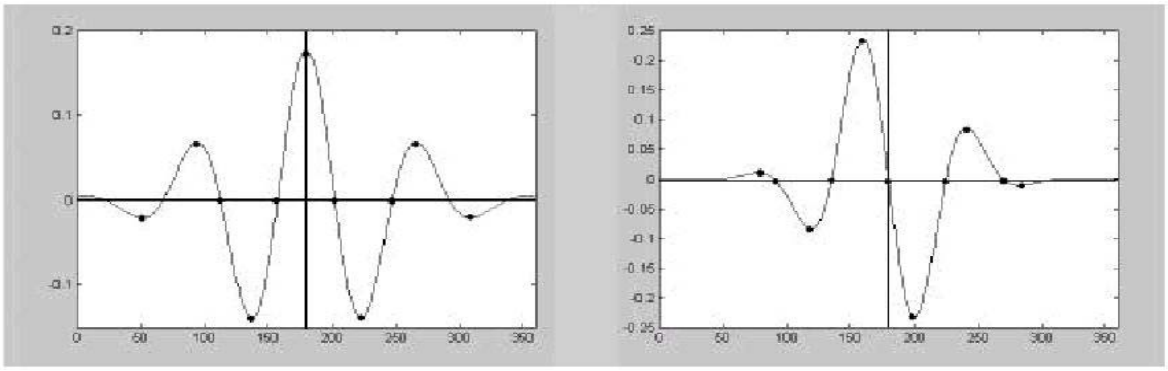


Figura ApIII.2 Perfiles de la función compleja de Gabor. Izquierda: parte real (función simétrica). Derecha: parte imaginaria (función asimétrica).

Hubel y Wiesel [HUBEL62] observaron que los perfiles unidimensionales de los campos receptivos asociados a las células simples eran de dos tipos: células simples de campo simétrico y células simples de campo asimétrico. Según Jones y Palmer (1987) [JONES_87]**,** cada célula Simple tiene asociados 2 Campos Receptivos retinianos adyacentes, con distinto perfil unidimensional (simétrico o asimétrico). Por otra parte, la función de Gabor es una función continua definida para el campo de los números complejos y, por tanto, consta de una parte real (perfil simétrico) y otra parte imaginaria (perfil asimétrico).

[HUB65] [HUB68] [HUB69] quienes, durante las décadas de los 60 y 70, desvelaron la arquitectura fisiológica de V1 y la describieron según una organización columnar, donde cada columna está sintonizada a una cierta orientación. [MAF77] mostraron que también subyacía, simultáneamente, otra organización laminar, donde cada lámina estaba sintonizada a una determinada frecuencia espacial, de modo que el córtex visual se comporta, a su vez, como un analizador de frecuencias espaciales.

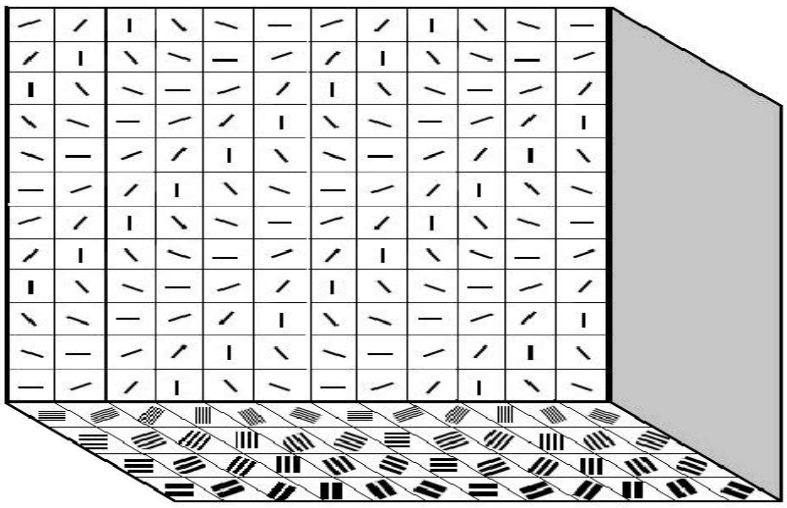


Figura ApIII.3 Esquema que refleja la disposición de las columnas de orientación y las láminas Adaptado

En la figura Ap.III.3 se muestra un esquema simplificado de un módulo visual, que cubriría unos 2 mm^2del córtex visual primario (V1) y que se ajusta a esta descripción neurofisiológica. Esquema que refleja la disposición de las columnas de orientación y las láminas de frecuencia espacial del modelo neurofisiológico que permite modelar una superficie de $2mm^2$ de V1 (área visual primaria).Cada columna de orientación responde, preferentemente, a una orientación representada mediante una línea inclinada en la cara frontal del bloque y cada capa laminar está sintonizada, óptimamente, a la frecuencia espacial representada en la base inferior del bloque. [MAF77].

AIII.2.2 Filtros temporales:

Se ha reseñado brevemente que gracias a varios experimentos se conoce la existencia de canales selectivos de frecuencias espaciales en el sistema visual humano, pero sin embargo ¿cuántos canales temporales tiene el ser humano?.

Hess y Snowden [HESS92] investigaron el procesamiento visual usando un paradigma de máscara. La *prueba* (una retícula que invertía el contraste a una frecuencia particular) se consideró como un umbral de detección óptimo. Posteriormente, midieron el contraste de una *máscara* (una banda estrecha que filtraba ruido espacial y a su vez invertía el contraste para un rango de frecuencias temporales) en la cual la *prueba* podría ser detectada.

Lo razonable del experimento es que si la *prueba* y la *máscara* son detectadas por canales temporales independientes, la *máscara* no interferirá nunca en la detección de la *prueba* .

Gracias e este experimento se encontró evidencias de 3 canales temporales

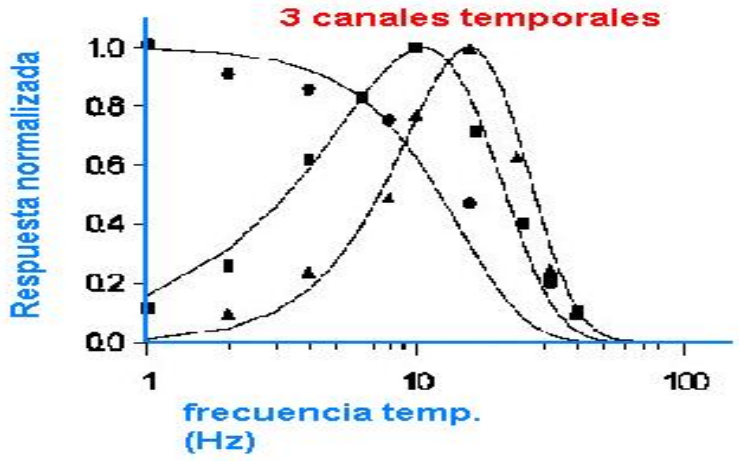


Figura ApIII.4 Esquema donde aparecen un ajuste a los 3 filtros temporales encontrados en el ser humano (2 filtros paso banda y un paso baja)

- Un canal paso baja
- Un canal paso banda con frec central aprox a 10 Hhz.
- Otro canal paso banda con frec central a 18 Hz.

Esos 3 canales pueden perfectamente ser modelados por un proceso de diferenciación temporal de una función causal (log(Gaussiana)) y se muestran en la fig ApIII.4 [JOH95]

AIII.3. Resumen: Derivadas difusas y ajustes de datos para modelos:

Por lo tanto se tiene un esqueleto de las áreas corticales V1 y V2, donde el brillo de la imagen se puede representar en la forma de un desarrollo en serie truncado de Taylor. Esta aproximación asume que el córtex visual representa la señal visual en términos de las propiedades geométricas locales de la superficie S(x,t,I(x,t)), formada a su vez por los valores del brillo de la imagen espacio-temporal. Esta representación proporciona una forma intuitiva de hacer predicciones del modelo de gradiente [JOH95] considerando la dirección en la cual un objeto cualquiera en movimiento describiría la superficie

Por lo tanto se tiene un esqueleto de las áreas corticales V1 y V2, donde el brillo de la imagen se puede representar en la forma de un desarrollo en serie truncado de Taylor. Esta aproximación asume que el córtex visual representa la señal visual en términos de las propiedades geométricas locales de la superficie S(x,t,I(x,t)), formada a su vez por los valores del brillo de la imagen espacio-temporal. Esta representación proporciona una forma intuitiva de hacer predicciones del modelo de gradiente [JOH95] considerando la dirección en la cual un objeto cualquiera en movimiento describiría la superficie S.

La superficie del gradiente será siempre ortogonal a los contornos de igual brillo. Estos datos fisiológicos y psicofísicos dan una prueba de que el sistema visual implementa un proceso de filtrado usando derivadas difusas, evidencia que se usa en la estructura del modelo. En el dominio espacial la derivada difusa del modelo tiene sólo un parámetro que ajustar, la varianza de la Gaussiana. y la forma de los filtros temporales sin embargo se fija con 2 parámetros. (*v* capítulo III).

El modelo detecta correctamente al dirección de movimiento en tipos de movimiento de primer orden (patrones de luminancia definida) y en segundo orden (patrones de contraste definido), sin embargo, falla detectando movimiento cuando están intercalados los dos tipos de patrones [BOT02]. Estando estos resultados perfectamente de acuerdo con los datos psicofísicos. También se encontró una explicación unificada de un número de ilusiones de

movimiento aparente (fenómeno phi, ilusión de Pantle, Pinna-Brelstaff). además de la detección de señales de 2º orden, como modulaciones de contraste y ruido modulado limitado en banda. [JOH95]. Se puede concluir que el modelo considerado en esta trabajo incorpora distintas etapas de procesamiento e integra las salidas de los filtros orientados que tienen un rango determinado de frecuencia espacial y temporal, consistente con las propiedades de las neuronas corticales.

AIII.4. Extensiones futuras, aprovechando estas propiedades:

Como extensión natural del McGM se ha elaborado una metodología que puede extenderse a otros dominios visuales. La detección de movimiento, orientación espacial, disparidad binocular y color pueden ser ubicadas en el espacio plenóptico.[ADE85].

Esta espacio o construcción describe la entrada desde el entorno del sistema visual, como una función multidimensional de la dirección, tiempo, longitud de onda, punto de vista y permite entonces una representación de las medidas básicas del proceso visual como la medida de la orientación en este espacio multidimensional tan particular. (de la misma forma que se ha efectuado en el capítulo II y capítulo III la medida de velocidad como una orientación en el espacio-tiempo).

Existe además una evidencia neurológica de que las células con los perfiles difusos están presentes en las áreas del cerebro cuya función está asociada a la orientación espacial y disparidad binocular, trabajando actualmente para modelar estos dos procesos perceptuales usando la misma filosofía que usa McGM. El mecanismo de transferir modelos desde un dominio plenóptico a otro, por ejemplo desde la energía de movimiento al estéreo se ha explorado en los últimos años, aunque esta tarea es ardua y difícil; pues una simple transferencia de modelos entre espacios no está soportada por evidencias psicofísicas, encaminándose las investigaciones de neurociencia (afines a este campo) a justificar estas transformaciones.

Sirva este breve apéndice como un pequeño reconocimiento a Torstein Wiesel y David Hubel, [HUB65][HUB68][HUB69] Premios Nobel de medicina en 1981, quienes dijeron “Los científicos conocen aproximadamente el 25 por ciento del funcionamiento del cerebro” precisando posteriormente que esta cifra es una estimación, pues “no se sabe aún donde está el final por descubrir”.

Las aportaciones de ambos sobre la capacidad del cerebro para analizar y procesar información visual fueron trascendentales para avanzar en el ámbito de la neurociencia, demostrando que las células de la corteza visual del cerebro son capaces de estimularse selectivamente a patrones de movimiento.

APÉNDICE IV
Ejemplos de aplicaciones reales

AIV.1. Breves ejemplos de estímulos reales

A continuación, se muestran brevemente algunas capturas de pantalla representativas de situaciones correspondientes a secuencias reales. (en las secuencias reales no es posible estimar el error cometido en el cálculo del flujo, debido al problema de la apertura).

En la figura AIV.1 se ilustra, un diagrama explicativo de cómo se codifica la fase de la velocidad (la orientación final) mediante colores , además se exponen cuatro estímulos con la tendencia global del movimiento (líneas rojas) superpuesta aproximadamente.

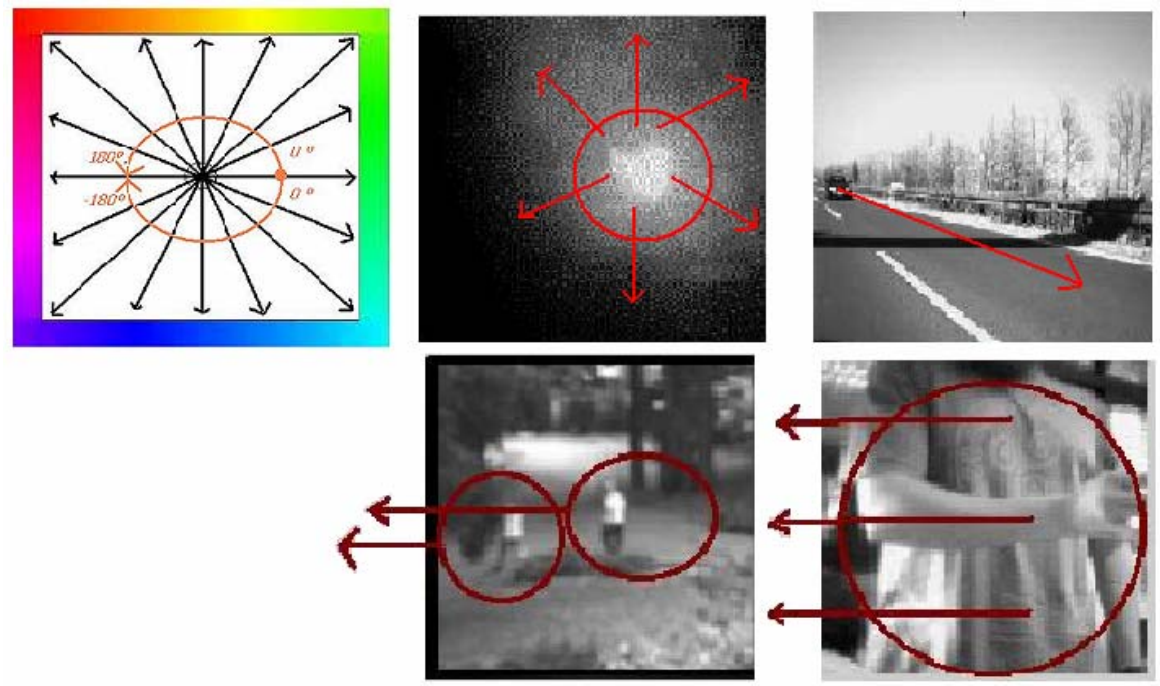


Figura AIV.1 (Superior Izq). Esquema de codificación por colores de la fase de de la velocidad.
(De arriba abajo y de izquierda a derecha), Explosión de gas expansiva, Secuencia de un coche donde destaca la sombra de una señalización superior, 2 personas caminando por el bosque, el autor de estas líneas

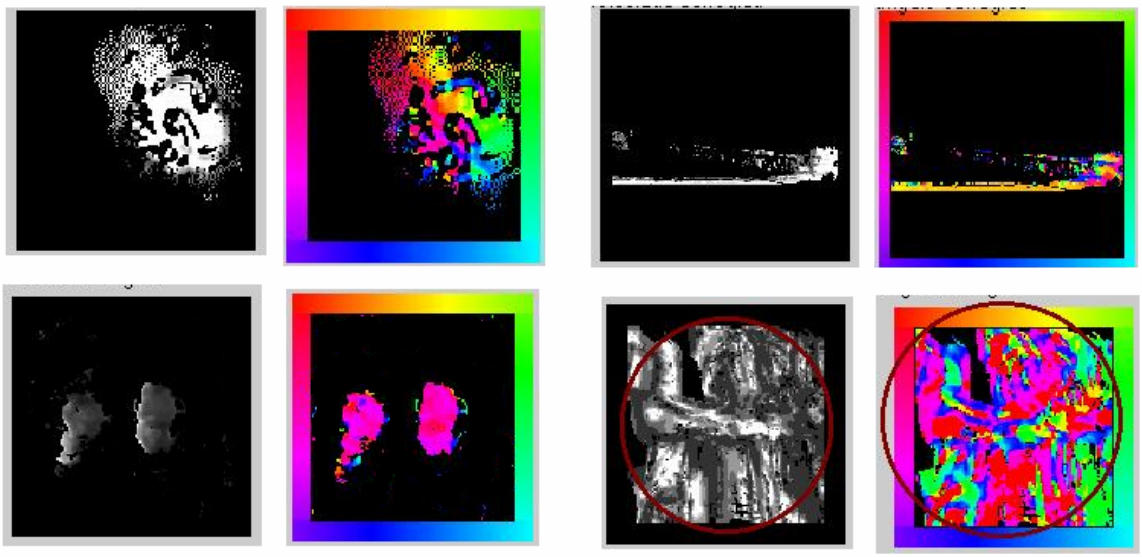


Figura AIV.2. Diagramas por pares, donde se muestra módulo y fase (izquierda y derecha) de cada grupo correspondiente a los 4 estímulos explicados en la figura AIV.1.

En la figura AIV.2 se ilustran los resultados de codificar fase (orientación) en color y módulo (intensidad de grises), para el algoritmo *software*, cada uno de los estímulos mostrados anteriormente. Llama la atención como la explosión de gas sigue la expansión en todo su recorrido angular, la sombra de una señalización sobre la carretera, sigue su movimiento hacia arriba, las dos personas caminando se segmentan perfectamente frente al resto inmóvil y una secuencia de baja resolución correspondiente a una *webcam* donde el se mueve el brazo y hemitórax hacía la derecha, apareciendo el movimiento más confuso y mezclado esta debido a los múltiples pliegues de la ropa, y la falta de contraste en las imágenes.

AIV.2. Pérdida de precisión en situaciones reales

Por último se muestra en la figura AIV.3 sendas estructuras espaciales piramidales explicadas en el capítulo III e implementadas a lo largo de todo este trabajo (hay que recordar que cada pirámide se codifica hacia la derecha en orden de diferenciación de x y hacia la izquierda en y) representando cada frente diagonal cada una de las derivadas de orden 0,1,2,3 Cada pirámide a su vez, representa un orden de diferenciación temporal.

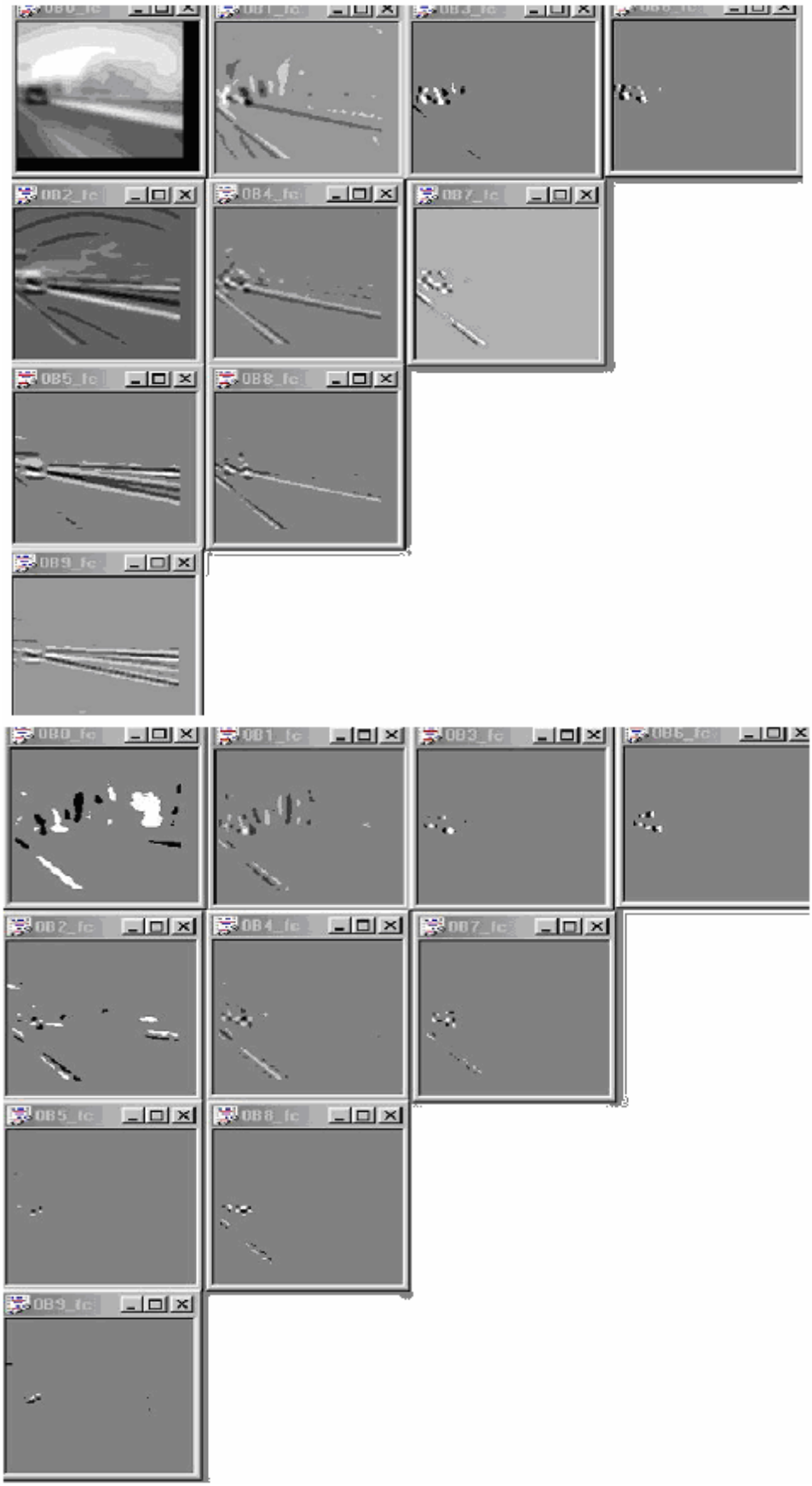


Figura AIV.3. Estructura de filtrado espacial piramidal correspondiente a la plataforma realizada en el capítulo IV , donde tenemos, derivada de orden cero temporal (superior) y derivada primera (inferior). Se aprecia claramente el efecto de la pérdida de precisión y como éste se acentúa a medida que el órden de derivación tanto espacial como temporal se incrementa.

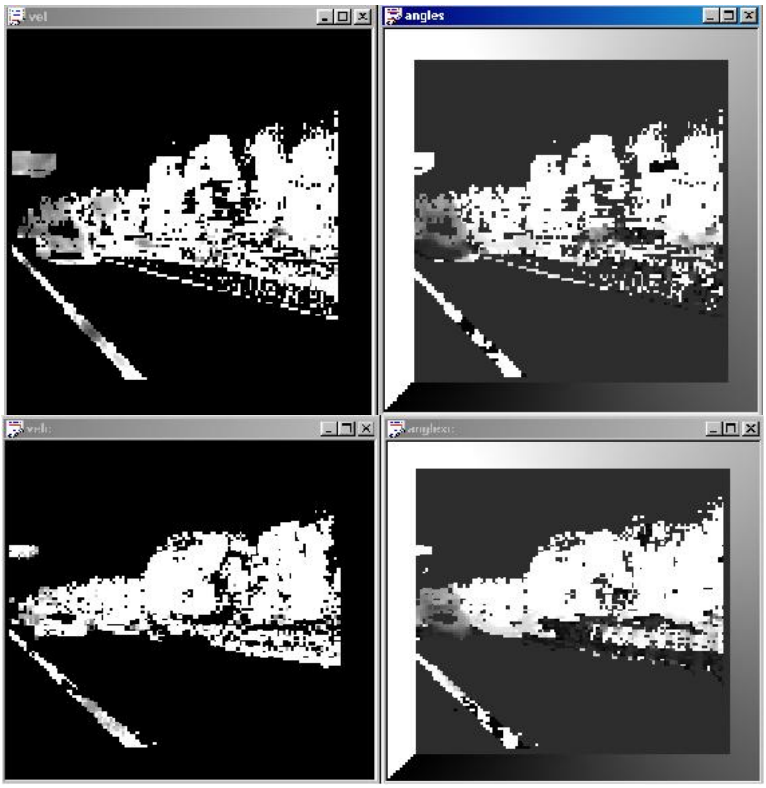


Figura AIV.4. Estructura final obtenida del sistema funcionando sobre la FPGA (esta vez se representan los colores on escala de grises, en la fase) Comparación del resultado de la plataforma semihardware (arriba), con el resultado de codiseño (abajo).

Se aprecia claramente el efecto de la pérdida de precisión y como éste se acentúa a medida que el orden de derivación tanto espacial como temporal se incrementa.
La figura AIV.3 representa la sálida en el capítulo IV (Plataforma de precisión controlada).

Por último se representa los resultados del proceso de co-diseño completo comparando en la figura AIV.4 la plataforma *semihardware* (pareja superior de salidas), frente a la obtenida la plataforma *hardware* (pareja inferior), hay que reseñar que se ha utilizado un umbral (medida de confianza explicado en el capítulo V), que filtra los datos en etapas tempranas del cauce que no cumplen una determinada condición relativa a las primeras derivadas temporales de la imagen.

BIBLIOGRAFÍA

[ACC98] Accame M., De Natale F. G. B., Giusto D. D., *High Performance Hierachical Block-based Motion Estimation for Real-Time Video Coding*, Real-Time Imaging 4, pp.67-79. (1998).

[ADE85] Adelson E. H., Bergen J. R., *Spatiotemporal Energy Models for the Perception of Motion*, Journal of the Optical Society of America A, 2(2)., pp.284-299. (1985).

[ALB93] Albright T. D., *Cortical Processing of Visual Motion*, In Miles & Wallman, Visual Motion and its Role in the Stabilisation of Gaze,pp.177-201. (1993).

[ALP06] http://www.alpha-data.com. (2006).

[AMD06] http://www.amdcompare.com. (2006).

[ANA89] Anandan P., A *Computational Framework and an Algorithm for the Measurement of Visual Motion*, International Journal of ComputerVision 2, pp.283-310. (1989).

[ANA93] Anandan P., Bergen J. R., Hanna K. J., Hingorani R., *Hierarchical Model-based Motion Estimation*. In M.I.Sean and R.L.Lagendijk,editors, Motion Analysis and Image Sequence Processing. Kluwer Academic Publishers. (1993).

[AND03] Anderson, A. & McOwan, P. *Humans deceived by predatory stealth strategy camouflaging motion*. Proceedings of the Royal Society, B 720, 18-20. (2003).

[ARI96] Arias-Estrada M., Tremblay M., Poussart D., *A Focal Plane Architecture for Motion Computation*, Real-Time Imaging 2,pp.351- 360. (1996).

[ARN93] Arnspang J. *Motion Constraint Equations Based on Constant Image Irradiance*, Image and Vision Computing 11(9)., pp.577-587. (1993).

[ARR04] Arredondo MA, Lebart K, Lane D *Optical flow using textures* . Pattern Recognition Letters 25, (4) pp 449-457. (2004).

[BAR94] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, *Performance of optical flow techniques,* International Journal of Computer Vision, vol. 12, no. 1, pp. 43-77,(1994).

[BAR96] Barr T. *Vector Calculus*, Prentice Hall, ISBN:0134000374.(1996).

[BEA95] S. S. Beauchemin, J. L. Barron. *The computation of optical flow*. ACM Computing Surveys (ACMCS1995) 27(3):433-467. (1995).

[BEA99] Beare R., Bouzerdoum A., *Biologically inspired Local Motion Detector Architecture*, Journal of the Optical Society of America A 16(9), pp.2059-2068. (1999).

[BER92] Bergen J. R., Burt P. J., *A Three-Frame Algorithm for Estimating Two-Component Image Motion*, IEEE Transactions On Pattern Analysis and Machine Intelligence 14(9)., pp.886-895. (1992).

[BOL97] Bolduc M., Levine M. D., *A Real-Time Foveated Sensor with Overlapping Receptive Fields*, Real-Time Imaging 3, pp.195- 212. (1997).

[BOT02] Botella G. *Estudio de un modelo de visión artificial bioinspirado*. Proyecto fin de carrera. Biblioteca de la facultad de ciencias.(2002).

[BOT03] Botella G, Ros E. Deliverable*: Best Motion Algorithm*. ECOVISION working Group. http://www.pspc.dibe.unige.it/ecovision/pubs/index.html. (2003).

[BOT04] Botella G, Ros E, Rodríguez M, García A, Romero S, Parrilla, L. *Modulo de procesamiento espacio-temporal robusto multicanal en hardware reconfigurable*. IV JCRAs. ISBN: 8-4688-7667-4. (2004).

[BOT06a] Botella G, Ros E, Rodríguez M, García A, Romero, S. *Pre-processor for bioinspired optical flow models:a customizable hardware implementation.* IEEE XIII MELECON Conference. ISBN: 1-4244-0088-0. (2006).

[BOT06b] Botella G, Ros E, Rodríguez M, García A. *Bioinspired robust optical flow in a FPGA system.* XXXII EUROMICRO Conference. ISBN: 3-9024-5711-2. (2006).

[BUR07] Burden R, Faires Douglas.J. *Analisis Numérico*. ISBN 9-7068-6608-6 Thomson Paraninfo. (2007).

[BUX95] Buxton H., Gong S., *Visual Surveillance in a Dynamic and Uncertain World*, Artificial Intelligence 78, pp431-459. (1995).

[CAM92] Campani M., Verri A., *Motion Analysis from First-Order Properties of Optical Flow*, CVGIP: Image Understanding 56(1)., pp.90-107. (1992).

[CEL06a] http://www.celoxica.com. (2006).

[CEL06b] http://www.celoxica.com/products/dk/default.asp. (2006).

[CEL06c] http://www.celoxica.com/technology/c_design/handel-c.asp.(2006).

[CHO02] K. Compton and S. Hauck, *Reconfigurable Computing: A Survey of Systems and Software*, ACM Computing Surveys (CSUR), Vol. 34, No. 2, pp. 171-210, (2002).

[CHR98] Christmas W. J., *Spatial Filtering Requirements for Gradient- Base Optical Flow Measurements*, Proceedings of the British Machine Vision Conference, pp.185-194. (1998).

[CM502] http://en.wikipedia.org/wiki/Thinking_Machines. (2002).

[COB98] P. Cobos, and F. Monasterio. *FPGA implementation of the Horn & Shunk Optical Flow Algorithm for Motion Detection in real time Images,* Proc. Of the XIII Design of Circuits and Integrated Systems Conference, Madrid, Spain., pp. 616-621.(1998).

[COL02] Collett T. S., *Insect Vision: Controlling Actions through Optic Flow*, Current Biology 12, pp.615-617. (2002).

[COR06] http://www.imaging.com; http:// www.dalsa.com. (2006).

[DAN98] Daniilidis K., Krauss C., Hansen M., Sommer G., *Real-Time Tracking of Moving Objects with an Active Camera*, Real- Time Imaging 4, pp.3-20. (1998).

[DEF95] Defaux F., Moscheni F., *Motion Estimation Techniques for Digital TV: A Review and a New Contribution*, Proceedings of the IEEE 83(6), pp.858-876 (1995).

[DEP04] http://atc.ugr.es/ficha_proyecto.php?id_proyecto=4. (2004).

[DER93] Deriche R . *Recursively Implementing the Gaussian and its derivatives.* INRIA Report 1893. ftp:\\ ftp.inria.fr/INRIA/tech-reports/RR/RR-1893.ps.gz . (1993).

[DEU98] Deutschmann R. A., Koch C., *Compact Analog VLSI 2-D Velocity Sensor*, Proceedings of the IEEE International Conference on Intelligent Vehicles, pp.359-364. (1998).

[DEV88] De Valois, R.L , De Valois, K.K. *Spatial Vision*. Oxford University Press. (1988).

[DIA97] Dias J., Araujo H., Paredes C., Batista J., *Optical Normal Flow Estimation on Log-polar Images. A Solution for Real-Time Binocular Vision*, Real-Time Imaging 3, pp.213-228. (1997).

[DIA06] J. Díaz, E. Ros, F. Pelayo, E. M. Ortigosa and S. Mota, "*FPGA based real-time optical-flow system*," IEEE Transactions on Circuits and Systems for Video Technology, vol. 16, no. 2, pp. 274-279, (2006).

[DOW87] Dowling, J. E. *The Retina: An Approachable Part of the Brain*. Cambridge, MA: Harvard University Press. (1987).

[EBE97] C. Ebeling, D. Cronquist and P. Franklin, "*Configurable Computing: TheCatalyst for High-Performance Architectures*," Proc. IEEE Intl. Conf. on66 Application-Specific Systems, Architectures and Processors, pp. 364 – 372.(1997).

[ECO03] Ecovision Consortium http://www.pspc.dibe.unige.it/ecovision/pubs/index.html.

[ENK91] Enkelmann. W., *Obstacle Detection by Evaluation of Optical Flow Fields from Image Sequences*., Image and Vision Computing 9(3)., pp.160-168. (1991).

[FLE89] Fleet D. J., Jepson A. D., *Hierarchical Construction of Orientation and Velocity Selective Filters*, IEEE Transactions on Pattern Analysis and Machine Intellignece 11(3), pp.315-325. (1989).

[FLE92] D. J. Fleet, *Measurement of Image Velocity*, Norwell, MA: Engineering and Computer Science, Kluwer Academic Publishers. (1992).

[FLE00] Fleet D. J., Black M. J., Yacoob Y., Jepson A. D., *Design and Use of Linear Models for Image Motion Analysis*, International Journal of Computer Vision 36(3)., pp.171-193. (2000).

[FRA92] Franceschini N., Pichon J. M., Blanes C., *From Insect Vision to Robot Vision*, Philosophical Transactions of the Royal Society of London B 337, pp.283-294. (1992).

[GAL98] B. Galvin, B. McCane, K. Novins, D. Mason and S. Mills, *Recovering Motion Fields: An Evaluation of Eight Optical Flow Algorithms,* in Proceedings of theBritish Machine Vision Converence (BMVC) '98, (1998).

[GHK91] M. Gokhale, W. Holmes, A. Kopser et al., *Building and Using a Highly Parallel Programmable Logic Array*, Computer, Vol. 24, No. 1, pp. 81-89.(1991).

[GHO97] Ghosal S., Mehrotra R., *Robust Optical Flow Estimation Using Semi-Invariant Local Features*, Pattern Recognition 30(2., pp.229- 237. (1997).

[GIA97] Giaccone P. R., Jones G. A., *Feed-Forward Estimation of Optical Flow*, Proceedings of the IEE International Conference on Image Processing and its Applications, pp.204-208. (1997).

[GIA98] Giaccone P. R., Jones G. A., *Segmentation of Global Motion using Temporal Probabilistic Classification*, Proceedings of the British Machine Vision Conference, pp.619-628. (1998).

[GIB50] Gibson, J. J. *Perception of the Visual World* (Houghton Mifflin, Boston, 1950).

[GOL97] Golland. P., Bruckstein A. M., *Motion from Colo*r, Computer Vision and Image Understanding 68(3)., pp.346-362. (1997).

[GUP95] Gupta N. C., Kanal L. N., 3-D *Motion estimation from Motion Field*, Artificial Intelligence 78, pp.45-86. (1995).

[GUP97] Gupta N. C., Kanal L. N., *Gradient Based Image Motion Estimation without Computing Gradients*, International Journal of Computer Vision 22(1), pp.81-101. (1997).

[HAR98] Harrison R. R., Koch C., *An Analogue VLSI Model of the Fly Elementary Motion Detector*, In Advances in Neural Information Processing Systems 10, MIT Press, Cambridge MA, pp.880-886. (1998).

[HAT98] Van Hateren J. H., Ruderman D. L., *Independent Component Analysis of Natural Image Sequences Yield Spatio-Temporal Filters Similar to Simple Cells in Primary Visual Cortex*, Proceedings of the Royal Society of London B 265, pp.2315- 2320. (1998).

[HAW97] J. Hauser and J. Wawrzynek, *Garp: A MIPS Processor with aReconfigurable Coprocessor*, Proc. IEEE Symp. on FPGAs for Custom ComputingMachines, pp. 12–21, (1997).

[HEL62] Helmholtz H. L. F. von., *Treatise on Physiological Optics*, C. Southall, Ed. New York: Dover Publications.(1962).

[HEI93] Heitz F., Bouthemy P., *Multimodal Estimation of Discontinuous Optical Flow Using Markov Random Fields*, IEEE Transactions on Patterns Analysis and Machine Intelligence 15(12)., pp.1217-1232.(1993).

[HES89] Hess, R. H., Baker, C. L. & Zihl, J. 1989 *The `motion-blind' patient: low-level spatial and temporal Filters*. J. Neuroscience. 9, 1628-1640.(1989).

[HES92a] Hess, R. F. & Snowden, R. J. *Temporal properties of visual filters; number, shapes and spatial covariation*. Vision Research, 32, 47-60. (1992).

[HES92b] Hess, R. F. & Snowden, R. J. *Temporal frequency filters in the human peripheral visual field.* Vision Research, 32, 61-72. (1992).

[HIG01] Higgins C. M., *Sensory Architectures for Biologically Inspired Autonomous Robotics*, Biological Bulletin 200(2)., pp.235- 242. (2001).

[HFH97] S. Hauck, T. Fry, M. Hosler et al. *The Chimaera Reconfigurable Functional Unit*, Proc. IEEE Symp. on FPGAs for Custom Computing Machines, pp. 87–96, (1997).

[HIL84] Hildreth E. C., *The Computation of the Velocity Field*, Proceedings of the Royal Society of London B 221, pp.189-220 .(1984).

[HMO03] H. Hunter and J. Moreno, *"A New Look at Exploiting Data Parallelism in Embedded Systems,"* Intl. Conf. on Compilers, Architecture and Synthesis forEmbedded Systems (CASES), pp. 159-169.(2003).

[HOP86] Hopfield J.J, Tank D.W. *Computing with neural circuits: a model*. Science 233, 625-633. (1986).

[HOP06] http://en.wikipedia.org/wiki/Hopfield_net.(2006).

[HOR81] Horn B. K. P., Schunck B. G., *Determining Optical Flow*, Artificial Intelligence 17, pp.185-203.(1981).

[HUA95] Huang C. and Chen. Y., *Motion Estimation Method Using a 3D Steerable Filter*. Image and Vision Computing 13(1)., pp.21-32.(1995).

[HUB65] Hubel. D. H., Wiesel T. N., *Receptive Fields and Functional Architecture in Two Non-Striate Visual Areas* (18 and 19). of the Cat., Journal of Neurophysiology 28, pp.229-289. (1965).

[HUB68] Hubel. D. H., Wiesel T. N. *Receptive fields and functional architecture of monkey striate cortex.* Journal of Physiology, 195, pp. 215-243. (1968).

[HUB77] Hubel. D. H., Wiesel T. N. *Sequence regularity and geometry of orientation columns in the monkey striate cortex*. J. Comp. Neurol. 158, 267-294.(1977).

[HUB79] Hubel. D. H., Wiesel T. N. *Mecanismos cerebrales de la visión.* Investigación y Ciencia, *38* , 100-114.(1979).

[IFE93] Ifeachor E.C. Jervis B.W. *Digital signal processing* Addison-Wesley ISBN 0-201-54413-X. (1993).

[IND99] Indiveri. G. Neuromorphic *Analog VLSI Sensor for Visual Tracking: Circuits and Application Examples*. IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing 46(11)., pp.1337-1347. (1999).

[IND00] Indiveri G., Dougla R., Neuromorphic *Vision Sensors*, Science 288, pp.1189-1190.(2000).

[INT06] http://www.intel.com.(2006).

[JIN97] Jin J. S. Gao Y .*Recursive Implementation of LoG Filtering*. Real-Time Imaging 3, pp 59-65.(1997).

[JOH92] Johnston, A., McOwan, P.W. and Buxton, H. *A computational model of the analysis of first and second order motion by simple and complex cells. Proc. R. Soc. Lond. B.*, 250, 297-306.(1992).

[JOH94] Johnston A., Clifford C. W. G., *A Unified Account of Three Apparent Motion Illusions*, Vision Research 35(8)., pp.1109- 1123.(1994).

[JOH95] Johnston A., Clifford C. W. G . *Perceived Motion of Contrast modulated Gratings: Prediction of the McGM and the role of Full-Wave rectification*, Vision Research 35, pp.1771-1783. (1995).

[JOH99] Johnston A., McOwen P. W., Benton C., *Robust Velocity Computation from a Biologically Motivated Model of Motion Perception*, Proceedings of the Royal Society of London B 266, pp.509-518. (1999).

[JOH03] Johnston, A., McOwan, P.W. and Benton, C.P. *Biological computation of image motion from flows over boundaries*, J Physiol. Paris, 97, 325-34.(2003).

[KOE86] Koenderink J. J., *Optic Flow*, Vision Research 26(1), pp.161-180. (1986).

[LIU97] Liu H., Hong T., Herman M., *A General Motion Model and Spatio-Temporal Filters for Computing Optical Flow*, International Journal of Computer Vision 22(2)., pp.141-172. (1997).

[LIU00] Liu S., Usseglio-Viretta A., *Visuo-motor Fly-like Responses of a Robot using a VLSI Motion-sensitive Chips*, Proceedings of the Second International ICSC Symposium on Neural Computation. (2000).

[LUC81] Lucas B. D., Kanade T., *An Iterative Image Registration Technique with an Application to Stereo Vision*, Proceedings of the Seventh International Joint Conference on Artificial Intelligence, pp.674-679. (1981).

[MAF77] Maffei, L., Fiorentini, A. *Spatial frequency rows in the striate visual cortex.* Vision Research, Vol. 17 , pp. 257-264. (1977).

[MAH99] Mahzoun M. R., Kim J., Sawazaki S., Okazaki K., Tamura S., *A Scaled Multigrid Optical Flow Algorithm Based on the Least RMS Error between Real and Estimated Second Images*, Pattern Recognition 32, pp.657-670 (1999).

[MAR82] Marr D., *Vision*, W.H.Freeman. (1982).

[MAR01] Marino F., Stella E., Branca A., Veneziani N., Distante A., *Specialised Hardware for Real-Time Navigation,* Real-Time Imaging 7, pp.97-108. (2001).

[MAR06] J.L. Martín, A. Zuloaga, C. Cuadrado, J. Lázaro, and U. Bidarte, *Hardware implementation of optical flow constraint equation using FPGAs*, ComputerVision and Image Understanding, vol.98, no. 3, pp. 462-490. (2005).

[MAT02] http://www.matrox.com. (2006)

[MAT06] http://www.mathworks.com.. (2006).

[MAX06] Maxfield C. *The design warrior's guide to FPGAs.* Elsevier. (2006).

[MEA90] Mead C., *Neuromorphic Electronic Systems,* Proceedings of the IEEE 78(10)., pp.1629-1636. (1990).

[MIR96] E. Mirsky and A. DeHon, *MATRIX: A Reconfigurable Computing Architecture with Configurable Instruction Distribution and Deployable Resources* Proc. IEEE Symp. on FPGAs for Custom Computing Machines, pp.157-166, (1996).

[MIT87] Mitiche A., *Experiments in Computing Optical Flow with the Gradient-Based, Multiconstraint Method*, Pattern Recognition 20(2), pp.173-179. (1987).

[MIT96] Mitiche A., Bouthemy P., *Computation and Analysis of Image Motion: A Synopsis of Current Problems and Methods*, International Journal of Computer Vision 19(1)., pp.29-55. (1996).

[MIY98] T. Miyamori and U. Olukotun, *A quantitative analysis of reconfigurable coprocessors for multimedia applications,* Proc. IEEE Symp. on FPGAs for Custom Computing Machines, pp. 2-11. (1998).

[MOR00] http://www.eng.uci.edu/morphosys. (2005).

[NAG83] Nagel H., *Displacement Vectors Derived from Second-Order Intensity Variations in Image Sequences*, Computer Vision, Graphics and Image Processing 21, pp.85-117. (1983).

[NAK85] Nakayama K., *Biological Image Motion Processing: A Review*, Vision Research 25, pp.625-660. (1985).

[NEW88] Newsome, W.T & Pare, E.B .*A selective impairment of motion perception following lessons of middle temporal visual area (MT).* Journal of neuscience 8, 2201-2211. (1988).

[NIT04] H. Niitsuma, and T. Maruyama, "Real-Time Detection of Moving Objects," *Lecture Notes in Computer Science, FPL 2004*, vol. 3203, pp. 1153-1157. (2004).

[NYQ06] http://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem. (2007).

[OHO00] Oh. H, Lee H. *Block Matching algorithm based on a adpatative reduction of the search area for motion estimation*, Real Time Imaging 6, pp. 407-414. (2000).

[ONG99] Ong E. P., Spann M, *Robust Optical Flow Computation Based on Least-Median-of-Squares Regression*, International Journal of Computer Vision 31(1), pp.51-82. (1999).

[PAJ03] Pajares, G. De la Cruz, Jesús. *Vision por Computador (imágenes digitales y aplicaciones)*. ISBN 84-7897-472-5. RA-MA. (2003).

[PAS94] Pasternak, T. & Merigan, W. H. 1994 *Motion perception following lesions of the superior temporal sulcus in the monkey.* Cerebral. Cortex 4, 247-259.(1994)

[POW06] http://www.ibm.com/powerpc. (2006).

[REI61] Reichardt W., *Autocorrelation, a Principle for the Evaluation of Sensory Information by the Central Nervous System*. Sensory Communication, pp.303-317, (1961).

[REI00] Reinery L *et al Simulink-based Hw/Sw codesign of embedded neuron-fuzzy systems* . International Journal of Neural Systems, Vol. 10, No. 3.211-226. (2000).

[RIV06] F. Rivera, M. Sánchez-Élez, M. Fernández, R. Hermida and N. Bagherzadeh, *Configuration scheduling for conditional branch execution onto multi-context reconfigurable architectures***,** Intl. Conf. on Field Programmable Logic and Applications (FPL). (2006).

[ROS98] P.L. Rosin, *Thresholding for Change Detection,* Proceedings of the International Conference of Computer Vision, pp.274- 279. (1998).

[SAN06] M. Sanchez-Elez, M. Fernandez, N. Bagherzadeh, R. Hermida, *A Low Energy Data Management for Reconfigurable Multi-context Architectures,* chapter in "New Algorithms, Architectures, and Applications for Reconfigurable Computing", Kluwer. (2005).

[SCA99] Scassellati B., *A Binocular, Foveated Active Vision System,* A.I. Memo No 1628, MIT Artificial Intelligence Laboratory. (1999).

[SHI03] Che-Shing . C, Chen Ming-Chieh C. *Recursive block-matching principle for error concealment algorithm.* Proceedings of International Symposium on Circuits and Systems 2003 Volume: 2,528-531. (2003).

[SHS02] G. Smit, P. Havinga, L. Smit et al., *Dynamic Reconfiguration in Mobile Systems* Intl. Conf. on Field Programmable Logic and Applications (FPL), pp. 171–181. (2002).

[SIM91] Simoncelli E. P., Adelson E. H., Heeger D. J., *Probability Distributions of Optical Flow,* Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (1991).

[SMI95] Smith S. M., *ASSET-2: Real-Time Motion Segmentation and Object Tracking*, Technical Report TR95SMS2b available from www.fmrib.ox.ac.uk/~steve. (1995).

[SIM98] Simoncelli E. P., Heeger D. J., *A Model of Neuronal Responses in Visual Area MT*, Vision Research 38(5)., pp.743-761. (1998).

[SOB91] Sobey P., and Srinivasan M. V., *Measurement of Optical Flow by a Generalized Gradient Scheme*, Journal of the Optical Society of America A 8(9)., pp.1488-1498. (1991).

[SON99] Sonka. M., Hlavac. V., Boyle. R., *Image Processing, Analysis, and Machine Vision*. Brooks/Cole Publishing Company. ISBN 0-534-95393-X. (1999).

[SPA06] http://es.wikipedia.org/wiki/Sun_SPARC.(2007).

[SPI71] Spiegel M. Calculus of Finite Differences and Differential equations. McGraw Hill. (1971).

[SPI88] Spiegel. M, Abellanas. L .*Fórmulas y tablas de matemática aplicada* . McGraw Hill. (1988).

[STA06] Stalling, W. *Organización y arquitectura de computadores: Diseño para optimizer prestaciones.* Pearson Educación. ISBN8420529931. (2006)

[STO03] M. Störring , T. Kocka, H.J Andersen and E.Granum .*Tracking regions of human skin through illumination changes.* Pattern Recognition Letters, 24 (11):1715-1723. (2003).

[SYS06] http://www.systemc.org .(2006).

[TCE95] E. Tau, D. Chen, I. Eslick et al. *A First Generation DPGA Implementation Proc.* Canadian Workshop on Field-Programmable Devices, pp. 138-143, May 1995.

[TEX06] http://www.ti.com .(2006).

[URA88] Uras S., Girosi F., Verri A., and Torre V., *A Computational Approach to Motion Perception*, Biological Cybernetics 60, pp.79-87. (1998).

[VER89] Verri A., Poggio T., *Motion Field and Optical Flow: Qualitative Properties*, IEEE Transactions on Pattern Analysis and Machine Intelligence 11, pp.490-498 . (1989).

[VLI98] van Vliet L. J., Young I.T, Verbeek P.M. *Recursive Gaussian Derivative Filters*. Proceedings of the XIV International Conference on Pattern Recognition pp. 509-514. (1998).

[WAH95] M. Wirthlin and B. Hutchings, *A Dynamic Instruction Set Computer* Proc. IEEE Symp. on FPGAs for Custom Computing Machines, pp. 99-107. (1995).

[WALL76] Wallach H., *On Perceived Identity: 1. The Direction of Motion of Straight Lines*, In On Perception, New York: Quadrangle. (1976).

[WAN99] Wandell B. A., *Computational Neuroimaging of Human Visual Cortex*, Annual Review of Neuroscience 22, pp.145-173. (1999).

[WEB94a] Weber K., Venkatesh S., Kieronska D., *Insect Based Navigation and its Application to the Autonomous Control of Mobile Robots*, Proceedings of the International Conference on Automation, Robotics and Computer Vision, (1994).

[WEB94b] Weber J., Malik J., *Robust Computation of Optical Flow in a Multi- Scale Differential Framework*, International Journal of Computer Vision 2, pp.5-19. (1994).

[WEB97] Weber K., Venkatesh S., Srinivasan M. V., *Insect Inspired Behaviours for the Autonomous Control of Mobile Robots*, in: From Living Eyes to Seeing Machines, Oxford University Press, pp.226-248. (1997).

[XIL06] http://www.xilinx.com. (2006).

[XIL07] http://www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex5/index.htm

[YAC99] Yacoob. Y. and Davis. L. S. *Temporal Multi-scale Models for Flow and Acceleration*. International Journal of Computer Vision 32(2), pp.1-17. (1999).

[YOU86] YOUNG, R.A. (1986). S*imulation of Human Retinal Function with the Gaussian Derivative Model*. IEEE Proc. Conference on Computer Vision and Pattern Recognition , pp. 564-569. (1986).

[YOU93] Young R. A., Lesperance R. M., *A Physiological Model of Motion Analysis for Machine Vision*, Technical Report GMR 7878. Warren, MI:General Motors Research Laboratories.

[YOU01] Young R. A., Lesperance R. M., *The Gaussian Derivative Model for Spatio-Temporal Vision: II. Cortical Data*, Spatial Vision 14(3., pp.321-389. (2001).

[ZAN96] Zanker J. M., *On the Elementary Mechanism Underlying Secondary Motion Processing*, Philosophical Transactions of the Royal Society of London B 351, pp.1725-1736. (1996).

[ZEK93] Zeki.S *A vision of the brain*, Blackwell Scientific Publications, Oxford. (1993).

[ZHA01] Zhang J. Z., Jonathan Wu. Q. M., *A Pyramid Approach to Motion Tracking*, Real-Time Imaging 7, pp.529-544. (2001).

[ZIH83] Zihl J. D. von Cramon *Selective Disturbance of Movement Vision after Bilateral Brain Damage*, Brain 106, pp.313-340. (1983).

[ZUL98] Zuloaga A., Bidarte U., Martin J., Ezquerra J., *Optical Flow Estimator Using VHDL for Implementation in FPGA*, Proceedings of the IEEE International Conference on Image Processing, pp.972-976. (1998).